DESIGN OF NUCLEAR POWER CONTROL SYSTEM BASED ON ARTIFICIAL NEURAL NETWORK LEVENBERG MARQUARDT AT THE NUCLEAR TECHNOLOGY CENTER FOR MATERIALS AND RADIOMETRY - NATIONAL NUCLEAR ENERGY AGENCY (PTNBR BATAN) BANDUNG

Name : Margaretha Maria Lucia Kloatubun
NRP : 2406 100 092
Departement : Physic Engineering FTI – ITS
Supervisor : Ir. Syamsul Arifin, MT.

Abstract

Fission reaction occurs in the inside of reactor tank between the nuclear fuel and neutrons. Fission reaction causes rise and fall of nuclear power reactors. In order to control the nuclear power reactor used five control rods which is moved up and down inside the reactor tank. There is a complex process and non-linear modeling in the reactor tank, conducted modeling reactor tank with Artificial Neural Networks with the structure of the Multi Layer Perceptron (MLP). The structure is a structure model derived from Nonlinear Auto Regressive with external input (NARX). Weights controlling of neural networks was carried out using Levenberg-Marquardt algorithm, which could give good results with RMSE and VAF good enough, ie 0.0209 and 98.8682. After getting the tank reactor model, then the reactor power control system designed using the Direct Inverse Control based on Artificial Neural Networks. The simulation results show a direct inverse control system based on neural networks have a good response. Direct inverse control follow the set point with maximum overshoot 0% for all set point.

Keywords: nuclear power, reactor tank, artificial neural networks.

PERANCANGAN SISTEM PENGENDALIAN DAYA REAKTOR NUKLIR MENGGUNAKAN JARINGAN SYARAF TIRUAN (JST) LEVENBERG MARQUARDT DI PUSAT TEKNOLOGI NUKLIR BAHAN DAN RADIOMETRI BADAN TENAGA NUKLIR NASIONAL (PTNBR BATAN) BANDUNG

Nama Mahasiswa : Margaretha Maria Lucia Kloatubun
NRP : 2406 100 092
Jurusan/Fakultas : Teknik Fisika-FTI-ITS
Dosen Pembimbing : Ir. Syamsul Arifin, MT.

Abstrak

Di dalam tanki reaktor terjadi reaksi fisi antara bahan bakar nuklir dengan neutron. Reaksi fisi inilah yang menyebabkan naik-turunnya daya reaktor nuklir. Untuk mengendalikan daya reaktor nuklir digunakan 5 (lima) batang kendali yang digerakkan naik-turun di dalam tanki reaktor. Oleh karena di dalam tanki reaktor terdapat proses yang kompleks dan non-linear maka pemodelan dilakukan dengan Jaringan Syaraf Tiruan dengan struktur *Multi Layer Perceptron (MLP)*. Struktur model yang diturunkan adalah struktur *Nonlinear Auto Regressive with eXternal input (NARX)*. Pengaturan bobot jaringan syaraf tiruan dilakukan menggunakan algoritma *Levenberg-Marquardt* yang mampu memberikan hasil dengan RMSE dan VAF cukup baik, yakni 0.0209 dan sebesar 98.8682. Setelah didapatkan model tanki reaktor, selanjutnya dirancang sistem kontrol daya reaktor dengan metode Direct Invers control berbasis Jaringan Syaraf Tiruan. Dari hasil simulasi menunjukkan sistem *direct invers control* berbasis Jaringan Syaraf Tiruan memiliki respon yang sangat baik. Pengendalian mengikuti set point dengan maximum overshoot 0 % untuk semua *set point.*

Kata kunci : daya reaktor, tanki reaktor, jaringan syaraf tiruan.

DAFTAR NOTASI

$R^{(l)}$	= Matriks Hessian
λ	= elemen pengendali konvergensi pada Algoritma Levernberg-marquadt
V_n	= kriteria sebenarnya (*true criterion)*
f	= fungsi aktivasi dari neuron
f_i	= fungsi aktivasi lapis tersembunyi
F_i	= fungsi aktivasi lapis *output*
φ_i	= *input* untuk neuron
$W_{i,j}$	= matrik bobot antara lapis tersembunyi ke lapis output
$W_{j,i}$	= matrik bobot antara lapis input kelapis tersembunyi
Wi,0	= bias
$Vn(\Theta,Z^{(N)})$	= Mean Square error
$\hat{y}(t)$	= output prediksi
y(t)	= output sistem
x(t)	= regressor
u(t)	= Input sistem
$Z^{(N)})$	= pasangan data input dan output
n_u	= history length untuk input sistem
n_y	= history length untuk output sistem

e(t) = error

N = Jumlah data

r(t+1) = set point

t_r = waktu naik

t_d = waktu tunda

t_p = waktu puncak

t_s = waktu turun

M_p = maksimum overshoot

P = daya reaktor (watt)

C = konsentrasi prekursor neutron

ρ = reaktivitas reaktor

Λ = waktu generasi neutron rata-rata

T_f = temperatur bahan bakar

T_c = temperatur pendingin

P_{eks} = reaktivitas ekstenal

λ = tetapan disintegrasi prekursor neutron

BAB II
TINJAUAN PUSTAKA

2.1 Pengendalian Daya Reaktor Nuklir

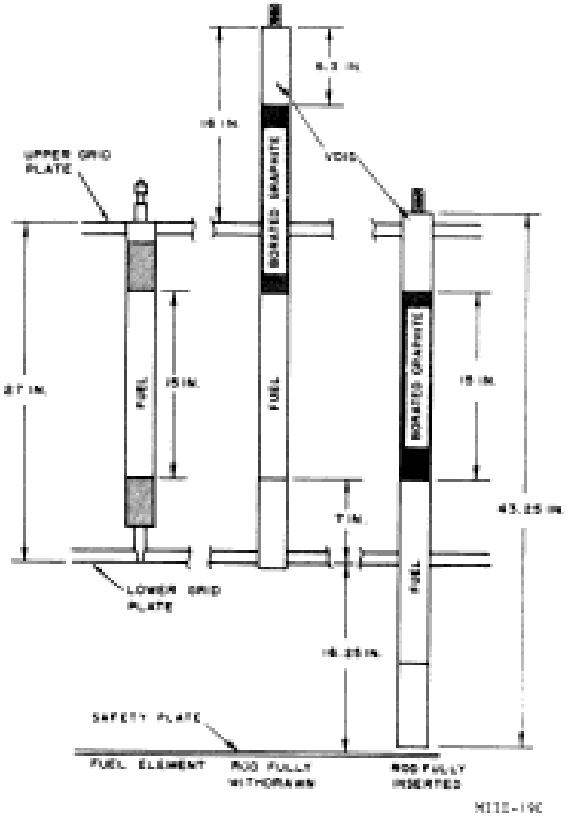
Pengendalian daya (reaktivitas) reaktor dilakukan dengan penyerapan neutron dengan batang kendali dan racun kimia-dapat-bakar *(pengendalian kompensasi kimiawi)*. Pengendalian kompensasi kimiawi dilakukan dengan melarutkan cairan asam borak (senyawa kimia penyerap neutron) ke dalam pendingin sistem primer. Konsentrasi asam ini diatur sedemikian rupa sehingga dapat mengendalikan proses penyerapan neutron yang menghambat reaksi fisi dalam teras reaktor. Jika asam borak ini menyerap neutron, unsur borak akan mengalami reaksi inti dan berubah manjadi unsur lain (terbakar). Oleh karena itu senyawa kimia seperti asam borak disebut sebagai racun-dapat-bakar *(burnable poison)*. Pelarutan cairan asam borak akan menyebabkan distribusi reaksi fisi (distribusi daya panas) menjadi rata pada seluruh teras sehingga menurunkan daya maksimum relatif dalam teras reaktor. Dalam sistem primer, jumlah untai (*loop*) dan jumlah perangkat pembangkit uap bergantung pada daya yang akan dibangkitkan. Pada reaktor air tekan terdapat perangkat pengatur tekanan sistem primer yang di dalamnya terdiri dari pemanas untuk menaikkan tekanan dan penyemprot air untuk menurunkan tekanan. Perangkat pengatur tekanan ini dapat mengendalikan tekanan, termasuk jika terjadi kenaikan tekanan pada sistem primer karena perubahan temperatur.

Hal penting yang perlu dicatat tentang hubungan antara reaktor dan turbin pembangkit listrik (turbin listrik: turbin dan generator listrik) adalah, daya reaktor harus selalu mengikuti beban listrik yang dipikul oleh turbin pembangkit listrik. Pada reaktor air tekan, berdasarkan prinsip koefisien
reaktivitas temperatur moderator (dalam hal ini adalah sama dengan pendingin sistem primer) daya reaktor dapat mengikuti beban yang dipikul oleh turbin pembangkit listrik. Jika beban listrik meningkat, maka diperlukan kenaikan jumlah

pembangkitan uap (kapasitas uap naik). Hal ini menyebabkan temperatur air pendingin yang masuk ke bejana tekan turun. Penurunan temperature pendingin primer akan menaikkan kemampuan moderasi neutron dan meningkatkan daya termal yang dibangkitkan reaktor. Fenomena ini secara otomatis menyebabkan reaktor melakukan pengendalian diri untuk mengikuti beban listrik, namun prinsip ini hanya akan bekerja dengan baik untuk perubahan beban listrik yang kecil. Bila beban listrik mengalami perubahan cukup besar, maka untuk mengikuti perubahan listrik digunakan cara pengendalian dengan mekanisme batang kendali. Dengan demikian dapat dikatakan bahwa pengendalian reaktor air tekan mengikuti prinsip "reaktor mengikuti beban turbin listrik".

Cara lain untuk menaikkan reaktivitas (daya reaktor) adalah dengan menarik batang kendali dari teras reaktor. Jika batang kendali ditarik keluar dari teras, reaktivitas atau reaksi fisi bertambah dan menghasilkan energi panas lebih banyak lagi (daya reaktor naik). Energi panas ini akan mendidihkan air lebih banyak, dan dengan demikian uap yang dihasilkan juga bertambah. Meningkatnya kandungan uap dalam air akan menurunkan kemampuan air dalam memoderasi partikel neutron. Jumlah neutron kecepatan rendah (neutron termal) yang akan menimbulkan reaksi fisi menjadi berkurang, sehingga akibatnya reaksi fisi (reaktivitas) juga berkurang. Jadi menaikkan daya reactor dengan cara menarik batang kendali akan selalu dikompensasi oleh produksi uap yang menekan daya. Proses kompensasi ini akan berakhir pada suatu kondisi stabil pada daya setimbang tertentu. Sebaliknya jika batang kendali disisipkan masuk ke dalam teras, reaksi fisi berkurang dengan hadirnya penyerap neutron (batang kendali) dalam teras. Produksi uap yang dihasilkan juga menurun karena produksi energi panas dari reaksi fisi berkurang. Akibatnya kemampuan air dalam memoderasi neutron bertambah, dan reaksi fisi akan mulai meningkat. Proses penurunan daya oleh batang kendali yang kemudian dikompensasi oleh penurunan daya karena membaiknya

kemampuan moderasi akan terus berlangsung hingga tercapai kondisi stabil pada suatu daya setimbang tertentu. Fenomena kompensasi oleh uap-air menjadi salah satu sarana penting dalam pengendalian-diri (*self control*) reaktor dan merupakan salah satu keunikan reaktor air didih.



Gambar 2.1 Batang Kendali FFCR [1]

Reaktor mempunyai 5 batang kendali[1]. Kelima batang kendali ini diselubungi oleh tabung baja tahan karat jenis 304 yang panjangnya kira-kira 109 cm dan berdiameter 3,43 cm.

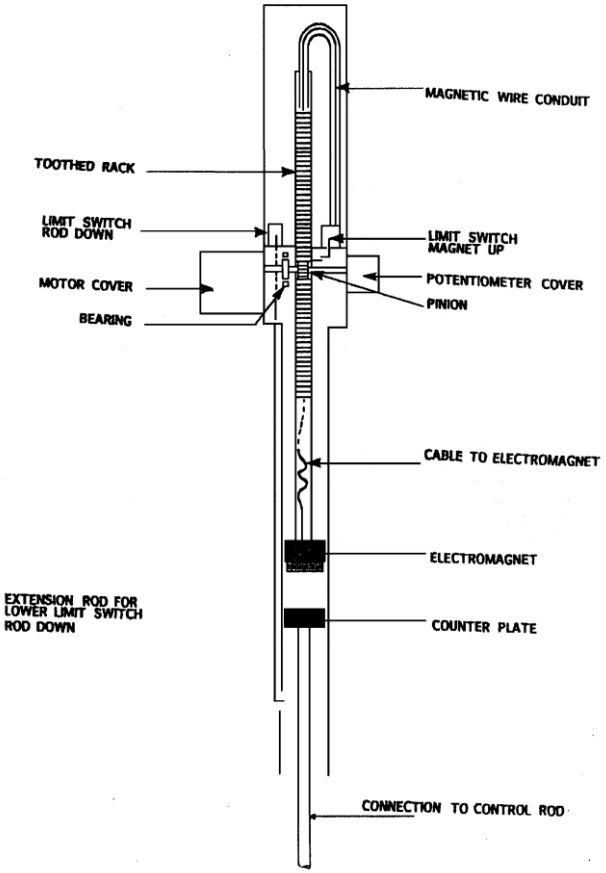
Bagian paling atas sepanjang 16,5 cm kosong berisi udara dan 38,1 cm berikutnya adalah penyerap neutron (*boron carbide* dalam bentuk padat). Selanjutnya di bawah penyerap neutron ini ada bagian *fuel follower* sepanjang 38,1 cm yang terbuat dari bahan bakar UZrH. Bagian bawah dari batang ini, yaitu sepanjang 16,5 cm, kosong berisi udara (lihat Gambar 2.1). Batang-batang kendali ini jatuh melalui dan dipandu oleh lubang berdiameter 3,81 cm pada kisi atas dan bawah lempeng. Kecepatan jatuh batang kendali ini cukup tinggi. Hanya dibutuhkan waktu 0,3-0,4 detik untuk mencapai dasar teras dari posisi tertinggi.

Perangkat penggerak batang kendali *(Control rod drive assemblies)* dipasang di atas jembatan di atas kolam reaktor. Setiap unit terdiri dari sebuah motor dengan *reduction gearing driving a rack* dan *pinion* seperti ditunjukkan dalam gambar 2.1. Suatu *helipot* yang disambung ke *pinion* menunjukkan indikasi posisi. Sebuah elektromagnet memegang batang kendali melalui lempeng pengikat (*counter plate*). Perputaran motor akan menggerakkan batang ke atas dengan kecepatan 0,3175 cm/detik. Dengan kecepatan ini, maka batang kendali dapat diangkat dari posisi terbawah sampai ke atas dalam waktu 120 detik.

Prinsip gagal selamat (*fail safe*) diterapkan dalam rancangan batang kendali ini. Jika aliran listrik terputus, maka elektromagnet akan melepaskan lempeng pengikat yang tadi dipegangnya. Akibat gaya gravitasi, batang kendali akan jatuh dengan cepat ke dalam teras dan memadamkan reaktor. Batang kendali pada dasarnya bekerja atas dasar gravitasi. Kenaikannya digerakkan oleh motor langkah (*stepping motor*) yang dihubungkan melalui gigi-gigi penggerak. Perputaran gigi-gigi ini yang dihubungkan dengan *helipot* menjamin keakuratan pembacaan posisi dari batang kendali. Karena perangkat penggerak batang kendali dipasang pada jembatan yang sangat masif di atas tangki, maka batang kendali dapat dijamin tidak akan bergeser.

Dalam keadaan *scram*, aliran listrik ke elektromagnet akan terputus. Lempeng pengikat yang tadi melekat pada

elektromagnet akan terlepas. Karena beratnya sendiri, maka batang kendali akan terhujam ke dalam teras. Selanjutnya suatu lubang dengan diameter 3,81 cm di lempeng kisi atas dan bawah menjamin bahwa batang kendali tidak melenceng dan keluar dari teras. Di bawah lempeng kisi bawah dipasang suatu lempeng penyelamat yang dibuat dari alumunium. Lempeng ini menjamin bahwa batang kendali tidak dapat lepas dari penggeraknya dan lolos ke bawah ke luar dari teras.



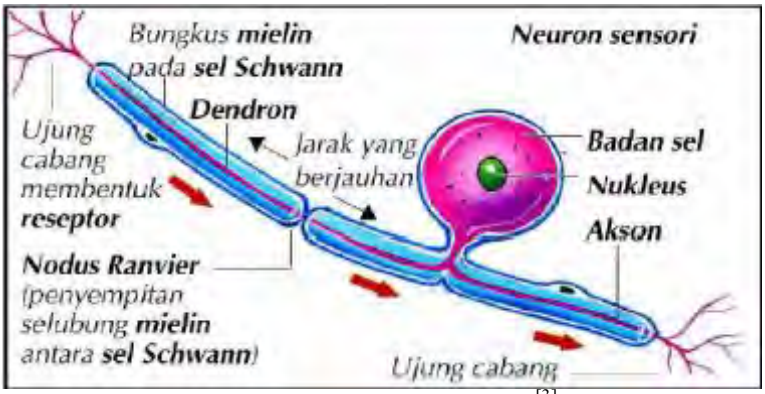
Gambar 2.2 Mekanisme Motor Batang Kendali [1]

2.2 Jaringan Syaraf Tiruan (JST)

Jaringan syaraf tiruan (JST) adalah salah satu algoritma pembelajaran mesin yang meniru cara kerja jaringan syaraf makhluk hidup. Jaringan saraf tiruan (*artificial neural network*) merupakan jaringan dari banyak unit pemroses kecil (yang disebut neuron) yang masing-masing melakukan proses sederhana, yang ketika digabungkan akan menghasilkan perilaku yang kompleks. Jaringan saraf tiruan dapat digunakan sebagai alat untuk memodelkan hubungan yang kompleks antara masukan (*input*) dan keluaran (*output*) pada sebuah sistem untuk menemukan pola-pola pada data[2]. Jaringan saraf tiruan dibuat berdasarkan model biologis otak manusia. Kemampuan komputer sudah melampaui otak manusia dalam hal komputasi numerik, tetapi otak manusia dapat mengerjakan persoalan lainnya secara lebih cepat dan akurat, misalnya pada persoalan pengenalan wajah, persoalan klasifikasi, dan persoalan penarikan keputusan. Oleh karena itu, dilakukanlah riset yang mencoba memodelkan proses yang terjadi di otak manusia. Riset-riset tersebut menghasilkan sebuah model matematis yang disebut jaringan saraf tiruan (*artificial neural network*) atau sering juga disebut *simulated neural network* atau hanya jaringan saraf (*neural network*).

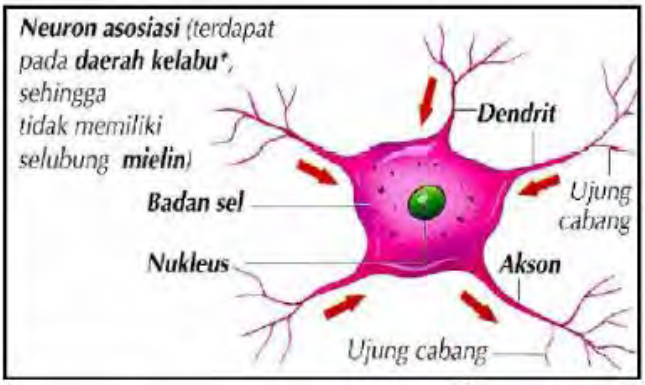
Otak manusia terdiri dari lebih dari 1011 sel saraf yang disebut neuron. Masingmasing sel saraf ini terdiri atas empat komponen dasar, yaitu dendrit, soma (badan sel) dan akson. Terdapat tiga macam neuron, yaitu neuron sensori, asosiasi dan motor. Masing-masing neuron terhubung dengan 200.000 hubungan melalui dendrit ke neuron lain pada otak. Dendrit menerima impuls dari neuron lain atau dari reseptor (bagian dari sistem pengindera yang mengirimkan impuls ketika dirangsang). Sebagian besar neuron memiliki beberapa dendrite pendek, kecuali neuron sensori yang hanya memiliki satu dendrite panjang. Dendrit panjang tersebut disebut dendron. Badan sel adalah bagian dari neuron yang di dalamnya terdapat nukelus (inti sel) dan sebagian besar sitoplasma sel. Masing-masing neuron

mempunyai akson, yaitu serabut saraf tunggal panjang yang membawa impuls dari badan sel. Akson akan menyampaikan impuls tersebut ke efektor (otot atau kelenjar) atau ke dendrit dari neuron lain. Daerah sempit tempat bertemunya akson dengan dendrit neuron lain disebut sinapsis. Pada celah sinapsis ini, impuls diteruskan ke dendrit sel lain dengan menggunakan zat kimia yang disebut neurotransmitter. Neuron sensori atau disebut juga neuron aferen merupakan neuron yang membawa impuls saraf. Ujung-ujung dendron (dendrit panjang) beberapa neuron sensori membentuk reseptor-reseptor di seluruh tubuh, yang mengirimkan impuls ke neuron ketika dirangsang. Kemudian reseptor akan terhubung dan mengantarkan impuls ke otak manusia melalui beberapa neuron sensori.



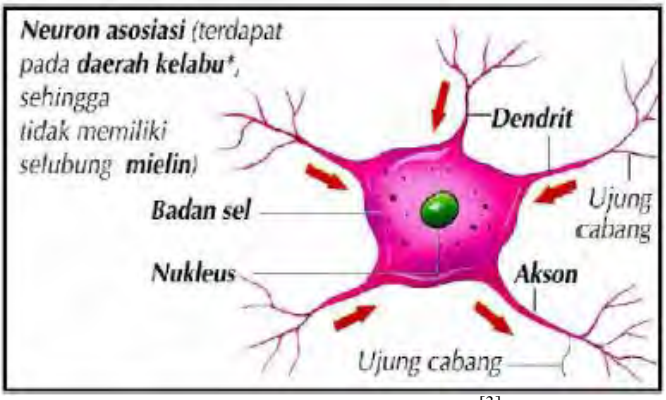
Gambar 2.3 Neuron Sensori[3]

Neuron asosiasi, yang disebut juga neuron penghubungatau interneuron merupakan neuron dengan jumlah paling banyak di otak dan sumsum tulang belakang. Neuron asosiasi ini terlibat dalam penerimaan input dari neuron sensori, penerjemahan *input* tersebut menjadi informasi serta penyampaian impuls ke neuron motor untuk melakukan gerakan/aksi.



Gambar 2.4 Neuron Asosiasi[3]

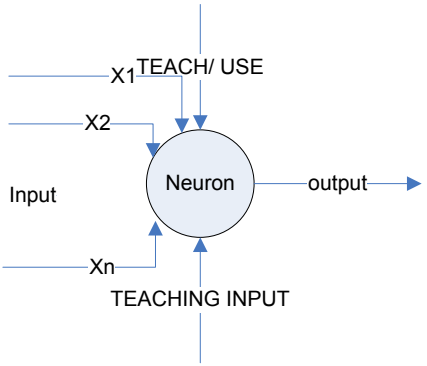
Neuron motor, disebut juga neuron aferen merupakan neuron yang membawa impuls saraf dari otak dan sumsum tulang belakang. Ujung akson neuron motor membentuk sambungan dengan otot atau kelenjar. Impuls yang dibawa oleh neuron motor dari neuron asosiasi akan merangsang organ-organ tubuh ini untuk bekerja.



Gambar 2.5 Neuron Motor[3]

Otak manusia bekerja dengan meneruskan impuls yang didapat dari neuron sensori ke neuron asosiasi dan kemudian menuju ke neuron motor. Hubungan antara neuron-neuron yang sangat banyak itu dapat menguatkan ataupun melemahkan, bahkan dapat terbentuk hubungan baru ataupun putus hubungan antara neruon-neuron tersebut, sesuai dengan aktivitas pada otak manusia. Pada awal pertumbuhan anak, perubahan yang terjadi pada keterhubungan neuron tersebut sangat banyak, seiring dengan pembelajaran yang dilakukan anak terhadap lingkungan sekitarnya. Saat manusia berkembang menjadi dewasa, hubungan antar neuron-neuron pada otaknya telah mampu membuatnya dapat berjalan, membaca, mendengar, mengambil keputusan, dan hal lain yang dapat dilakukan oleh manusia dewasa.

Jaringan Syaraf Tiruan memiliki pendekatan yang berbeda untuk memecahkan masalah bila dibandingkan dengan sebuah komputer konvensional. Umumnya computer konvensional menggunakan pendekatan algoritma (computer konvensional menjalankan sekumpulan perintah untuk memecahkan masalah). Jika suatu perintah tidak diketahui oleh komputer konvensional maka komputer konvensional tidak dapat memecahkan masalah yang ada. Sangat penting mengetahui bagaimana memecahkan suatu masalah pada computer konvensional dimana komputer konvensional akan sangat bermanfaat jika dapat melakukan sesuatu dimana pengguna belum mengatahui bagaimana melakukannya. Jaringan Syaraf Tiruan dan suatu algoritma komputer konvensional tidak saling bersaing namun saling melengkapi satu sama lain. Pada suatu kegiatan yang besar, sistem yang diperlukan biasanya menggunakan kombinasi antara keduanya (biasanya sebuah komputer konvensional digunakan untuk mengontrol Jaringan Syaraf Tiruan) untuk menghasilkan efisiensi yang maksimal. Jaringan Syaraf Tiruan tidak memberikan suatu keajiban tetapi jika digunakan secara tepat akan menghasilkan sesuatu hasil yang luar biasa.



Gambar 2.6 Sebuah JST Sederhana[4]

2.2.1. Karakteristik Jaringan Syaraf Tiruan

Dengan struktur dasar seperti yang dijelaskan pada bagian sebelumnya, algoritma jaringan saraf tiruan memiliki karakteristik-karakteristik sebagai berikut :

1. Masukan dapat berupa nilai diskrit atau real yang memiliki banyak dimensi
2. Keluaran berupa vektor yang terdiri dari beberapa nilai diskrit atau real
3. Dapat mempelajari permasalahan secara *black box*, dengan hanya mengetahui nilai masukan serta keluarannya saja.
4. Mampu menangani pembelajaran terhadap data yang memiliki derau (*noise*)
5. Bentuk dari fungsi target pembelajaran tidak diketahui, karena hanya berupa bobot-bobot nilai masukan pada setiap neuron.
6. Karena harus mengubah banyak nilai bobot pada proses pembelajaran, maka waktu pembelajaran menjadi lama, sehingga tidak cocok untuk masalah-masalah yang memerlukan waktu cepat dalam pembelajaran.
7. Jaringan saraf tiruan hasil pembelajaran tiruan dapat dijalankan dengan cepat.

Sebuah jaringan yang sederhana mempunyai struktur *feedforward* dimana sinyal bergerak dari *input* kemudian melewati lapisan tersembunyi dan akhirnya mencapai unit *output* (mempunyai struktur perilaku yang stabil). Tipe jaringan *feedforward* mempunyai sel syaraf yang tersusun dari beberapa lapisan. Lapisan *input* bukan merupakan sel syaraf. Lapisan ini hanya memberi pelayanan dengan mengenalkan suatu nilai dari suatu *variabel*. Lapisan tersembunyi dan lapisan output sel syaraf terhubung satu sama lain dengan lapisan sebelumnya. Kemungkinan yang timbul adalah adanya hubungan dengan beberapa unit dari lapisan sebelumnya atau terhubung semuanya (lebih baik). Yang termasuk dalam struktur *feedforward* :

- *Single-layer perceptron*
- *Multilayer perceptron*
- *Radial-basis function networks*
- *Higher-order networks*
- *Polynomial learning networks*

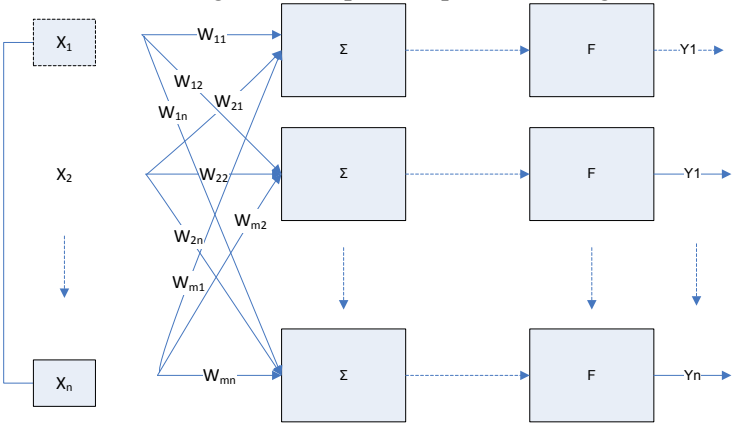
Jaringan Syaraf Tiruan biasanya mempunyai 3 *group* atau lapisan yaitu unit-unit :

1. lapisan *input* yang terhubung dengan lapisan tersembunyi yang selanjutnya terhubung dengan lapisan *output***.** Aktifitas unit-unit lapisan *input* menunjukkan informasi dasar yang kemudian digunakan dalam Jaringan Syaraf Tiruan.
2. Aktifitas setiap unit-unit lapisan tersembunyi ditentukan oleh aktifitas dari unit-unit *input* dan bobot dari koneksi antara unit-unit *input* dan unit-unit lapisan tersembunyi.
3. Karakteristik dari unit-unit *output* tergantung dari aktifitas unit-unit lapisan tersembunyi dan bobot antara unit-unit lapisan tersembunyi dan unit-unit *output*.

2.2.2. Perceptron

Perceptron termasuk kedalam salah satu bentuk Jaringan Syaraf Tiruan yang sederhana. Perceptron biasanya digunakan untuk mengklasifikasikan suatu tipe pola tertentu yang sering

dikenal dengan istilah pemisahan secara linear. Pada dasarnya perceptron pada Jaringan Syaraf dengan satu lapisan memiliki bobot yang bisa diatur dan suatu nilai ambang. Algoritma yang digunakan oleh aturan perceptron ini akan mengatur parameter-parameter bebasnya melalui proses pembelajaran. Fungsi aktivasi dibuat sedemikian rupa sehingga terjadi pembatasan antara daerah positif dan daerah negatif. Perceptron dapat dilihat di gambar 2.9.



Gambar 2.7 Bentuk Perceptron[5]

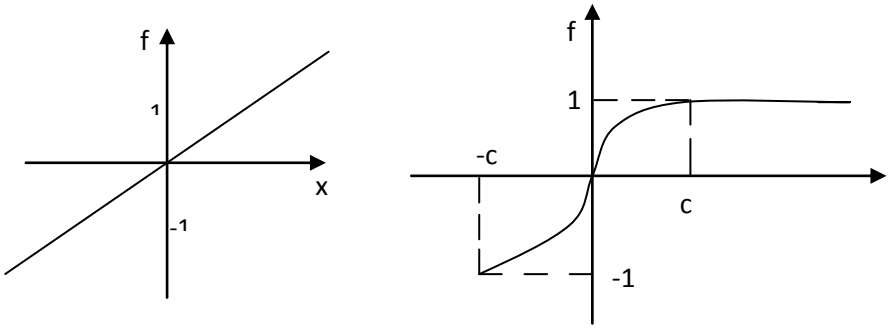
Fungsi aktivasi yang sering dipakai dalam JST antara lain:

1. Fungsi aktivasi linier
   Dengan f(x) = x, untuk semua harga x

2. Fungsi aktivasi tangent hyperbolic

$$f(x) = \frac{exp^{(-cx)} - exp^{(cx)}}{exp^{(-cx)} + exp^{(cx)}} \quad \text{..... (2.1)}$$

Kurva output untuk fungsi aktifasi *liniear* dan *tangent hyperbolic* dapat dilihat pada Gambar 2.8.



Gambar 2.8 Fungsi aktifasi linear dan fungsi aktifasi tangent hiperbolik[8]

3. Fungsi aktivasi sigmoid
4. Fungsi aktivasi step

Untuk membuat Jaringan Syaraf Tiruan untuk melakukan beberapa kerja khusus. Harus dipilih bagaimana unit-unit dihubungkan antara satu dengan yang lain dan harus mengatur bobot dari hubungan tersebut secara tepat. Hubungan tersebut menentukan apakah mungkin suatu unit mempengaruhi unit yang lain. Bobot menentukan kekuatan dari pengaruh tersebut. Dapat dilakukan pembelajaran terhadap 3 lapisan pada Jaringan Syaraf Tiruan untuk melakukan kerja khusus dengan menggunakan prosedur dibawah ini :

1. Memperkenalkan Jaringan Syaraf Tiruan dengan contoh pembelajaran yangterdiri dari sebuah pola dari aktifitas untuk unit-unit *input* bersama dengan pola yang diharapkan dari aktifitas untuk unit-unit *output*.
2. Menentukan seberapa dekat *output* sebenarnya dari Jaringan Syaraf Tiruan sesuai dengan *output* yang diharapkan.
3. Mengubah bobot setiap hubungan agar Jaringan Syaraf Tiruan menghasilkan suatu perkiraan yang lebih baik dari output yang diharapkan.

2.2.3. Algoritma Belajar (*Learning Algorithm*)

Suatu algoritma yang digunakan pada tahap pelatihan untuk mengatur nilai dari bobot (weight) jaringan syaraf tiruan. Adapun tipe algoritma belajar antara lain:

- Orde satu: melibatkan turunan pertama *(Gradient)* error terhadap bobot.
  Contoh: *Backpropagation algorithm*
- Orde dua: melibatkan turunan kedua *(Hessian)* error terhadap bobot.
  Contoh: *levenberg marquardt*

Algoritma *Levenberg Marquardt*

Algoritma pembelajaran yang digunakan dalam penelitian ini adalah algoritma *Levenberg Marquardt*. Algoritma *Levenberg Marquardt* memiliki kelebihan karakteristiknya lebih cepat konvergen *(rapid convergence),* namun demikian algoritma *Levenberg Marquardt* membutuhkan penurunan yang lebih rumit dibanding algoritma *Levenberg Marquardt*. Algoritma *Levenberg Marquardt* merupakan algoritma belajar orde 2 sehingga melibatkan persamaan gradient dan matrik Hessian. Namun demikian persamaan matrik Hessian yang digunakan adalah *Gauss Newton Hessian matrik*. Pada algoritma *Levenberg Marquardt* hubungan antara matrik Hessian dengan gradient dinyatakan sebagai berikut[7]:

$$\left|R\left(\theta^{(i)}\right) + \lambda^{(i)} I\right| f^{(i)} = -G(\theta^{(i)}) \quad \text{..... (2.2)}$$

dan

$$\theta^{(i+1)} = \theta^{(i)} + \mu^{(i)} f^{(i)} \quad \text{..... (2.3)}$$

Pada persamaan (2.1) terlihat bahwa diagonal utama matrik Hessian ditambahkan dengan elemen λ. Faktor λ akan mengontrol kovergensi pada algoritma *Levenberg Maquardt* dan besarnya nilai λ diatur berdasarkan rasio persamaan sebagai berikut:

$$r = \frac{V_N(\theta^{(i)}, Z^N) - V_N(\theta^{(i)} + f^{(i)}, Z^N)}{V_N(\theta^{(i)}, Z^N) - L^{(i)}(\theta^{(i)} + f^{(i)})} \quad \text{..... (2.4)}$$

dengan

$$L(\theta^{(i)} + f) = \sum_{t=1}^{N} y(t) - \hat{y}\left(t \middle| \theta^{(i)} - f^T \left[\frac{\delta \hat{y}(t|\theta)}{\delta \theta}\right]\right)^2$$
$$= V_N(\theta^{(i)}, Z^N) + f^T G(\theta^{(i)}) + \frac{1}{2} f^T R(\theta^{(i)}) f \quad \text{... (2.5)}$$

Dari persamaan (2.19) diperoleh hubungan :
$R(\theta^{(i)}) f^{(i)} = -G(\theta^{(i)}) - \lambda f^{(i)}$, substitusi ke persamaan (2.22) didapatkan:

$$V_N(\theta^{(i)}, Z^N) - L^{(i)}(\theta^{(i)} + f^{(i)}) = \frac{1}{2}\left[-\left(f^{(i)T} G\left(\theta^{(i)} + \lambda^{(i)} \middle| f^{(i)}\right|^2\right)\right] \quad \text{..... (2.6)}$$

Dari persamaan (2.23) terlihat bahwa nilai dari *predicted decrease* hanya merupakan fungsi dari gradient serta *search direction*. Pengaturan nilai λ dilakukan berdasarkan nilai ratio r dengan ketentuan sebagai berikut :

- Jika $r^{(i)} > 0.75 \rightarrow \lambda^{(i)} = {}^{\lambda^{(i)}}/_{2}$ (diperkecil λ)
- Jika $r^{(i)} < 0.25 \rightarrow \lambda^{(i)} = 2\lambda$ (diperbesar λ)

Secara umum algoritma *Levenberg Maquardt* dapat ditulis sebagai berikut :

1. Pilih vektor bobot awal $\theta^{(i)}$ dan harga awal $\lambda^{(i)}$ dimana θ adalah bobot dan λ diberikan harga awal.
2. Tentukan arah pencarian :
   $\left|R(\theta^{(i)}) + \lambda^{(i)} I\right| f^{(i)} = -G(\theta^{(i)})$ ..... (2.7)
   maka diperoleh *f* dan dimasukkan ke :
   $\theta = \arg\min V_N(\theta, Z^N)$ ..... (2.8)
   $\theta^{(i+1)} = \theta^{(i)} + \mu^{(i)} f^{(i)}$ ..... (2.9)

   Jika $V_N(\theta(i) + f(i), Z^N) < V_N(\theta(i), Z^N)$ sehingga memenuhi $\theta(i+1) = \theta(i) + f(i)$ sebagai iterasi baru

maka $\lambda(i+1) = \lambda(i)$. Jika tidak maka mencari λ baru dengan harga r.

$$r = \frac{V_N(\theta^{(i)}, Z^N) - V_N(\theta^{(i)} + f^{(i)}, Z^N)}{V_N(\theta^{(i)}, Z^N) - L^{(i)}(\theta^{(i)} + f^{(i)})} \quad \text{..... (2.10)}$$

Jika $r^{(i)} > 0.75 \rightarrow \lambda^{(i)} = \lambda^{(i)}/2$ (diperkecil λ)

Jika $r^{(i)} < 0.25 \rightarrow \lambda^{(i)} = 2\lambda$ (diperbesar λ)

Dimana :

$$V_N(\theta, Z^N) = L^{(i)}(\theta) - \frac{1}{2N}\sum[y(k) - \hat{y}(k|\theta)^r]\,[y(k) - \hat{y}(k|\theta)] \quad \text{.. (2.11)}$$

$$L(\theta^{(i)} + f^{(i)}) = V_N(\theta^{(i)}, Z^N) + f^T G(\theta^{(i)}) + \frac{1}{2} f^T R(\theta^{(i)}) f \quad \text{..... (2.12)}$$

3. Jika kriteria tercapai, maka perhitungan berhenti. Jika kriteria belum tercapai maka mengulangi langkah no. 2.

2.2.4. Identifikasi dengan Jaringan Syaraf Tiruan

Sistem identifikasi merupakan usaha untuk mendapatkan deskripsi matematik (model) suatu sistem dinamik berdasarkan data pengukuran dan pengamatan yang diperoleh dari sistem tersebut. Secara umum model suatu sistem dapat dikategorikan menjadi 2, yakni :

1. Fundamental model (first principle model): didasarkan pada kaidah-kaidah hukum fisika dan kimia (*mass energy balance*, hukum Newton, dll).
   Keuntungan : dapat diperkirakan ke ekstrapolasi pada daerah operasi yang tidak digunakan pada data latih.
   Kelemahan : model dinamik yang dihasilkan mungkin sangat kompleks.
2. Empirical model: didasarkan pada hubungan input-output sistem
   Keuntungan : detail proses yang terjadi tidak perlu dicari terlebih dahulu dan dapat digunakan untuk model yang sangat komplek.

Sesuai dengan karakteristik yang dimiliki jaringan syaraf tiruan, maka model yang dihasilkan oleh jaringan syaraf

tiruan merupakan empirical model serta non parametric model. Fokus utama dari sistem identifikasi dengan jaringan syaraf tiruan hanya untuk sistem non linier yang dinyatakan sebagai berikut :

$y(t) = f(x(t))$ ..... (2.13)

$x(t) = \left[u(t-1) \dots u(t-n_u) \middle| y(t-1) \dots y(t-n_y)^T\right]$ ... (2.14)

keterangan :

f = fungsi nonlinier
x(t) = *regressor*
y(t) = *output* sistem
u(t) = *input* sistem
n_u = *history length* untuk *input* sistem
n_y = *history length* untuk *output* sistem

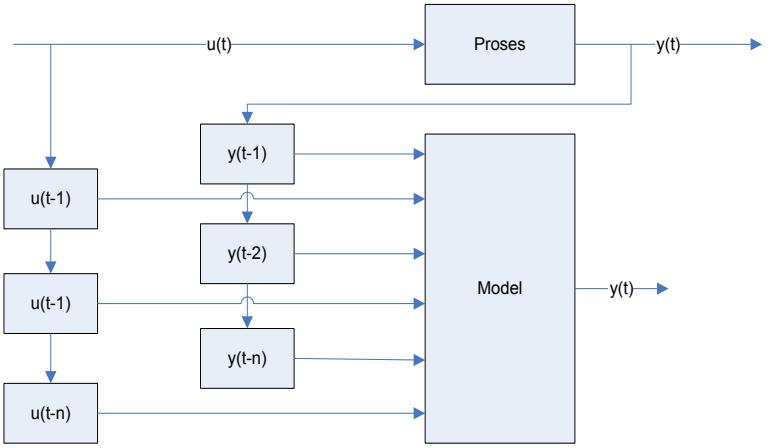
Pada prinsipnya, sistem identifikasi non linier dapat dibedakan menjadi 2, yaitu :

1. *Series-parallel/ nnarx (Neural Network Auto Regressive with eXternal input) model :*

   $\hat{y}(t) = \hat{f}(x(t))$ ..... (2.15)

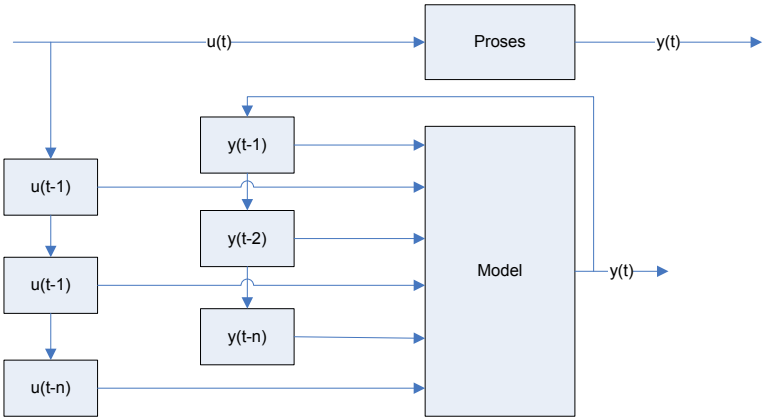
   $x(t) = \left[u(t-1) \dots u(t-n_u) y(t-1) \dots y(t-n_y)^T\right]$ (2.16)

   Gambar 2.9 adalah sistem identifikasi non linear dengan NNARX.



Gambar 2.9 NNARX model[7]

2. *Parallel/ NNOE (Neural Network Output Error) model.*
   Persamaannya :
   $\hat{y}(t) = \hat{f}(x(t))$ ..... (2.17)
   $x(t) = \left[u(t-1) \dots u(t-n_u) y(t-1) \dots y(t-n_y)^T\right]$ (2.18)
   Gambar 2.10 adalah sistem identifikasi non linear dengan NNOE.



Gambar 2.10 NNOE model[7]

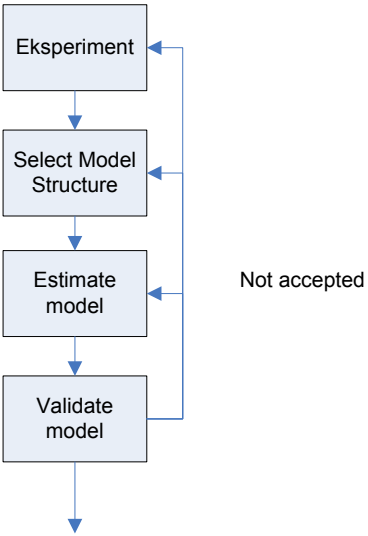
Prosedur dalam sistem identifikasi :

1. *Eksperiment*, meliputi input sequence design. Eksperimen dilakukan untuk mendapatkan serangkaian data input-output yang menerangkan perilaku proses pada suatu range daerah operasi tertentu. Ide utama dari proses eksperimen adalah untuk memasukkan input yang bervariasi, u dan mengamati akibatnya pada output, y. Pasangan data yang berhubungan dengan input dan output :
   $Z^n = \{[u(t), y(t)]T = 1, \dots, N\}$..... (2.19)

   kemudian digunakan untuk mendapatkan sebuah model dari sistem. Apabila sistem yang akan diidentifikasi menjadi tidak stabil atau mengandung sedikit peredaman dinamik, maka pembangkitan data dilakukan dalam keadaan lup tertutup.

Beberapa parameter penting dalam melakukan eksperimen antara lain: pemilihan sampling frekuensi, pemilihan sinyal input yang sesuai dan pemrosesan data.

2. *Select model structure,* meliputi structur selection, noise modeling. Pemilihan struktur model menyangkut jumlah sinyal input-output (regressor) yang digunakan sebagai masukan bagi model dalam menghasilkan output prediksi. Stuktur model adalah pasangan kandidat model. Masalah utama dalam pemilihan model struktur adalah :
   - Memilih sebuah ”keluarga” dari struktur model untuk mendiskripsikan sebuah sistem, contohnya: struktur model linier, jaringan *multilayer percepteron,* jaringan *radial basis function, wavelets* atau model *Hammerstein.*
   - Memilih sebuah *subset* dari keluarga yang telah ditentukan. Pada struktur sistem linier, dapat berupa sebuah struktur model ARX(3,2,1), dimana (3,2,1) adalah waktu tunda dari suatu periode sampling dan *output* saat ini tergantung dari dua *output* masa lampau dan tiga *input* masa lampau.
3. *Estimate parameter,* meliputi parameter estimation. Jika struktur model telah ditentukan, maka tahap berikutnya adalah melakukan estimasi terhadap parameter model agar mampu memberikan hasil yang baik berdasarkan criteria tertentu. Kriteria tersebut dapat dirumuskan dengan berbagai cara, tetapi harus secara ideal menghubungkan penggunaan model yang diharapkan. Strategi yang paling umum adalah dengan mengambil yang menyediakan *onestep a head prediction* paling bagus dengan *squared error* terkecil antara *output* sistem dengan *output* prediksi.
4. *Model validation,* diperlukan untuk mengetahui apakah model yang telah diperoleh mampu memenuhi kebutuhan yang diperlukan



Gambar 2.11 Prosedur Identifikasi[3]

2.2.5. Kriteria Performansi Pemodelan JST

Kriteria yang digunakan untuk menilai pemodelan JST menggunakan *Root Mean Square Error* (RMSE) dan *Variance Accounted For* (VAF).

- RMSE adalah akar rata-rata total kuadrat *error* yang terjadi antara keluaran model dan keluaran proses. Semakin kecil nilai RMSE (mendekati nilai nol) maka makin besar tingkat keberhasilan *training*, sebaliknya semakin besar nilai RMSE maka makin kecil tingkat keberhasilan *training*. Persamaan nilai RMSE dapat dituliskan sebagai berikut:

$$RSME = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}} \quad \text{..... (2.20)}$$

dengan : y_i = *output* proses
$\hat{y}_i$ = *output* model
N = jumlah data

- VAF
  juga dinyatakan dalam VAF (*Variance Accounted For*) dalam persen. Dengan ketentuan bahwa semakin besar nilai VAF (mendekati nilai 100) maka makin besar tingkat keberhasilan *training*. Persamaan VAF dapat dituliskan sebagai berikut:

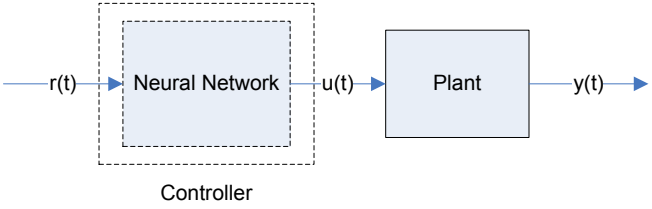
$$VAF = \left\{1 - \frac{var[y(t) - \hat{y}(t)]}{var[y(t)]}\right\} x100\% \quad \text{..... (2.21)}$$

2.3 Sistem Kontrol dengan Jaringan Syaraf Tiruan

Aplikasi dari jaringan syaraf tiruan adalah untuk suatu sistem kontrol. Karakteristik Jaringan syaraf tiruan sebagai sistem non linier sangatlah sesuai untuk suatu sistem kontrol non linier. Jaringan syaraf tiruan untuk sistem kontrol non linier dibedakan menjadi :

Direct Control System Design

Design sistem kontrol langsung dimana jaringan syaraf tiruan digunakan secara langsung untuk kontrol non - linier. Hal ini berarti jaringan syaraf tiruan akan membangkitkan sinyal kontrol yang diaplikasikan pada plant.



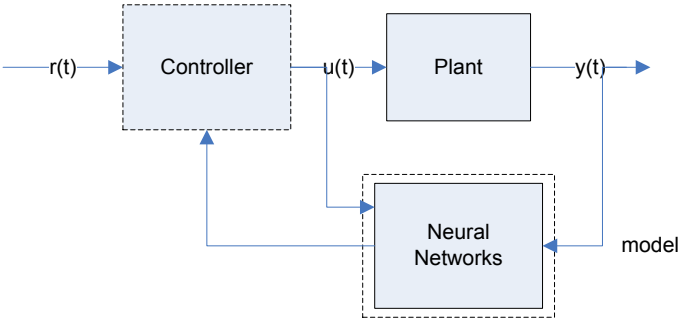
Gambar 2.12 Direct Control System[7]

Contoh dari *direct control system design* antara lain:
- Direct inverse control
- Optimal control

Indirect Control System Design

Design sistem kontrol tak langsung dimana jaringan syaraf tiruan akan digunakan sebagai model proses non linier dari pada

pengendali non linier. Perancangan tersebut meliputi model *based control system*. Hal ini berarti sebuah model digunakan secara eksplisit pada perhitungan untuk mengarahkan sinyal kontrol yang akan diaplikasikan pada plant.



Gambar 2.13 Indirect Control System[7]

Direct Inverse Control

Direct inverse control merupakan salah satu sistem pengendalian berbasis JST dengan kontrol yang dihubungkan secara seri antara model pengendali JST dengan model *plant* JST.

Dari hasil *training* pemodelan *plant* didapatkan nilai bobot, jumlah *history length* dan *hidden node* yang terintegrasi dalam *forward* sebagai model *plant* yang akan digunakan dalam simulasi DIC. Begitu pula hasil training pemodelan pengendali didapatkan nilai bobot, jumlah *history length* dan *hidden node* yang terintegrasi dalam *inverse* sebagai model pengendali yang akan digunakan dalam simulasi DIC.

Untuk mengetahui tingkat keberhasilan hasil simulasi *direct inverse control* berbasis JST ini dilakukan uji *tracking setpoint* dengan parameter nilai *maximum overshoot* (Mp) dan nilai *settling time* (ts) dari respon pengendali yang di hasilkan.

Direct inverse control, seperti yang ditunjukkan pada gambar 2.14. Prinsip dari proses ini dapat dideskripsikan sebagai berikut :

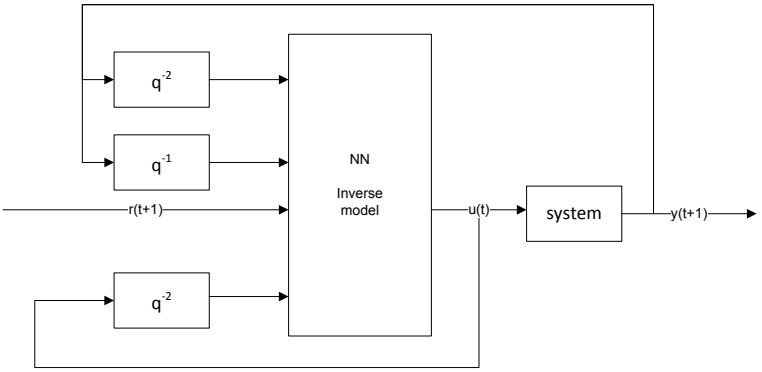
$$y(t+1) = g[y(t) \dots, y(t-n+1), u(t), \dots, u(t-m)] \dots \quad (2.22)$$

Jaringan yang digunakan untuk melatih proses inverse adalah :

$$\hat{u}(k) = \hat{g}^{-1}[y(t+1), y(t), \dots, y(t-n+1, u(t), \dots u(t-m)] \quad (2.23)$$

Kemudian model inverse diaplikasikan sebagai kontrol untuk suatu proses dengan memasukkan output yang diinginkan. Sebelum sistem kontrol aktual bekerja maka model inverse harus dilatih. Metode pembelajaran untuk kontrol berbasis jaringan syaraf tiruan dapat dibedakan menjadi 2 metode, yaitu :

1. Generalized Training
   Pada metode ini jaringan syaraf tiruan ditraining secara offline untuk meminimalisasi mean square error (MSE) di sinyal kontrol yang akan diterapkan pada proses melalui percobaan sinyal kontrol yang dihasilkan dari jaringan.
2. Specialized Training
   Metode ini berfungsi untuk meminimalisasi nilai *mean square error* (MSE) antara sinyal referensi dan output dari proses. Metode ini telah berjalan dengan baik secara online dengan menggunakan *recursive training* algoritma.



Gambar 2.14 Direct Inverse Control[7]

Kelebihan dan kelemahan dari *direct inverse control* adalah[7] :

Kelebihan:

- Intuisi sederhana
- Sederhana penerapannya
- Dengan pelatihan specialized kontoller dapat di optimalkan untuk *specific reference trajectory*
- Pada prinsipnya penerapan pelatihan *specialized* tepat pada bermacam-macam sistem

Kelemahan:

- Tidak bekerja untuk sistem dengan inverse yang tidak stabil, yang mana sering terjadi ketika menggunakan frekuensi sampling tinggi.
- Kurang pilihan tuningnya
- Biasanya diharapkan menunjukkan kepekaan yang tinggi untuk gangguan dan kebisingan.

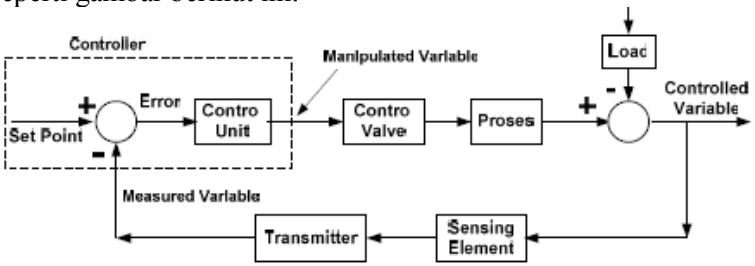
2.4 Strategi Pengendalian

Untuk sistem pengendalian yang umum digunakan pada industri menggunakan umpan balik atau *feed back.* Berikut pula akan ditunjukkan performansi dari pengendalian pada tanggapan transient pada subbab dibawah ini.

2.4.1. Sistem Pengendalian Umpan Balik (Feed Back*)*

Pengendalian *feedback* merupakan jenis pengendalian yang sangat umum, dimana aplikasinya banyak sekali digunakan di berbagai bidang, baik itu telekomunikasi, transportasi, kesehatan, industri dan lain-lain. Khusus di bidang industri, pengendalian *feedback* adalah kunci dan merupakan dasar dari pengendalian yang ada pada *plant*. Ada beberapa buah komponen utama dalam pengendalian *feedback*, yakni *Sensing Element* yang berfungsi untuk melakukan pengukuran besarnya *process variable. Transmitter* digunakan untuk mentransmisikan nilai *measured variable* ke pengendali. Pengendali berfungsi untuk mengukur terjadinya *error* dan memberikan nilai kompensasi yang tepat untuk mengembalikan process variable sesuai nilai

setpoint. Dan *final control element* merupakan komponen terakhir yang bertindak sebagai eksekutor dari *manipulated variable* yang dikirimkan dari pengendali, biasanya untuk proses di industri *final control element* berupa *control valve*. Dalam bentuk diagram blok, sistem pengendalian dapat digambarkan dalam bentuk seperti gambar berikut ini.



Gambar 2.15 Diagram Blok Pengendalian Umpan Balik[11]

Didapatkan hubungan *error* sebagai berikut;

$e = SP - MV$ ..... (2.24)

keterangan :

e = nilai *error*

SP = nilai *setpoint*

MV = nilai variabel yang terukur *(measured variable)*

Nilai *error* yang terukur di pengendali akan dihitung untuk mendapatkan nilai *manipulated variable* (MV) yang akan diberikan ke *control valve*. Besarnya MV hasil perhitungan tergantung dari jenis pengendali yang digunakan, karena setiap jenis pengendali memiliki prinsip perhitungan yang berbeda-beda.

Keberadaan pengendali dalam sebuah sistem pengendalian mempunyai kontribusi yang sangat besar terhadap perilaku sistem. Pada prinsipnya hal itu disebabkan oleh tidak dapat diubahnya komponen penyusun sistem tersebut artinya karakteristik *plant* harus diterima sebagaimana adanya sehingga perubahan perilaku sistem hanya dapat dilakukan melalui perubahan subsistem yaitu pengendali.

2.4.2. Penggolongan Tanggapan Transien

Tanggapan transien suatu sistem pengendali secara praktek selalu menunjukkan osilasi teredam sebelum mencapai keadaan tunaknya. Dalam menggolongkan karakteristik tanggapan transien suatu sistem pengendali terhadap masukan tangga satuan, umum dikelompokkan sebagai berikut:

- Waktu tunda (td)
  Adalah waktu yang digunakan oleh tanggapan untuk mencapai setengah nilai akhir untuk waktu yang pertama.

- Waktu naik (tr)
  Adalah waktu yang diperlukan oleh tanggapan untuk naik dari 10% menjadi 90%, 5% menjadi 95%, atau 0% menjadi 100% dari nilai akhir yang biasa digunakan. Untuk sistem atas redaman waktu naik yang biasa digunakan 10% menjadi 90%.

- Waktu puncak (tp)
  Adalah waktu yang diperlukan tanggapan untuk mencapai puncak pertama *maximum overshoot*.

- *Maximum overshoot (Mp)*
  Adalah nilai puncak kurva tanggapan diukur dari satuan. Apabila nilai akhir keadaan tunak tanggapannya jauh dari satu, maka biasa digunakan persen maksimum *overshoot* dan didefinisikan sebagai berikut :

$$\%\, maksimumovershoot = \frac{c(tp)-c(\infty)}{c(\infty)} \text{x} 100\% \quad \text{..... (2.25)}$$

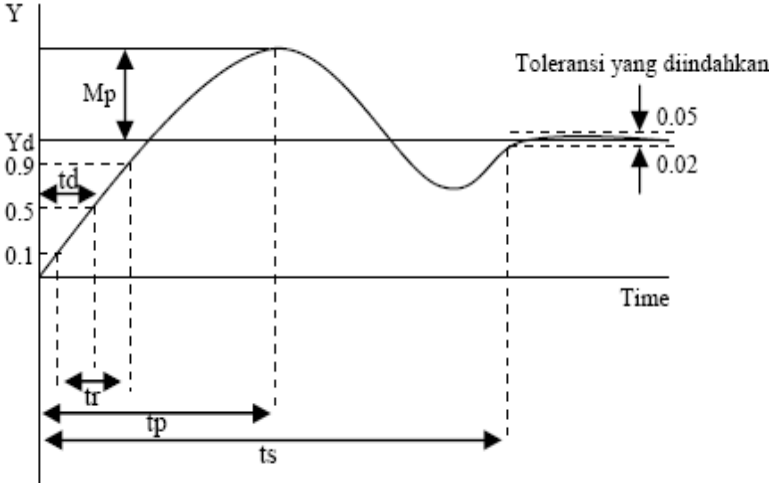
dimana : c(tp) = Keadaan pada waktu puncak
c(∞) = Keadaan pada setpoint

Besarnya persen *maximum overshoot* menunjukkan kestabilan *relative* dari sistem

- Waktu turun (ts)
  Adalah waktu yang diperlukan untuk menanggapi kurva agar dapat mencapai dan tetap berada dalam gugus akhir ukuran yang disederhanakan dengan presentase mutlak harga akhirnya

(biasanya 2% atau 5%). Waktu turun tadi dihubungkan dengan tetapan waktu terbesar sistem pengendali.

Secara grafik ditunjukan pada gambar berikut ini :



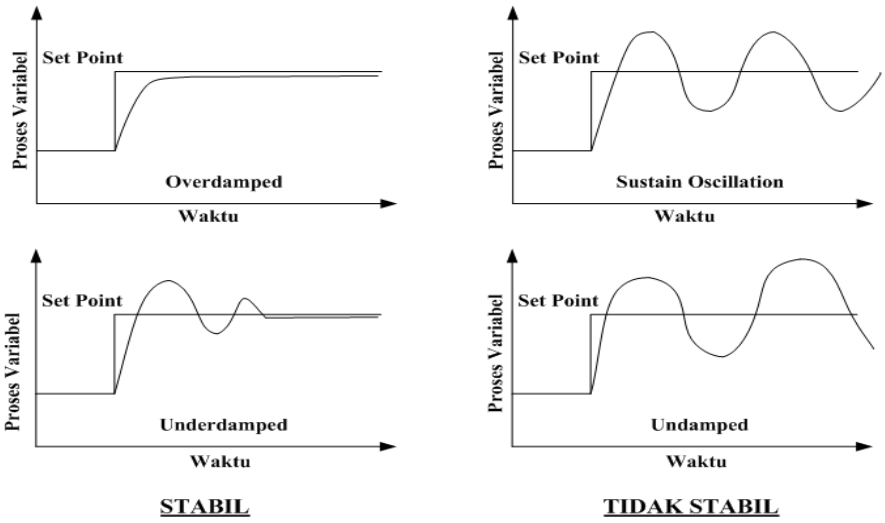
Gambar 2.16 Grafik Tanggapan Transient[12]

Spesifikasi daerah waktu menjadi penting karena kebanyakan sistem pengendali merupakan sistem daerah waktu yaitu dapat menerima tanggapan waktu. Ini berarti bahwa sistem pengendali harus dimodifikasi sampai tanggapan transien dipenuhi. Perhatikan bahwa jika kita cirikan nilai td, tr, tp, ts dan Mp, maka bentuk kurva tanggapan dapat ditentukan.

Perlu diperlihatkan bahwa semua spesifikasi di atas tidak dapat diterapkan untuk semua kasus. Sebagai contoh, untuk sistem atas redaman maka waktu puncak dam *maximum overshoot* tak dapat diterapkan. Untuk sistem yang memberikan kesalahan keadaan tunak untuk masukan langkah, kesalahan ini harus dibuat pada tingkat presentase tertentu.

Kondisi dari pada saat awal proses merespon nilai input *setpoint* sampai menjelang mencapai *settling time* sistem dikatakan pada kondisi transisi *(transient respon),* sedangkan

pada saat sistem sudah mencapai kriteria *error steady state* 2–5% sistem dikatakan pada kondisi *steady*. Respon sistem sendiri dapat dibagai menjadi *stable* dan *unstable*. Kondisi *stable* dibagi lagi menjadi 2 diantaranya *underdamped dan overdamped*. Respon *underdamped* ditandai dengan adanya *overshoot* pada kondisi *transient* sedangkan untuk respon *overdamped* ditandai dengan tidak adanya *overshoot* dan settling time yang lebih besar dibandingkan *underdamped*.



Gambar 2.17 Grafik Respon Sistem Pengendalian[12]

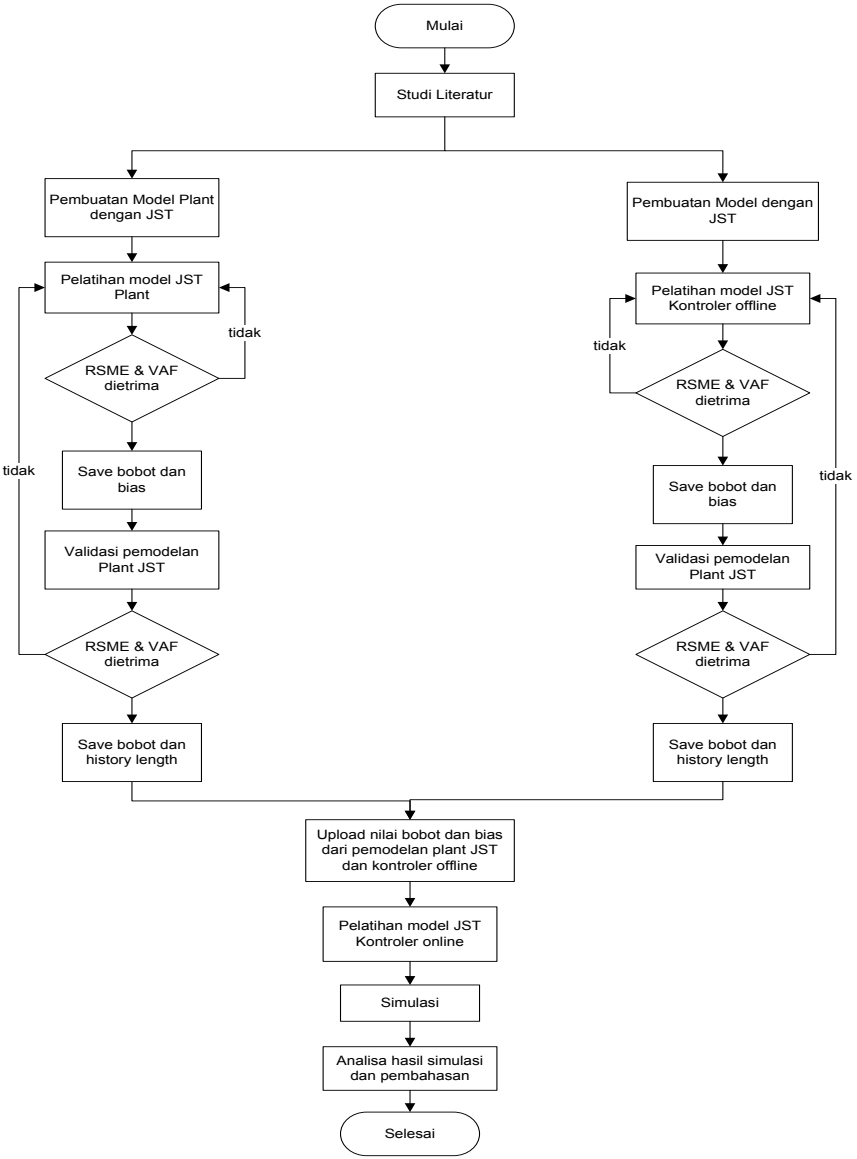
Sedangkan untuk respon sistem *unstable* dibagi menjadi *sustain oscillation* dan *undamped* (tidak teredam) pada kondisi *sustain oscillation*, sistem akan berosilasi dengan amplitudo yang konstan dan *process variable* yang tidak akan pernah mendekati nilai *setpoint*, nilai PV akan naik turun disekitar *setpoint* sedangkan untuk respon sistem *undamped*, sistem akan berosilasi dengan amplitudo yang semakin membesar.

BAB III
PERANCANGAN DAN METODOLOGI

Untuk menyelesaikan Tugas Akhir (TA) ini diperlukan beberapa tahapan dalam pengerjaannya, antara lain studi literatur, studi lapangan yang meliputi pengambilan data di lapangan, pemodelan plant dan pengendali menggunakan JST *Levenberg Marquardt*, melakukan validasi dari hasil pemodelan. Untuk lebuh jelasnya dapat dilihat pada flowchart di bawah ini.

Pada bab ini akan dibahas tahapan-tahapan perancangan sistem pengendalian reaktor nuklir. Sistem yang akan dirancang dibagi menjadi 2 (dua) bagian utama, yaitu *plant* dan sistem kontrol. Elemen penting dalam sistem pengendalian reaktor nuklir ini adalah batang kendali. Posisi batang kendali akan dikendalikan untuk mendapatkan besarnya keluaran reaktivitas yang berhubungan langsung dengan besarnya daya yang dihasilkan reaktor. Selain itu, batang kendali juga mengendalikan temperatur sistem pendingin primer di teras reaktor.

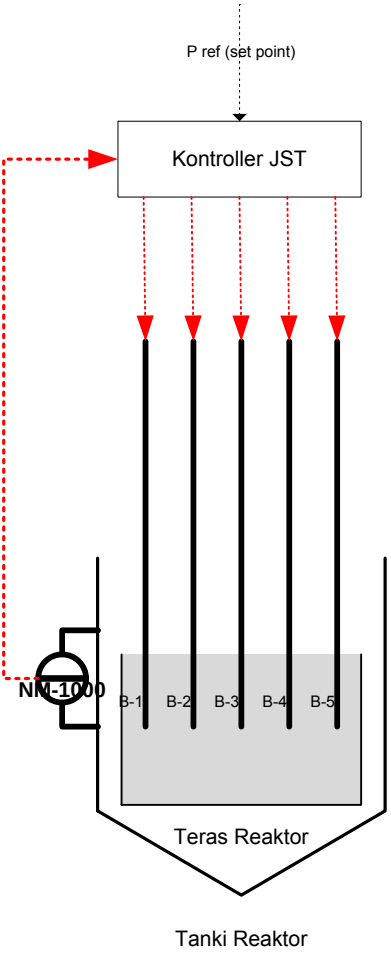
Di dalam merancang sebuah sistem pengendali menggunakan jaringan syaraf tiruan alogoritma *Levenberg Marquardt*, pada dasarnya terdiri dari 3 tahap utama sesuai dengan konsep dari jaringan syaraf tiruan (tahap pembelajaran dari tahap pemakaian) yaitu pemodelan plant, pelatihan *offline* pengendali (tahap pembelajaran) dan pelatihan *online* pengendali. Setiap tahapan ini harus dilakukan simulasi untuk melihat respon dari struktur plant jaringan syaraf tiruan yang berupa bobot dan bias tertentu dari plant sesungguhnya. Selanjutnya pada pelatihan *offline controller* dilakukan dengan tujuan mendapatkan bobot dan bias awal dari pengendali untuk digunakan pada pelatihan *online*. Dan untuk melihat kemampuan dari mengatasi perubahan *setpoint* dan gangguan harus dilakukan pelatihan *online* dengan menggabungkan pengendali jaringan syaraf tiruan dan model plant JST yang didapatkan dari tahap sebelumnya.



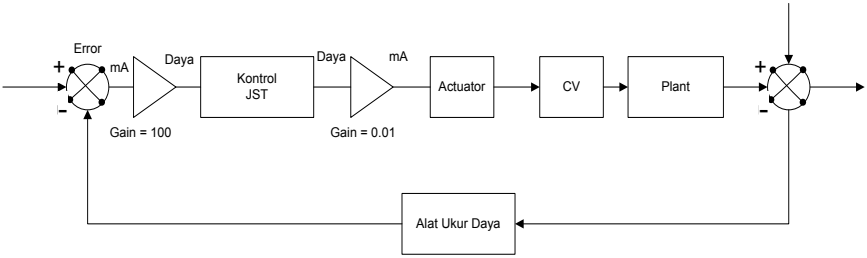
Gambar 3.1 Metodologi Penelitian Tugas Akhir

3.1. Skema Perancangan Sistem Pengendalian Daya Berbasis JST

Perancangan ulang sistem pengendalian daya reaktor nuklir berikut diagram blok ditunjukkan oleh gambar dibawah ini:



Gambar 3.2 Skema Perancangan Sistem Pengendalian Daya Reaktor Nuklir Berbasis JST



Gambar 3.3 Diagram Blok Perancangan Sistem Pengendalian Daya Reaktor Nuklir Berbasis JST

Pada sistem pengendalian daya reaktor, besarnya daya diukur oleh alat ukur daya (NP-1000), digunakan untuk mengetahui besarnya daya yang dihasilkan dari proses tersebut. Sistem pengendalian daya pada *real plant* masih secara manual. Maka pada penelitian ini akan dirancang sistem pengendalian secara otomatis mengunakan jaringan syaraf tiruan. Diagram blok perancangan, diagram tersebut menerangkan alur sistem output dari daya dibandingkan dengan daya referensi (*set point*) menghasilkan sinyal kesalahan (*error*, e).

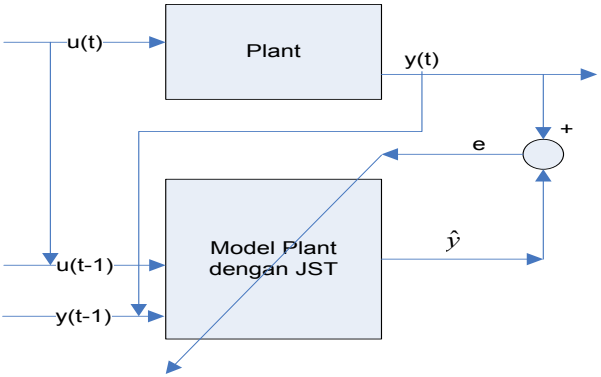
Sinyal kesalahan diolah oleh pengendali berbasis JST menghasilkan keluaran yang sesuai, selanjutnya aksi kendali diterima batang kendali untuk bergerak naik atau turun untuk menghasilkan daya yang diinginkan.

Pada sistem pengendalian ini variabel yang dikendalikan adalah daya reaktor nuklir, sedangkan variabel yang dimanipulasi adalah posisi batang kendali.

3.2. Pemodelan Plant Jaringan Syaraf Tiruan

Berdasarkan data masukan dan keluaran yang telah ditentukan dengan menggunakan metode pembelajaran JST akan dicari model *plant* yang dapat mempresentasikan dinamika proses pengendalian daya reaktor. Pada identifikasi proses dengan menggunakan JST, data masukan yang digunakan adalah posisi

kelima batang kendali. Sedangkan data keluaran yang digunakan adalah daya reaktor dalam teras reaktor. Berikut ini adalah diagram blok pemodelan proses menggunakan JST.



Gambar 3.4 Blok Diagram Pemodelan Plant JST[7]

Pemodelan *plant* menggunakan JST ini menggunakan 5 masukan, 1 referensi masukan dari keluaran *plant* untuk dilakukan pembelajaran guna mendapatkan keluaran dari pemodelan *plant* menggunakan JST yaitu $\hat{y}$ dimana merupakan keluaran proses pemodelan yang diinginkan. Masukan pada pemodelan *plant* ini menggunakan variabel masukan yang masuk ke dalam *plant* reaktor, yaitu variabel u1(t) merupakan masukan posisi batang kendali 1, variabel u2(t) merupakan masukan posisi batang kendali 2, variabel u3(t) merupakan masukan posisi batang kendali 3, variabel u4(t) merupakan masukan posisi batang kendali 4 dan variabel u5(t) merupakan masukan posisi batang kendali 5. Data masukan dan keluaran proses *plant* pada saat masa lampau (u(t-1), u(t-2),…,u(t-n)) menjadi data-data masukan model JST hingga menghasilkan prediksi keluaran $\hat{y}$. Data-data pada masa lampau dipakai untuk melatih jaringan hingga diperoleh bobot yang diinginkan. Pada saat pembelajaran data-

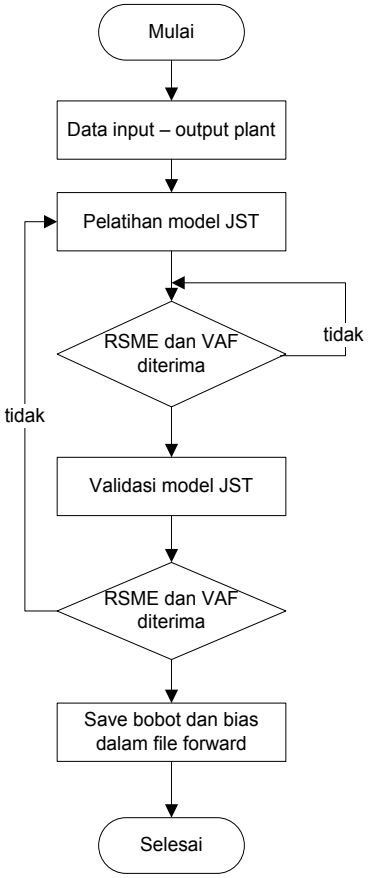
data ini diberikan ke jaringan kemudian jaringan akan memproses dan mengeluarkan keluaran.

Kemudian data keluaran $\hat{y}$ tersebut dibandingkan dengan data keluaran *plant* (y(t)) sebenarnya untuk diketahui berapa besar *error(e)* yang terjadi. Pada proses pembelajaran ini dicari *error* yang terkecil secara terus menerus sampai didapatkan nilai error yang kecil, hal ini biasa dikenal dengan istilah proses iterasi untuk mendapatkan bobot yang sesuai sehingga didapatkan nilai keluaran model *plant* JST yang sesuai dengan kriteria parameter yang telah ditentukan.

Dari 326 data *history real plant*, akan dipilih data-data yang akan digunakan untuk proses pembelajaran atau *training* oleh JST dan untuk validasi pemodelan *plant*. Dengan menggunakan arsitektur *Multi Layer Perceptron (MLP)* dimana pada arsitektur ini terdapat 3 layer yaitu *input layer*, *hidden layer*, dan *output layer*. Pada tiap-tiap *layer* terdapat fungsi aktifasi yang terdapat pada arsitektur jaringan tersebut. Fungsi aktifasi pada *hidden layer* menggunakan *tangent hyperbolic* sedangkan pada *output layer* menggunakan fungsi aktifasi *linear*. Untuk mendapatkan model *plant* JST yang bagus, sehingga mampu memprediksi keluaran proses dengan baik, struktur jaringan syaraf tiruan diuji coba dengan mengganti-ganti struktur jaringan. Diantaranya adalah jumlah *hidden node* dan jumlah *history length*.

Algoritma pembelajaran yang digunakan dalam penelitian ini adalah algoritma *Levenberg Marquardt*. Algoritma *Levenberg Marquardt* memiliki kelebihan karakteristiknya lebih cepat konvergen *(rapid convergence),* namun demikian algoritma Algoritma *Levenberg Marquardt* membutuhkan penurunan yang lebih rumit dibanding algoritma *backpropagation*.

Gambar 3.5 menunjukkan alur pemodelan *plant* dengan JST.



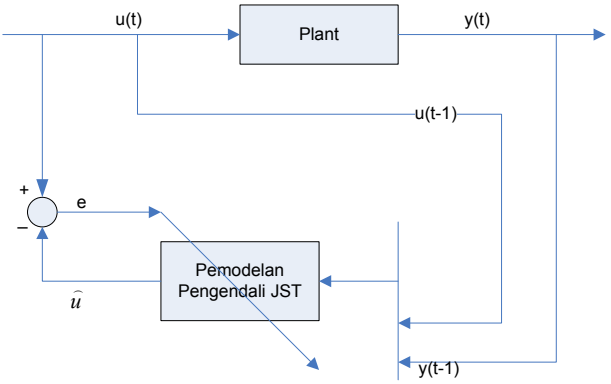
Gambar 3.5 Diagram alir pemodelan plant

Tujuan dari proses *training* adalah untuk mendapatkan bobot yang menghasilkan keluaran paling baik. Kriteria yang digunakan untuk menilai keluaran adalah *Root Mean Square Error* (RMSE) dan *Variance Accounted For* (VAF). Apabila keluaran model telah memenuhi nilai RMSE dan VAF yang paling baik maka bobot yang dihasilkan dari proses *training* yakni w1f (bobot dari *input layer* ke *hidden layer*) dan w2f (bobot dari

hidden layer ke *output layer*), *hidden node* serta *history length* disimpan sebagai *forward* untuk digunakan pada tahap *validasi* dan simulasi *Direct Inverse Control*.

3.3. Pemodelan Pengendali Jaringan Syaraf Tiruan

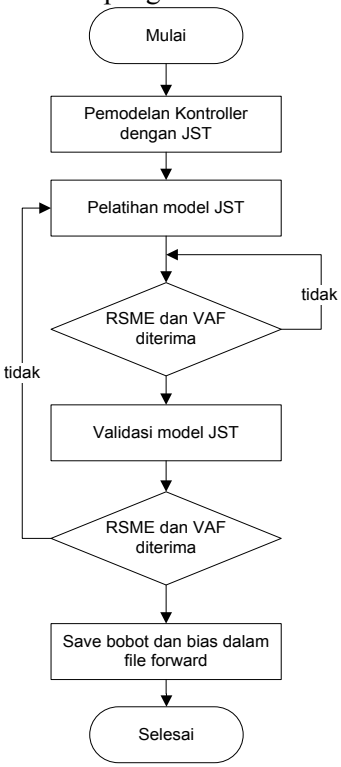
Proses pemodelan pengendali menggunakan JST memiliki konsep yang sama dengan pemodelan *plant* menggunakan JST, masih menggunakan hubungan masukan dan keluaran, hanya saja penggunaan data masukan dan keluarannya sekarang dibalik. Untuk model pengendali data masukan yang digunakan adalah daya reaktor, sedangkan data keluaran yang digunakan adalah posisi batang kendali.



Gambar 3.6 Diagram alir pemodelan pengendali[7]

Sama seperti pemodelan *plant*, terlebih dahulu dilakukan normalisasi data yaitu dengan melakukan penskalaan data atau normalisasi data menjadi berada dalam *range* 0-1. Selanjutnya data tersebut dilatih dan divalidasi dengan JST sehingga menghasilkan keluaran pemodelan yang berupa $\hat{u}$. Kemudian data keluaran model pengendali $\hat{u}$ tersebut dibandingkan dengan data masukan *plant* (u(t)) sebenarnya untuk diketahui berapa besar *error(e)*. Dari nilai *error* tersebut yang akan dipakai untuk member informasi untuk meng-*update* nilai

bobot JST secara terus menerus sampai didapatkan nilai error yang kecil pada pemodelan pengendali.

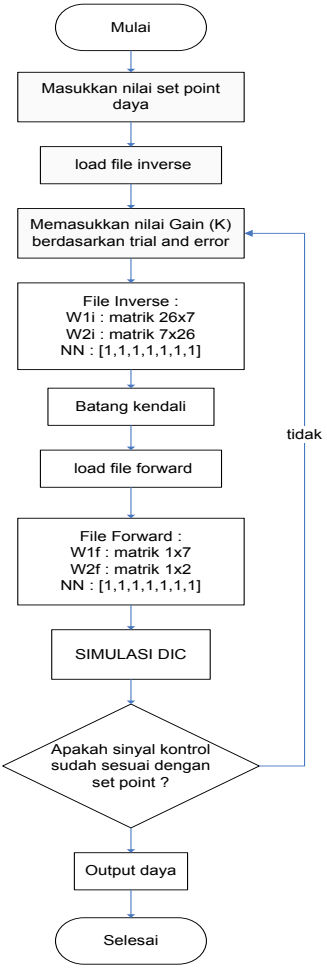


Gambar 3.7 Diagram alir pemodelan pengendali

Dari 326 *data history real plant*, akan dipilih data-data yang akan digunakan untuk proses pembelajaran atau training dan validasi pemodelan pengendali dengan JST. Apabila keluaran model telah memenuhi nilai RMSE dan VAF yang paling baik maka bobot yang dihasilkan dari proses *training* yakni w1i (bobot dari *input layer* ke *hidden layer*) dan w2i (bobot dari *hidden layer* ke *output layer*), *hidden node* serta *history length* disimpan sebagai *inverse* untuk digunakan pada tahap *validasi* dan simulasi *Direct Inverse Control*.

3.4. Simulasi Direct Inverse Control

Direct inverse control adalah kontrol yang dihubungkan secara seri antara model kontroller JST dengan model *plant* JST. Dari nilai bobot, jumlah *history length* dan *hidden node* hasil *training* pemodelan *plant* JST dan *training* pemodelan kontroller JST digunakan untuk simulasi *direct inverse control.*



Gambar 3.9 Alur Simulasi Direct Inverse Control Berbasis JST

Gambar 3.9 menunjukkan alur simulasi *direct inverse control* berbasis JST. Matrik dari bobot pemodelan kontroler yang tersimpan dalam W1f merupakan matrik 26x7, sedangkan matrik pemodelan plant tersimpan dalam W2i yang merupakan matrik 1x7. Besarnya nilai bobot dapat dilihat pada lampiran B. Pada saat awal memasukkan data input-output, nilai bobot dipilih secara acak hingga pada saat proses training dilakukan beberapa kali *epoch* (iterasi) sehingga diperoleh nilai bobot yang tepat agar nilai output sesuai dengan nilai target. Bobot disini dipergunakan untuk menghubungkan input dengan *hidden layer* dan *hidden layer* dengan output.

Untuk mengetahui tingkat keberhasilan hasil simulasi *direct inverse control* berbasis JST ini dilakukan uji tracking setpoint dengan parameter nilai maksimum overshoot (Mp) dan nilai settling time (ts) dari respon kontrol yang di hasilkan. Maksimum overshoot (Mp) adalah nilai puncak kurva tanggapan diukur dari satuan. Sedangkan waktu turun/settling time (ts) adalah waktu yang diperlukan untuk menanggapi kurva agar dapat mencapai dan tetap berada dalam gugus akhir ukuran yang disederhanakan dengan presentase mutlak harga akhirnya (biasanya 2% atau 5%).

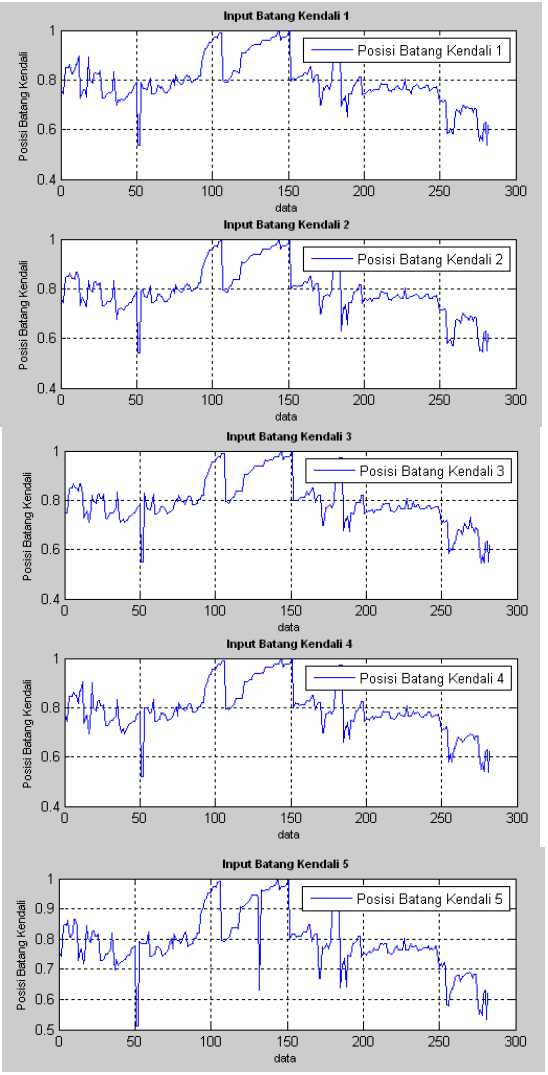
Halaman ini sengaja dikosongkan

BAB IV
PENGUJIAN DAN ANALISA HASIL SIMULASI

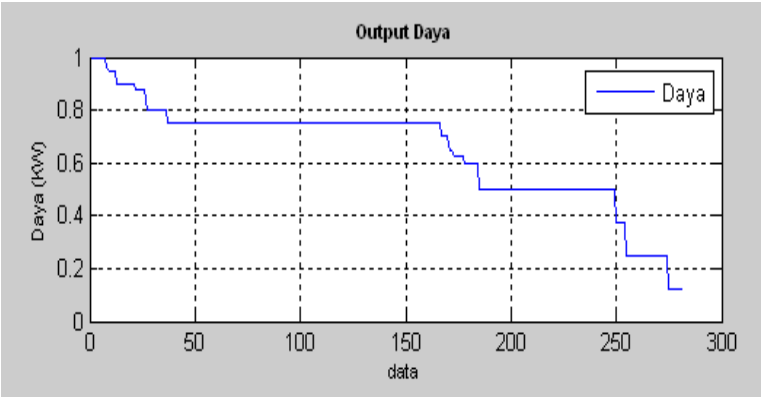
Pada bab ini akan dijelaskan mengenai hasil simulasi dari sistem pengendalian daya reaktor nuklir. Rancangan ini tidak mungkin di implementasikan secara langsung pada reaktor di BATAN Bandung.Karena itu rancangan sistem pemodelan dan pengendalian yang dibuat hanya dilakukan melalui program simulasi dengan menggunakan perangkat lunak matlab. Data – data proses variabel yang digunakan dalam perancangan sistem pengendalian daya reaktor nuklir ini didapat dari data yang ter-*record* pada Log book ReaktorBATAN Bandung. *Plant* yang dimodelkan adalah Teras Reaktor pada Pusat Teknologi Nuklir Bahan dan Radiometri (PTNBR) BATAN Bandung.Data yang digunakan adalah daya reaktor dan posisi batang kendali.Data input-output diambil dari *real plant* selama 3 tahun, sehingga jumlah data adalah 357 data. Dari data tersebut yang digunakan adalah 326 data, karena ada data yang direkam ketika *plant* sedang *shutdown*.

4.1. Pemodelan Plant menggunakan JST

Identifikasi pemodelan *plant* dengan JST melalui dua tahap yaitu *training* dan *validasi*.*Input* dan target untuk pemodelan diperoleh dari data *real plant*, dari semua data yang ada akan dibagi menjadi dua untuk digunakan dalam *training* dan *validasi.*Pada tugas akhir ini akan digunakan 282 data untuk *training* dan 44 data untuk *validasi.*Data input yang akan digunakan adalah posisi batang kendali sedangkan data output adalah daya reaktor. Data input dan output tersebut akan dipelajari oleh JST untuk mendapatkan hubungan masukandan keluaran dalam membentuk pola dalam pembelajaran guna mendapatkan pemodelan *plant* berdasarkan JST.

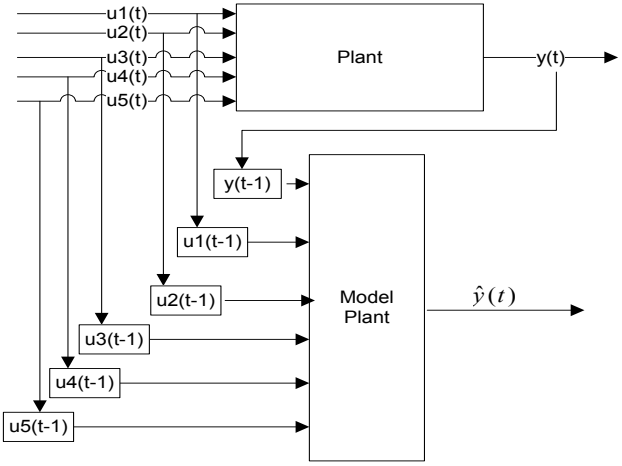


Grafik 4.1 Data input plant reaktor



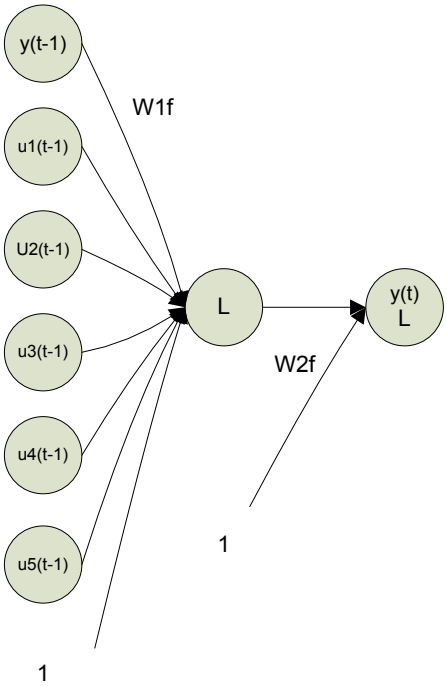
Grafik 4.2 Data Output Plant

Sistem identifikasi berbasis jaringan syaraf tiruan untuk sistem nonlinear dilakukan dengan melakukan pendekatan model menggunakan struktur *NNARX (Neural Network AutoRegressive eXternal input)*. Dimana menggunakan hubungan masukan dan keluaran untuk mendapatkan suatu model proses dari sebuah *plant*. Gambar 4.1 menunjukkan diagram model NNARX untuk sistem ini.



Gambar 4.1 NNARX Model pada Sistem Pemodelan Plant[7]

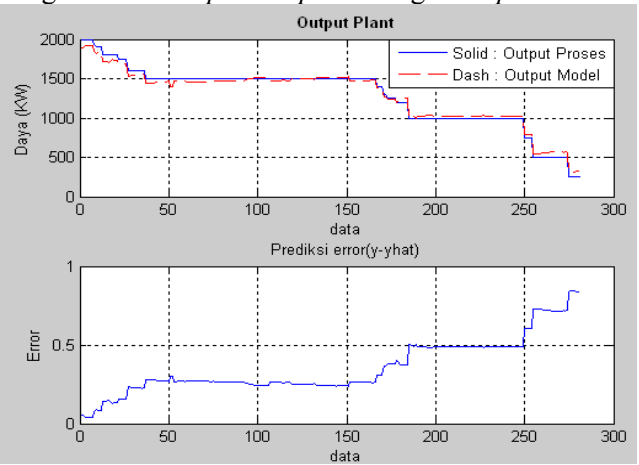
Pemodelan *plant* dilakukan menggunakan jaringan syaraf tiruan dengan metode *Multi Layer Perceptron (MLP)*. Arsitektur struktur JST *Multi layer perceptron* ini terdiri dari tiga layer yaitu *input layer, hidden layer*, dan *output layer*. Prosedur pertama dalam pemodelan dengan JST adalah melakukan *eksperiment.Eksperiment* disini adalah melakukan percobaan untuk mendapatkan model yang paling baik, yaitu dengan merubah parameter JST seperti jumlah *hidden node* dan *history length*. Percobaan dilakukan dengan merubah nilai *history length* mulai dari 1 sampai 5, sedangkan jumlah *hidden node* dimulai dari 1 sampai 15 untuk mendapatkan nilai RMSE yang paling kecil dan VAF yang paling besar. Dan didapatkan struktur JST sebagai berikut :



Gambar 4.2 Struktur JST untuk pemodelan plant

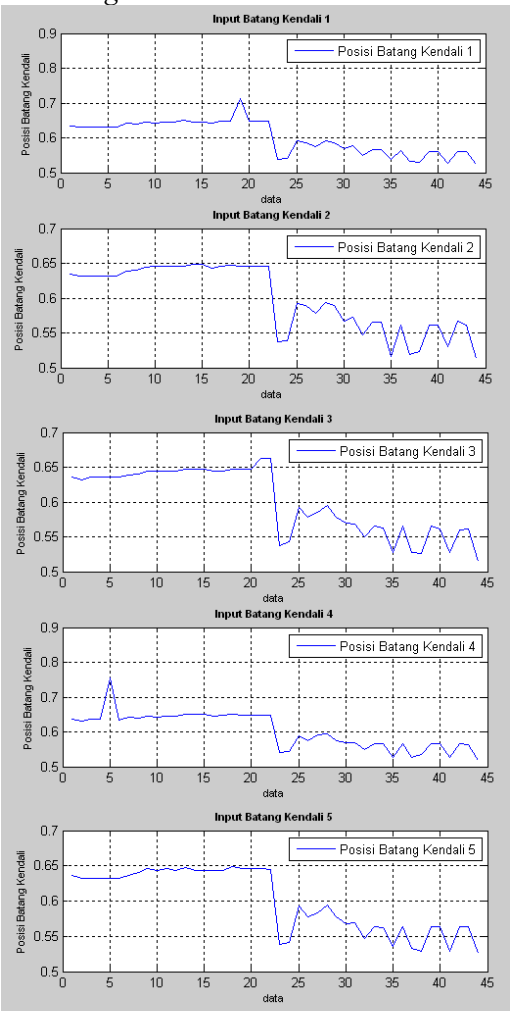
Pada tugas akhir ini arsitektur JST terdiri dari tiga *layer* yaitu *input layer, hidden layer*, dan *output layer.* Variabel yang ada dalam JST yang digunakan adalah sebagai berikut; *input* (u), *output* (y), bobot dari *input layer* ke *hidden layer* (w1f), bobot dari *hidden layer* ke *output* (w2f). *Plant* terdiri dari 5*input* dan satu *output*.Hasil dari *training* pemodelan *plant* ini berupa bobot-bobot yang tersimpan dalam *fileforward* yang nantinya digunakan pada pengendalian*direct inverse control*.

Gambar 4.2 menunjukkan struktur Jaringan Syaraf Tiruan yang paling baik digunakan untuk pemodelen *plant.* Setelah dilakukan *eksperiment* diperoleh hasil yang paling baik adalah dengan 1 *history length* dan 1 *hidden node*. Fungsi aktivasi yang digunakan pada *hidden layer* adalah *linier* sedangkan pada *output layer* juga menggunakan fungsi aktivasi *linear*, dengan struktur tersebut dihasilkan RMSE sebesar 0.0209 dan VAF sebesar 98.8682. Grafik 4.3 menunjukkan *input* yang digunakan untuk proses *training* yaitu 326 data, grafik 4.4 menunjukkan *output* daya yang merupakan pasangan dari setiap data *input* posisi batang kendali, sedangkan grafik 4.5 menunjukkan perbandingan antara *outputreal plant* dengan *output* dari JST.

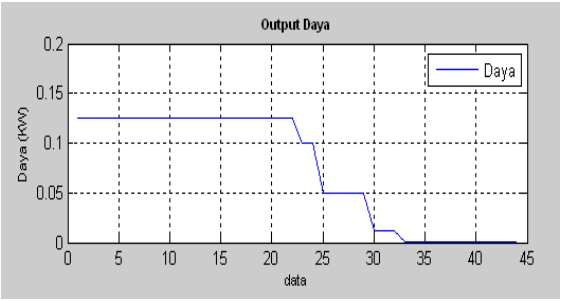


Grafik 4.3 Output training JST dan error

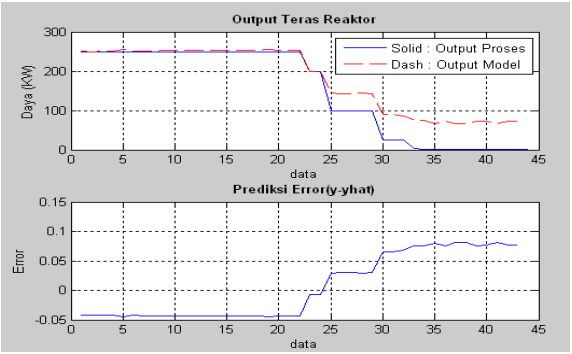
Setelah *plant* dimodelkan, dilakukan proses validasi. Grafik di bawah ini merupakan grafik yang menunjukkan pasangan data *input output* yang digunakan untuk proses *validasi*, data ini adalah 44 data yang berbeda dengan data yang digunakan dalam proses *training*.



Grafik 4.4 Input validasipemodelan plant



Grafik 4.5Output Validasi Pemodelan Plant



Grafik 4.6 Output validasi JST dan Error

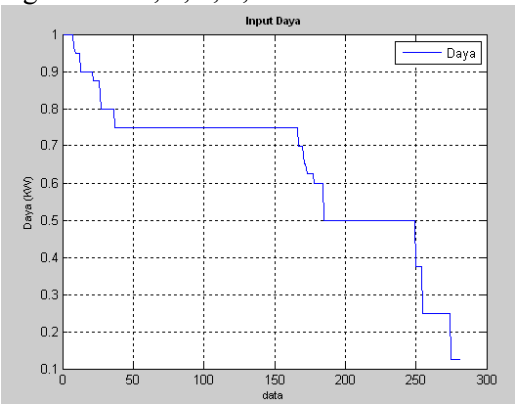
Grafik diatas adalah grafik yang menunjukkan perbandingan antara *output real plant* (biru) dengan output pemodelan JST (merah), dan *error* yang terjadi pada setiap pasangan data. Setelah melakukan proses training maka akan dihasilkan bobot w1f dan w2f, nilai dari bobot ini disimpan untuk digunakan pada proses validasi. Pada proses validasi pasangan data yang digunakan adalah 44 data yang tidak digunakan pada proses training. Pada proses validasi dihasilkan nilai RMSE sebesar 0.1261 dan nilai VAF sebesar 83.2816. Grafik 4.6 menunjukkan *input* yang digunakan untuk proses *validasi* yaitu 44 data, grafik 4.7 menunjukkan *output* temperatur yang merupakan pasangan dari setiap data *input* untuk *validasi*,

sedangkan grafik 4.8 menunjukkan perbandingan antara *output real plant* dengan *output* dari *validasi* JST. *Output real plant* digambarkan dengan warna biru sedangkan *output* dari *validasi* JST digambarkan secara *dash* dengan warna merah.

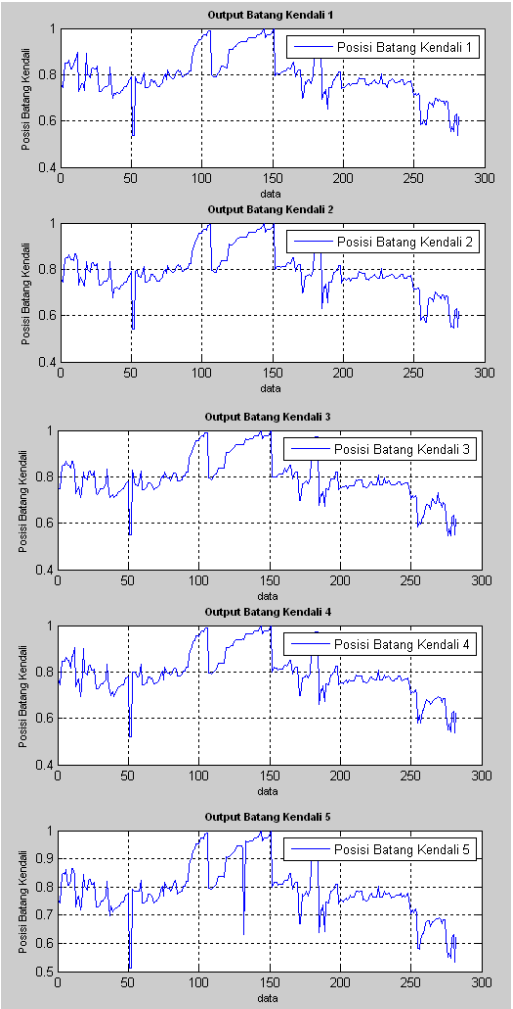
4.2. Pemodelan Pengendali menggunakan JST

Pemodelan pengendali dengan JST mempunyai langkah yang hampir sama dengan pemodelan *plant* dengan JST, yang membedakan adalah struktur JST yang digunakan untuk pemodelan pengendali adalah kebalikan dari struktur JST yang digunakan dalam pemodelan *plant* atau dapat dikatakan pemodelan pengendali adalah *inverse* dari pemodelan *plant*. Maka pada pemodelan pengendali pasangan data yang digunakan adalah daya *demand* dari operator sebagai data *input*, sedangkan posisi batang kendali sebagai data *output*. Tahap pertama adalah mencari struktur JST untuk pemodelan pengendali yang paling baik, maka dilakukan *eksperiment* dengan merubah nilai *history length* dan jumlah *hidden node* untuk mendapatkan nilai RMSE yang paling kecil dan nilai VAF yang paling besar.

Grafik 4.7 adalah *input* yang digunakan untuk pemodelan pengendali yaitu sebanyak 282 data, grafik 4.10 adalah *output*untuk pemodelan pengendali yang terdiri dari *output* posisi batang kendali 1, 2, 3, 4, dan 5.



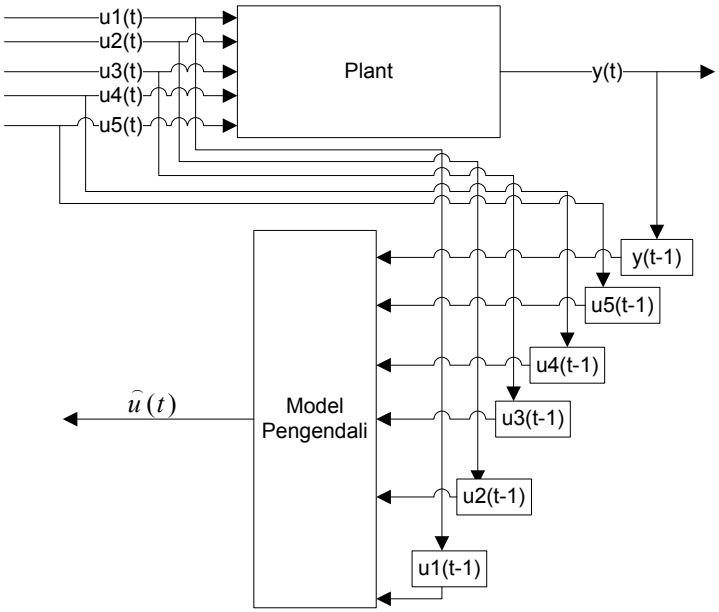
Grafik 4.7 Input Training Pemodelan Pengendali



Grafik 4.8 Output Training Pemodelan Pengendali

Sistem identifikasi pemodelan pengendali berbasis jaringan syaraf tiruan ini pun dilakukan dengan melakukan pendekatan model menggunakan struktur NNARX (Neural Network

AutoRegressive eXternal input). Adapun diagram model NNARX untuk sistem ini adalah sebagai berikut:

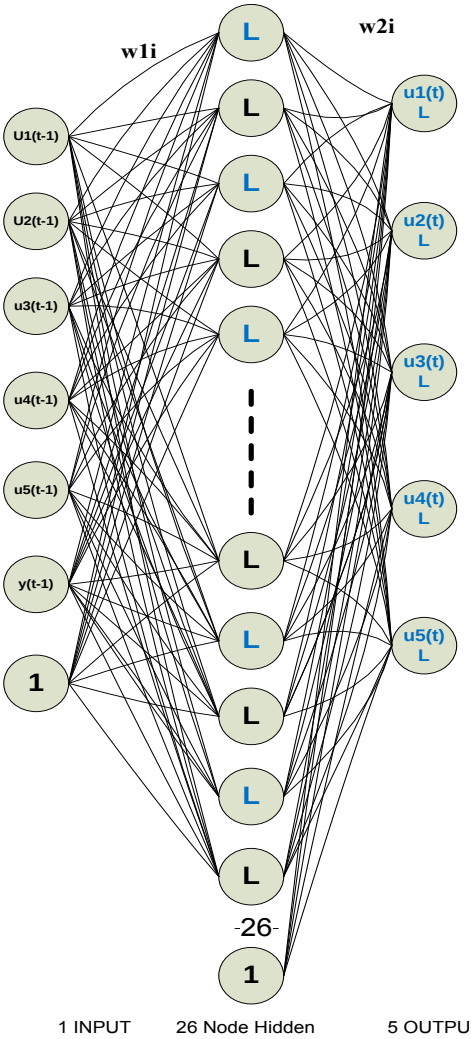


Gambar 4.3 NNARX Model pada Sistem Pemodelan Pengendali[7]

Pemodelan pengendali dilakukan menggunakan jaringan syaraf tiruan dengan metode *Multi Layer Perceptron (MLP)*. Arsitektur strutktur JST *Multi layer perceptron* pada pemodelan pengendali ini terdiri dari tiga layer yaitu *input layer, hidden layer,* dan *output layer*. Pada tugas akhir ini akan dilakukan eksperimen untuk mendapatkan arsitektur struktur JST yang dapat memodelkan pengendali secara tepat. Eksperimen disini yaitu dengan melakukan perubahan parameter JST seperti *hidden node* dan *history length* pada struktur JST untuk mendapatkan model pengendali yang paling baik.

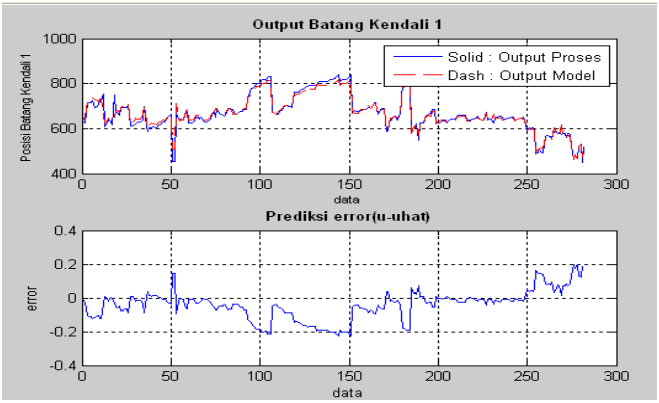
Dari hasil penelitian yang dilakukan didapatkan struktur JST pemodelan pengendali dengan 1 *history length* dan

26*hiddennode*. Fungsi aktivasi yang digunakan pada *hidden layer* dan *output layer* adalah *linear*.Hal tersebut dimaksudkan untuk mendapatkan bobot *inverse* yang tepat untuk pengendali DIC.

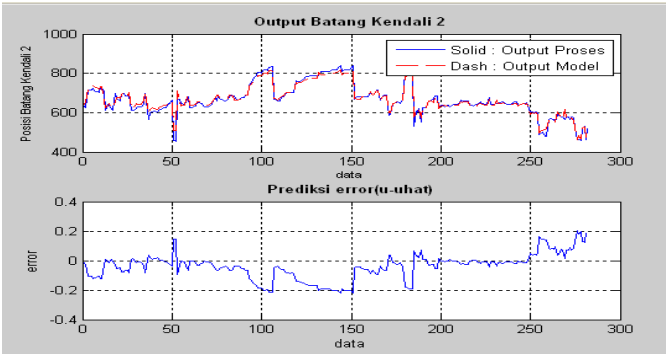


Gambar 4.4 Struktur JST untuk Pemodelan Pengendali

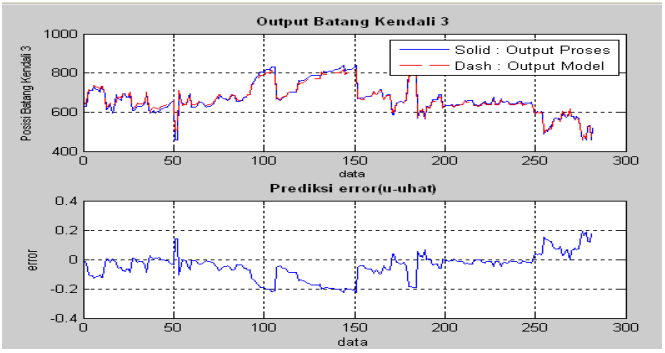
Pada proses *training* model ini menghasilkan RMSE sebesar 0.0178 dan VAF sebesar 96.5654 untuk *output* posisi batang kendali 1, untuk *output* posisi batang kendali 2 menghasilkan RMSE sebesar 0.0161 dan VAF sebesar 97.2107, untuk *output* posisi batang kendali 3 menghasilkan RMSE sebesar 0.0137 dan VAF sebesar 97.9039, untuk *output* posisi batang kendali 4 menghasilkan RMSE sebesar 0.0189 dan VAF sebesar 96.1168, untuk *output* posisi batang kendali 5 menghasilkan RMSE sebesar 0.0262 dan VAF sebesar 92.5715. Grafik 4.9 adalah perbandingan antara *output real plant* dengan *output* JST untuk posisi batang kendali 1, grafik 4.10 menunjukkan perbandingan antara *output real plant* dengan *output* JST posisi batang kendali 2, grafik 4.11 adalah perbandingan antara *output real plant* dengan *output* JST untuk posisi batang kendali 3, grafik 4.12 menunjukkan perbandingan antara *output real plant* dengan *output* JST posisi batang kendali 4, sedangkan grafik 4.13 adalah perbandingan antara *output real plant* dengan *output* JST untuk posisi batang kendali 5. *Output plant* ditunjukkan dengan warna biru sedangkan *output* pemodelan JST ditunjukkan dengan warna merah.Hasil dari *training* pemodelan pengendali ini berupa bobot-bobot yang tersimpan dalam *fileinverse* yang nantinya digunakan pada pengendalian *direct inverse control*.



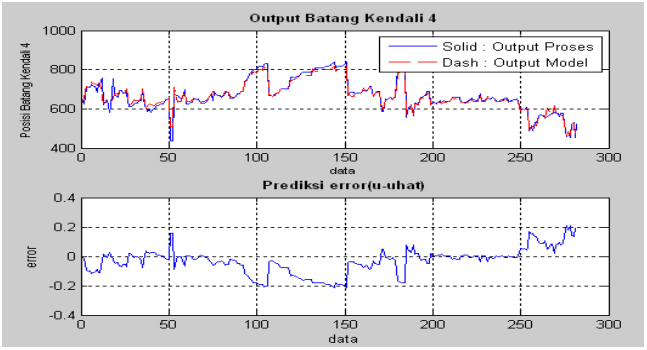
Grafik 4.9 Output Training JST dan Error (batang kendali 1)



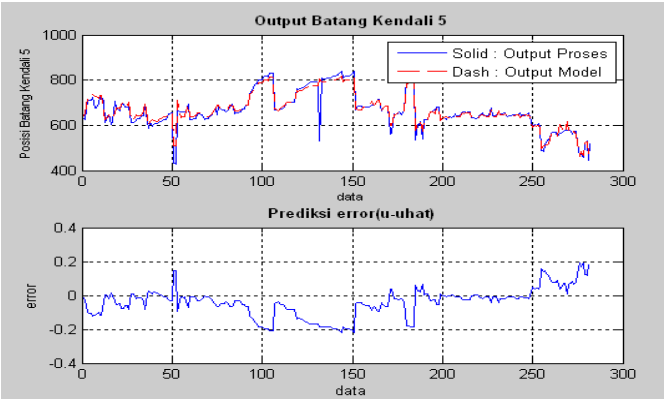
Grafik 4.10 Output Training JST dan Error (batang kendali 2)



Grafik 4.11 Output Training JST dan Error (batang kendali 3)

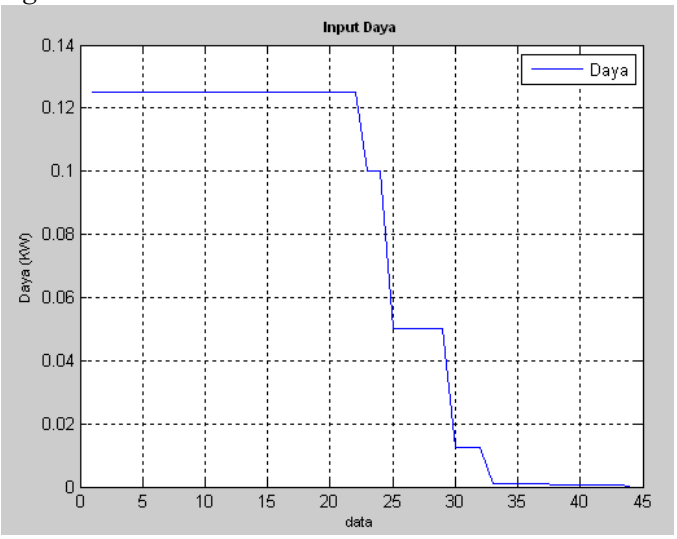


Grafik 4.12 Output Training JST dan Error (batang kendali 4)

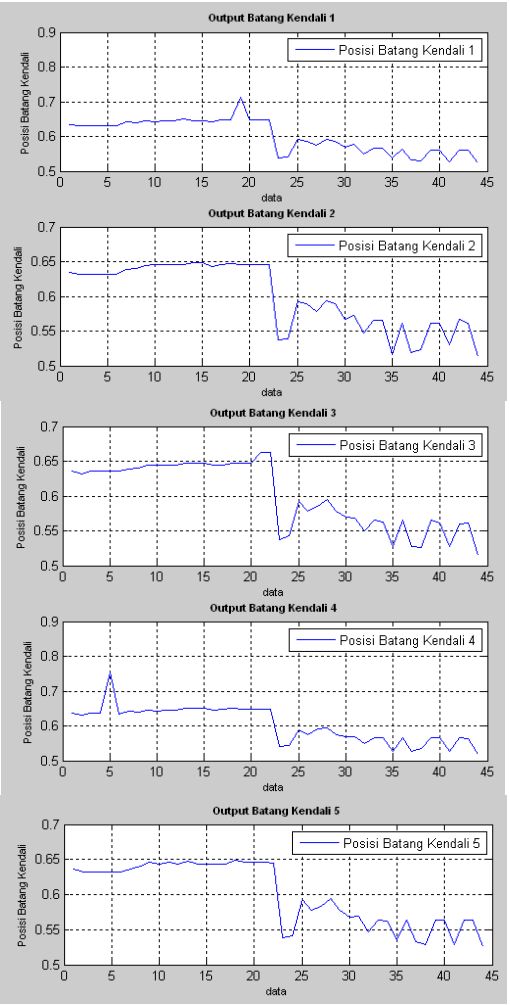


Grafik 4.13 Output Training JST dan Error (batang kendali 5)

Setelah melakukan proses training maka akan dihasilkan bobot w1i dan w2i, nilai dari bobot ini disimpan untuk digunakan pada proses *validasi*. Pada proses *validasi* pasangan data yang digunakan adalah 44 data yang tidak digunakan pada proses *training*.

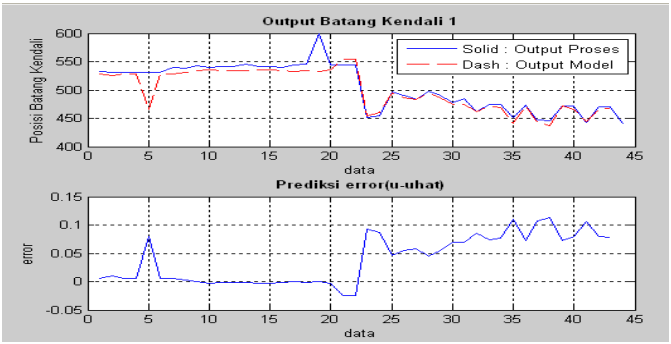


Grafik 4.14 Input Validasi Pemodelan Pengendali

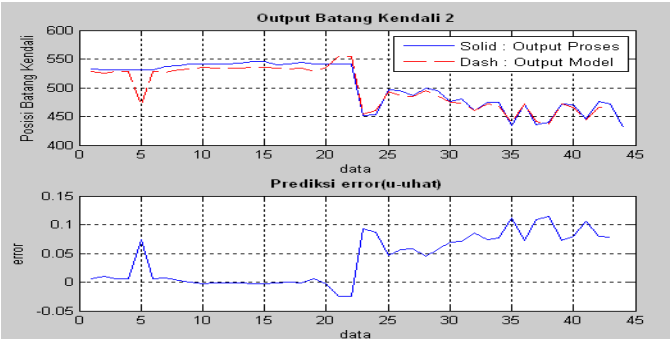


afik 4.15 Output Validasi Pemodelan Pengendali

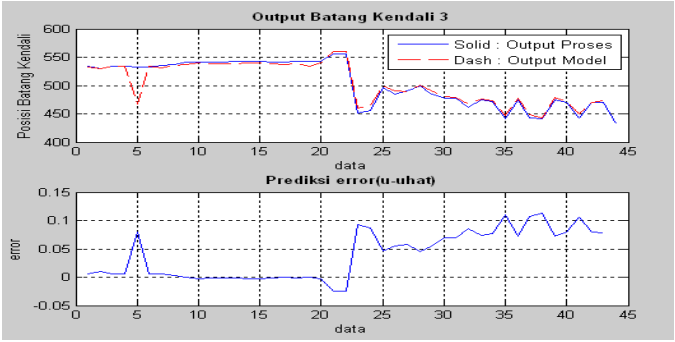
Grafik 4.14 adalah *input* yang digunakan untuk validasi yaitu sebanyak 44 data, gambar 4.15 adalah *output* untuk *validasi* yang terdiri dari *output* posisi batang kendali 1, 2, 3, 4 dan 5.



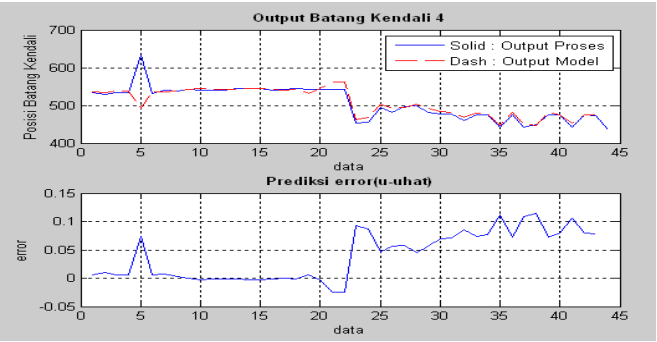
Grafik 4.16 Output Validasi JST dan Error (batang kendali 1)



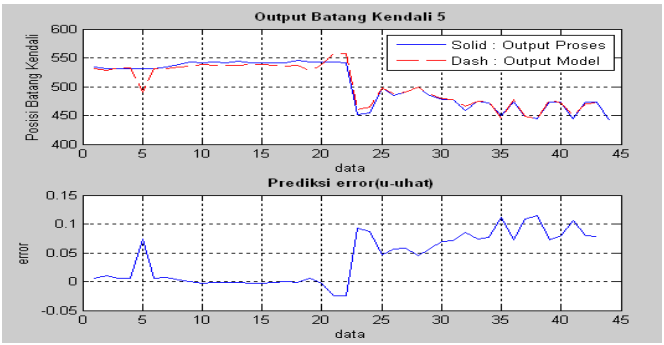
Grafik 4.17 Output Validasi JST dan Error (batang kendali 2)



Grafik 4.18 Output Validasi JST dan Error (batang kendali 3)



Grafik 4.19 Output Validasi JST dan Error (batang kendali 4)



Grafik 4.20 Output Validasi JST dan Error (batang kendali 5)

Pada proses *validasi* dihasilkan nilai RMSE sebesar 0.0185 dan VAF sebesar 87.1010 untuk *output* posisi batang kendali 1, untuk *output* posisi batang kendali 2 menghasilkan RMSE sebesar 0.0137 dan VAF sebesar 92.4092, untuk *output* posisi batang kendali 3 menghasilkan RMSE sebesar 0.0129 dan VAF sebesar 91.8641, untuk *output* posisi batang kendali 4 menghasilkan RMSE sebesar 0.0278 dan VAF sebesar 70.5352, untuk *output* posisi batang kendali 5 menghasilkan RMSE sebesar 0.0098 dan VAF sebesar 94.9508. Grafik 4.18 adalah perbandingan antara *output real plant* dengan *output* JST untuk posisi batang kendali 1, grafik 4.19 menunjukkan perbandingan

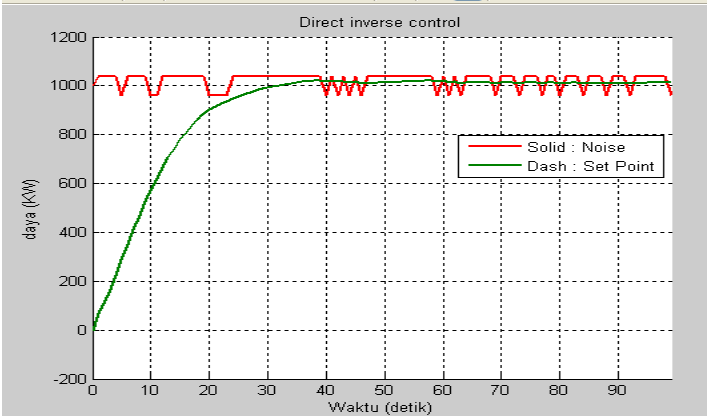
antara *output real plant* dengan *output* JST posisi batang kendali 2, grafik 4.20 adalah perbandingan antara *output real plant* dengan *output* JST untuk posisi batang kendali 3, grafik 4.21 menunjukkan perbandingan antara *output real plant* dengan *output* JST posisi batang kendali 4, sedangkan grafik 4.22 adalah perbandingan antara *output real plant* dengan *output* JST untuk posisi batang kendali 5. *Output plant* ditunjukkan dengan warna biru sedangkan *output* pemodelan JST ditunjukkan dengan warna merah.

4.3. Uji dan Simulasi Pengendali DIC

Setelah melakukan pemodelan *plant* dan pemodelan pengendali dengan JST maka kita mendapatkan bobot dari pemodelan *plant* yaitu w1f dan w2f yang disimpan dalam file *forward* dan bobot dari pemodelan pengendali yaitu W1i dan W2i yang disimpan dalam file *inverse*. Bobot tersebut digunakan untuk melakukan simulasi *direct inverse control*.

4.3.1 Uji Performansi DIC

Uji performansi DIC, dengan memberikan *noise* yang memberikan gangguan pada sistem pengendalian. Berikut grafik simulasi DIC dengan sinyal ganguan;



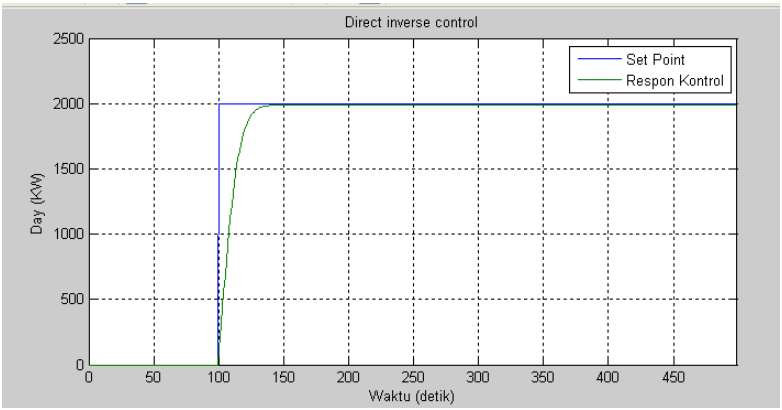
Grafik 4.21 Simulasi DIC dengan sinyal gangguan

Grafik 4.21 menunjukkan hasil dari uji performansi DIC dengan sinyal gangguan.Grafik hijau merupakan respon sistem sedangkan grafik biru merupakan sinyal gangguan.

Dari grafik uji performansi DIC di atas didapatkan hasil pengujian pada *set point* 1000 Kw dengan menambahkan sinyal gangguan, Dapat dilihat pada grafik di atas, bahwa perfomansi pengendalian *Direct Inverse Control* ini baik, dengan ditunjukkannya respon sistem pengendalian yang mengikuti *set point* sinyal gangguan dengan baik. *Set point* diberikan pada daya sebesar 500kW dengan variasi gangguan

4.3.2 Uji Respon Set Point Sinyal Step

Pada pengujian ini, akan dilakukan pengujian performansi sistem *Direct Inverse Control* dengan memberikan sistem *set point* berupa sinyal step.



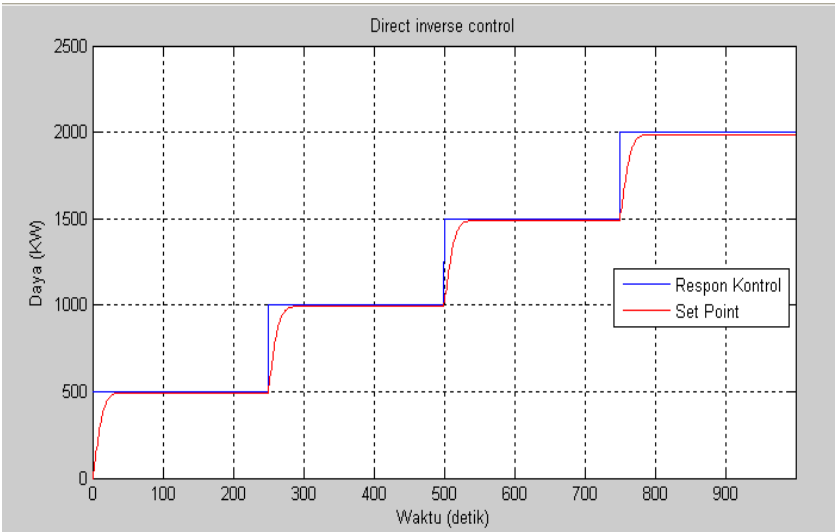
Grafik 4.22 Simulasi uji sinyal step

Pada grafik 4.22 di atas, garis warna hijau adalah proses variabel pada sistem *Direct Inverse Control*, sedangkan garis biru adalah set point berupa sinyal step yang diberikan pada sistem. Berdasarkan pada grafik tersebut, diketahui, bahwa perfomansi pengendalian *Direct Inverse Control* ini baik, dengan

ditunjukkannya respon sistem pengendalian yang mengikuti *set point* berupa sinyal step.

4.3.3 Simulasi DIC

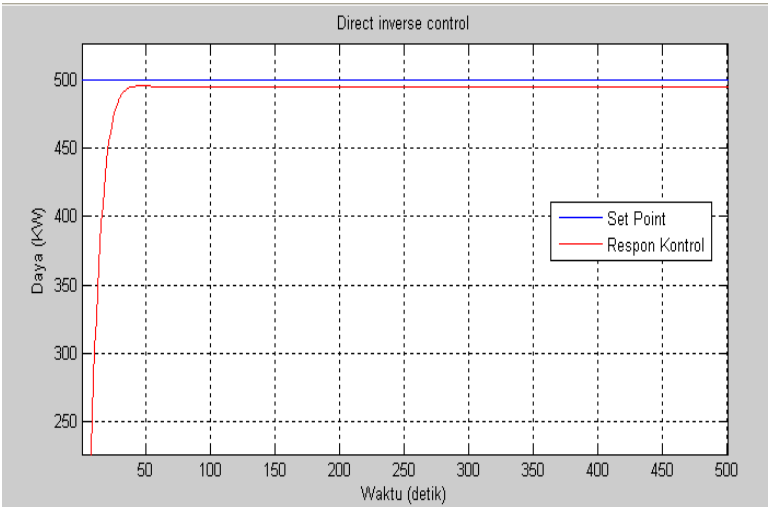
Simulasi dilakukan dengan mengubah nilai *set point* sebanyak empat kali, yaitu pada 500kW, 1000kW, 1500kW, 2000kW. Grafik 4.22 menunjukkan respon sistem dari *direct inverse contro*l.



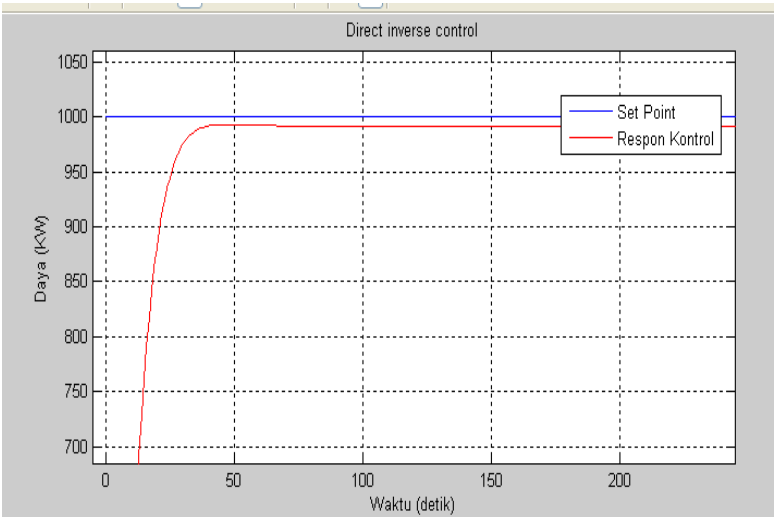
Grafik 4.23 Respon simulasi direct invers control berbasis JST

Tabel 4.1 Respon Pengendali DIC

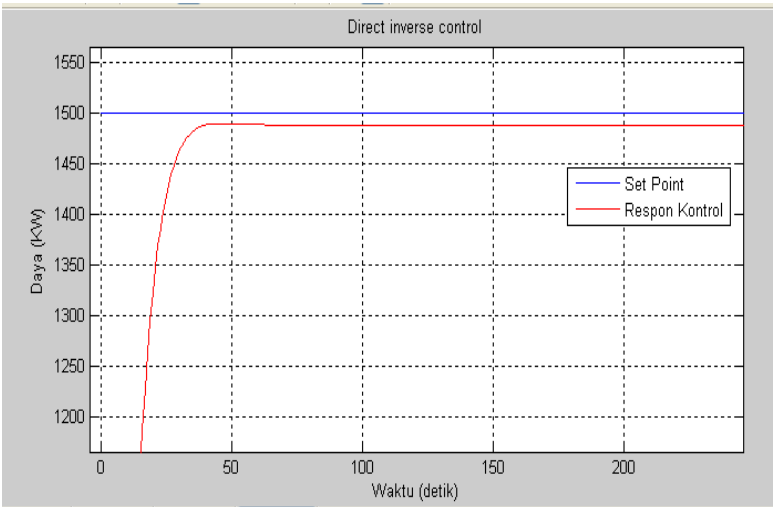
DIC	500 kW	1000 kW	1500 kW	2000 kW
max. overshoot (%)	0	0	0	0
settling time (detik)	43	39	22	41
rise time (detik)	20	19.26	19.23	19.21
delay time (detik)	8.83	8.8	8.8	8.79
Ess	5	8.34	11.27	14



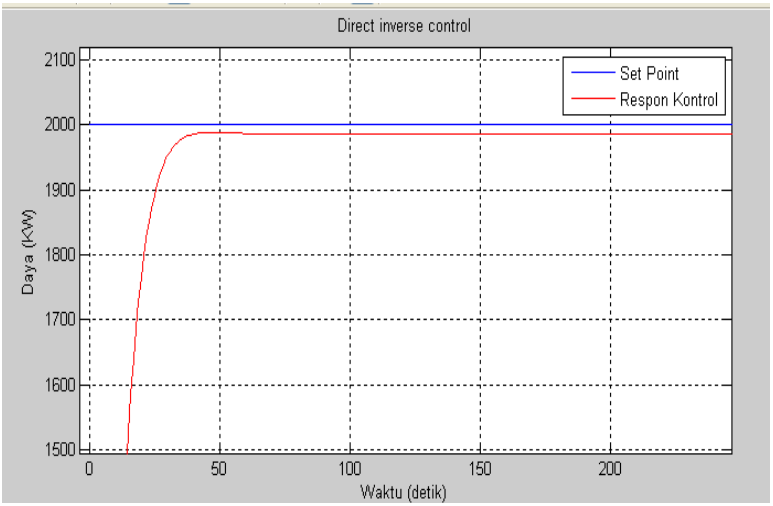
Grafik 4.24 Perbesaran respon simulasi pada set point 500 kW



Grafik 4.25 Perbesaran respon simulasi pada set point 1000 kW

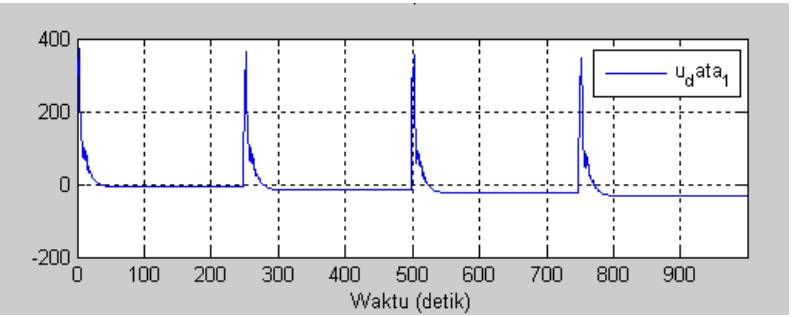


Grafik 4.26 Perbesaran respon simulasi pada set point 1500 kW



Grafik 4.27 Perbesaran respon simulasi pada set point 2000kW

Dari grafik respon diatas didapatkan hasil respon pengendalian pada 1000 data, Pengendalian daya reaktor ini cukup bagus didalam mengikuti setpoin. Dapat diketahui bahwa pengendalian mengikuti *setpoint*dengan *maximum overshoot* 0% untuk semua *set point* dan *error steadystate* 5 pada *set point* 500 KW, 8.34 pada *set point* 1000 KW, 11.27 pada *set point* 1500 KW dan 14 pada *set point* 2000 KW.



Grafik 4.28 Sinyal control simulasi Direct Invers control berbasis JST

Sinyal *control* pada pengendalian daya reaktor berbasis JST diperlihatkan pada grafik 4.28. Pada gambar tersebut dapat dilihat bahwa sinyal *control* mulai stabil ketika setelah melewati waktu dimana terjadi perubahan *set point.* Terdapat empat waktu perubahan *setpoint*. Hal ini disebabkan karena pada waktu-waktu tersebut pengendalian daya berbasis JST masih melakukan pembelajaran atau training.

4.4 Performansi Pengendali PID

Untuk mengetahui performansi dari sistem pengendalian daya reaktor menggunakan pengendali PID, maka dibuat persamaan keadaan menggunakan persamaan dinamika reaktor titik[10] sehingga diperoleh persamaan-persamaan sebagai berikut :

$$\dot{P} = \frac{\rho-\beta}{\Lambda} P + \lambda . C \quad \text{..... (4.1)}$$

$$\dot{C} = -\frac{\beta}{\Lambda}P + \lambda.C \quad \text{..... (4.2)}$$

$$M_f C_f \frac{dT_f}{dt} = P(T) - P_1(T) \quad \text{..... (4.3)}$$

$$M_c C_c \frac{dT_c}{dt} = P_1(T) - P_2(T) \quad \text{..... (4.4)}$$

$$P_1(t) = \frac{1}{R}\left[T_f(t) - T_c(t)\right] \quad \text{..... (4.5)}$$

$$P_2(T) = 2W.C_c[T_c(t) - T_i(t)] \quad \text{..... (4.6)}$$

Karena :

$$\rho = \rho_{eks} + \alpha_f T_f + \alpha_m T_c \quad \text{..... (4.7)}$$

sehingga persamaan 4.1 menjadi :

$$\dot{P} = -\frac{\beta}{\Lambda}P + \lambda.C + \frac{P}{\Lambda}(\rho_{eks} + \alpha_f T_f + \alpha_m T_c) \quad \text{..... (4.8)}$$

dimana,

P : daya reaktor (watt)
C : konsentrasi prekursor neutron
ρ : reaktivitas reaktor
Λ : waktu generasi neutron rata-rata
T_f : temperatur bahan bakar
T_c : temperatur pendingin
ρ_{eks} : reaktivitas ekstenal
λ : tetapan disintegrasi prekursor neutron

Harga daya, konsentrasi neutron precursor dan temperatur dari waktu ke waktu selalu berubah maka variabel-variabel tersebut merupakan variabel keadaan. Dengan menyatakan $P = X_1; C = X_2; T_f = X_3; T_c = X_4$.

Maka persamaan dinamika reaktor dapat ditulis dalam bentuk persamaan keadaan sebagai berikut :

$$\dot{X} = A.X + B.U \quad \text{..... (4.9)}$$

$$Y = C.X + D.U \quad \text{..... (4.10)}$$

Karena output dari *plant* reaktor berupa daya maka :

$$y = [1 \quad 0 \quad 0 \quad 0]X_1 \quad \text{..... (4.11)}$$

Untuk proses linierisasi, pada persamaan dinamika reaktor titik dimisalkan berharga sama dengan nol. Persamaan (4.8) jika diuraikan lebih lanjut maka akan didapat persamaan sebagai berikut :

$$\dot{P} = -\frac{\beta}{\Lambda}P + \lambda.C + \frac{P}{\Lambda}\rho_{eks} + \frac{P}{\Lambda}\alpha_f T_f + \frac{P}{\Lambda}\alpha_m T_c) \quad \text{..... (4.12)}$$

atau

$$\dot{X}_1 = -\frac{\beta}{\Lambda}\mathrm{X}_1 + \lambda \mathrm{X}_2 + \frac{\alpha_F.\mathrm{X}_1}{\Lambda}\mathrm{X}_3 + \frac{\alpha_m.\mathrm{X}_1}{\Lambda}\mathrm{X}_4 + \frac{P}{\Lambda}U \quad \text{..... (4.13)}$$

$$\dot{X}_2 = -\frac{\beta}{\Lambda}\mathrm{X}_1 + \lambda.\mathrm{X}_2 \quad \text{..... (4.14)}$$

$$\dot{X}_3 = \frac{1}{M_F.C_F}\mathrm{X}_1 - \frac{1}{\tau_f}\mathrm{X}_3 + \frac{1}{\tau_f}\mathrm{X}_4 \quad \text{..... (4.15)}$$

$$\dot{X}_4 = \frac{1}{\tau_c}\mathrm{X}_3 - \left(\frac{1}{\tau_c} + \frac{2W}{M_c}\right)\mathrm{X}_4 \quad \text{..... (4.16)}$$

Semua variabel $\dot{X}$ dimisalkan sama dengan nol, sehingga diperoleh persamaan sebagai berikut :

$$-\frac{\beta}{\Lambda}\mathrm{X}_1 + \lambda \mathrm{X}_2 + \frac{\alpha_F.\mathrm{X}_1}{\Lambda}\mathrm{X}_3 + \frac{\alpha_m.\mathrm{X}_1}{\Lambda}\mathrm{X}_4 + \frac{P}{\Lambda}U = 0 \quad \text{..... (4.17)}$$

$$-\frac{\beta}{\Lambda}\mathrm{X}_1 + \lambda.\mathrm{X}_2 = 0 \quad \text{..... (4.18)}$$

$$\frac{1}{M_F.C_F}\mathrm{X}_1 - \frac{1}{\tau_f}\mathrm{X}_3 + \frac{1}{\tau_f}\mathrm{X}_4 = 0 \quad \text{..... (4.19)}$$

$$\frac{1}{\tau_c}\mathrm{X}_3 - \left(\frac{1}{\tau_c} + \frac{2W}{M_c}\right)\mathrm{X}_4 = 0 \quad \text{..... (4.20)}$$

Persamaan (4.18), (4.19) dan (4.20) dibuat dalam bentuk X_1,

$$\mathrm{X}_2 = -\frac{\beta}{\lambda\Lambda}\mathrm{X}_1 \quad \text{..... (4.21)}$$

$$\mathrm{X}_3 = \frac{\tau_f}{M_F.C_F}\mathrm{X}_1 + \mathrm{X}_4 \quad \text{..... (4.22)}$$

$$\left(\frac{1}{\tau_c} + \frac{2W}{M_c}\right)\mathrm{X}_4 = \frac{1}{\tau_c}\mathrm{X}_3 \quad \text{..... (4.23)}$$

$$\mathrm{X}_4 = \frac{M_c}{M_c + \tau_c 2W}\mathrm{X}_3 \quad \text{..... (4.24)}$$

Subsitusi persamaan (4.22) dan (4.23) serta memasukkan nilai $M_c, \tau_c, 2W$ menghasilkan persamaan

$$X_3 = \frac{2\tau_f}{M_F.C_F} X_1 \quad \text{..... (4.25)}$$

Subsitusi persamaan (4.22) dan (4.24)

$$X_4 = \frac{M_c}{M_c+\tau_c 2W} \frac{2\tau_f}{M_F.C_F} X_1 \quad \text{..... (4.26)}$$

Persamaan (4.16) dan (4.17) disubtitusi sehingga didapat persamaan sebagai berikut :

$$\frac{\alpha_F.X_1}{\Lambda} X_3 + \frac{\alpha_m.X_1}{\Lambda} X_4 + \frac{X_1}{\Lambda} U = 0 \quad \text{..... (4.27)}$$

$$(\alpha_F. X_3 + \alpha_m. X_4 + U) \frac{1}{\Lambda} X_1 = 0 \quad \text{..... (4.28)}$$

Jika $X_1 \neq 0$ maka persamaan (4.22) dapat direduksi menjadi

$$(\alpha_F. X_3 + \alpha_m. X_4 + U) = 0 \quad \text{..... (4.29)}$$

Dengan mensubtitusi persamaan (4.24) dan (4.25) ke persamaan (4.28) diperoleh

$$U = \left(\alpha_F \frac{2\tau_f}{M_F.C_F} + \alpha_m \frac{M_c}{M_c+\tau_c 2W} \frac{2\tau_f}{M_F.C_F} \right) X_1 \quad \text{..... (4.30)}$$

Untuk menetapkan setiap nilai *steady state* untuk x1, variabel R digunakan, yang nilainya dapat diatur dalam interval

$$n_o \leq R \leq R_{max} \quad \text{..... (4.31)}$$

Dimana n_o merupakan densitas neutron ketika reaktivitas eksternal bernilai nol.

Persamaan dinamika reaktor titik (4.13) – (4.16) dinotasikan sebagai ($\overline{X_1}, \overline{X_2}, \overline{X_3}, \overline{X_4}, \overline{U}$) dimana

$$\overline{X_1} = R \quad \text{..... (4.32)}$$

$$\overline{X_2} = -\frac{\beta}{\lambda\Lambda} R \quad \text{..... (4.33)}$$

$$\overline{X_3} = \frac{2\tau_f}{M_F.C_F} R \quad \text{..... (4.34)}$$

$$\overline{X_4} = \frac{M_c}{M_c+\tau_c 2W}\frac{2\tau_f}{M_F.C_F}\mathrm{R} \quad \text{..... (4.35)}$$

$$\overline{U} = \left(\alpha_F \frac{2\tau_f}{M_F.C_F} + \alpha_m \frac{M_c}{M_c+\tau_c 2W}\frac{2\tau_f}{M_F.C_F}\right)\mathrm{R} \quad \text{..... (4.36)}$$

Prosedur linierisasi untuk ke empat persamaan diferensial berdasarkan sistem

$$\dot{X}_1 = \mathrm{f}_1(\mathrm{X}_1, \mathrm{X}_2, \mathrm{X}_3, \mathrm{X}_4, \mathrm{U}) \quad \text{..... (4.37)}$$
$$\dot{X}_2 = \mathrm{f}_2(\mathrm{X}_1, \mathrm{X}_2, \mathrm{X}_3, \mathrm{X}_4, \mathrm{U}) \quad \text{..... (4.38)}$$
$$\dot{X}_3 = \mathrm{f}_3(\mathrm{X}_1, \mathrm{X}_2, \mathrm{X}_3, \mathrm{X}_4, \mathrm{U}) \quad \text{..... (4.39)}$$
$$\dot{X}_4 = \mathrm{f}_4(\mathrm{X}_1, \mathrm{X}_2, \mathrm{X}_3, \mathrm{X}_4, \mathrm{U}) \quad \text{..... (4.40)}$$

Jika dinotasikan dalam bentuk ($\overline{X_1}, \overline{X_2}, \overline{X_3}, \overline{X_4}, \overline{U}$) menjadi

$$Q_1 = \mathrm{X}_1 - \overline{X_1} \quad \text{..... (4.41)}$$
$$Q_2 = \mathrm{X}_2 - \overline{X_2} \quad \text{..... (4.42)}$$
$$Q_3 = \mathrm{X}_3 - \overline{X_3} \quad \text{..... (4.43)}$$
$$Q_4 = \mathrm{X}_4 - \overline{X_4} \quad \text{..... (4.44)}$$

Maka persamaan keadaan untuk *plant* reaktor adalah sebagai berikut :

$$\begin{vmatrix} \dot{Q}_1 \\ \dot{Q}_2 \\ \dot{Q}_3 \\ Q_4 \end{vmatrix} = \begin{vmatrix} M_1 & M_2 & M_3 & M_4 \\ N_1 & N_2 & N_3 & N_4 \\ O_1 & O_2 & O_3 & O_4 \\ S_1 & S_2 & S_3 & S_4 \end{vmatrix} \begin{vmatrix} \mathrm{Q}_1 \\ \mathrm{Q}_2 \\ \mathrm{Q}_3 \\ \mathrm{Q}_4 \end{vmatrix} + \begin{vmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{vmatrix} \mathrm{U}$$

Dimana notasi-notasi untuk matrix A dan matrix B diperoleh melalui persamaan-persamaan sebagai berikut :

$$M_1 = \frac{\delta \mathrm{f}_1(\overline{X_1}, \overline{X_2}, \overline{X_3}, \overline{X_4}, \overline{U})}{\delta \mathrm{X}_1} \qquad M_3 = \frac{\delta \mathrm{f}_1(\overline{X_1}, \overline{X_2}, \overline{X_3}, \overline{X_4}, \overline{U})}{\delta \mathrm{X}_3}$$

$$M_2 = \frac{\delta \mathrm{f}_1(\overline{X_1}, \overline{X_2}, \overline{X_3}, \overline{X_4}, \overline{U})}{\delta \mathrm{X}_2} \qquad M_4 = \frac{\delta \mathrm{f}_1(\overline{X_1}, \overline{X_2}, \overline{X_3}, \overline{X_4}, \overline{U})}{\delta \mathrm{X}_4}$$

$$N_1 = \frac{\delta f_2(\overline{X_1}, X_2, X_3, X_4, U)}{\delta X_1} \qquad N_3 = \frac{\delta f_2(X_1, X_2, X_3, X_4, U)}{\delta X_3}$$
$$N_2 = \frac{\delta f_2(X_1, X_2, X_3, X_4, U)}{\delta X_2} \qquad N_4 = \frac{\delta f_2(X_1, X_2, X_3, X_4, U)}{\delta X_4}$$

$$O_1 = \frac{\delta f_3(\overline{X_1}, X_2, X_3, X_4, U)}{\delta X_1} \qquad O_3 = \frac{\delta f_3(X_1, X_2, X_3, X_4, U)}{\delta X_3}$$
$$O_2 = \frac{\delta f_3(X_1, \overline{X_2}, X_3, X_4, U)}{\delta X_2} \qquad O_4 = \frac{\delta f_{31}(X_1, X_2, X_3, X_4, U)}{\delta X_4}$$

$$S_1 = \frac{\delta f_4(\overline{X_1}, X_2, X_3, X_4, U)}{\delta X_1} \qquad S_3 = \frac{\delta f_4(X_1, X_2, \overline{X_3}, X_4, U)}{\delta X_3}$$
$$S_2 = \frac{\delta f_4(X_1, \overline{X_2}, X_3, X_4, U)}{\delta X_2} \qquad S_4 = \frac{\delta f_4(X_1, X_2, X_3, \overline{X_4}, U)}{\delta X_4}$$

$$T_1 = \frac{\delta f_1(\overline{X_1}, X_2, X_3, X_4, U)}{\delta U} \qquad T_3 = \frac{\delta f_3(\overline{X_1}, X_2, \overline{X_3}, X_4, U)}{\delta U}$$
$$T_2 = \frac{\delta f_2(\overline{X_1}, X_2, X_3, \overline{X_4}, U)}{\delta U} \qquad T_4 = \frac{\delta f_4(\overline{X_1}, X_2, X_3, \overline{X_4}, U)}{\delta U}$$

Hasil dari persamaan dinamika reaktor titik (4.13) – (4.16) serta persamaan keadaan di atas adalah :

$$\begin{vmatrix} \dot{Q}_1 \\ \dot{Q}_2 \\ \dot{Q}_3 \\ \dot{Q}_4 \end{vmatrix} = \begin{vmatrix} -\frac{\beta}{\Lambda} & \lambda & \frac{\alpha_F}{\Lambda}R & \frac{\alpha_m}{\Lambda}R \\ \frac{\beta}{\lambda\Lambda} & -\lambda & 0 & 0 \\ \frac{1}{M_F.C_F} & 0 & -\frac{1}{\tau_f} & \frac{1}{\tau_f} \\ 0 & 0 & \frac{1}{\tau_c} & -\left(\frac{1}{\tau_c}+\frac{2W}{M_c}\right) \end{vmatrix} \begin{vmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \end{vmatrix} + \begin{vmatrix} \frac{R}{\Lambda} \\ 0 \\ 0 \\ 0 \end{vmatrix} U$$

Harga konstanta dan perubah keadaan yang digunakan dalam reaktor PTNBR BATAN Bandung[10] adalah sebagai berikut :

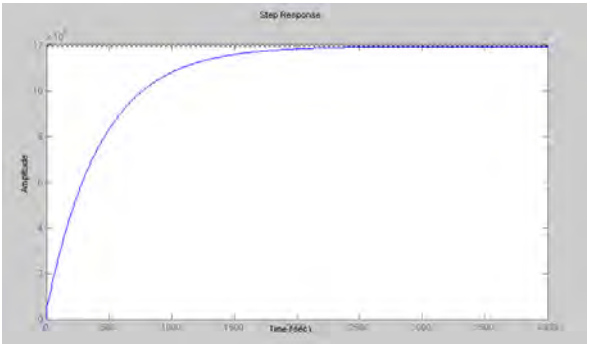
$X_1 = 1000\ Watt;$ $X_4 = 27°C;$
$X_2 = 500\ {}^{n}/_{cm^3};$ $\beta = 0.0027;$
$X_3 = 31°C;$ $\Lambda = 0.000028\ detik;$

$\lambda = 0.0768 detik^{-1};$
$M_f = 169.500 gram;$
$C_f = 0.005899 \left({}^{cal}/_{gr°C}\right);$
$M_c = 19605.000\ gram;$
$\alpha_F = -0.0000306\ {}^{\Delta k}/_{k.C^{-1}};$
$\alpha_m = -0.00054\ {}^{\Delta k}/_{k.C^{-1}};$
$\tau_f = 1.5 detik;$
$\tau_c = 25.2 detik;$
$W = 9600\ {}^{gr}/_{detik};$

Dari persamaan keadaan serta data di atas maka didapat *transfer function* dari *plant* reaktor sebagai berikut :

$$\frac{O(s)}{I(s)} = \frac{3.57 * 10^7 s^3 + 6.304 * 10^7 s^2 + 2.805 * 10^7 s + 1.798 * 10^6}{s^4 + 98.2 s^3 + 164.8 s^2 + 65.35 s + 0.1506}$$

Berdasarkan persamaan *transfer function* di atas maka dapat diketahui performansi dari sistem pengendalian daya reaktor menggunakan pengendali PID, yaitu :



Grafik 4.29 Tanggapan transient pengendalian daya reaktor dengan PID

Grafik di atas menunjukkan tanggapan transient dari sistem pengendalian daya menggunakan pengendali PID. Dapat dilihat bahwa respon dari sistem menghasilkan pengendalian yang kurang maksimal jika kita bandingkan dengan pengendalian daya reaktor menggunakan *direct inverse control* berbasis JST, hal

tersebut dapat kita lihat dari lamanya waktu yang dibutukan pengendali PID untuk mencapai keadaan *steady*. Untuk lebih jelasnya dapat dilihat di bawah performansi dari pengendalian daya reaktor menggunakan pengendali PID :

Rise Time	: 1. 2702*10^3 detik
Settling Time	: 1. 6744*10^3 detik
Maximum Overshoot	: 0%
Peak Time	: 2.9657e*10^3 detik

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari permasalahan yang di jabarkan dalam bab pertama, maka diperoleh hasil dari penelitian Tugas Akhir ini sebagai berikut:

1. Pemodelan *plant* dengan JST yang terbaik didapat pada *history length* 1 dan jumlah *hidden node* 9 dengan menghasilkan RMSE = 0.0209 dan VAF = 98.8682.
2. Pemodelan kontroller yang terbaik didapat pada *history length* 1 dan *hidden node* 26 dengan menghasilkan RMSE sebesar 0.0178 dan VAF sebesar 96.5654 untuk *output* posisi batang kendali 1, untuk *output* posisi batang kendali 2 menghasilkan RMSE sebesar 0.0161 dan VAF sebesar 97.2107, untuk *output* posisi batang kendali 3 menghasilkan RMSE sebesar 0.0137 dan VAF sebesar 97.9039, untuk *output* posisi batang kendali 4 menghasilkan RMSE sebesar 0.0189 dan VAF sebesar 96.1168, untuk *output* posisi batang kendali 5 menghasilkan RMSE sebesar 0.0262 dan VAF sebesar 92.5715.
3. Simulasi *Direct Inverse Control* yang diperoleh respon sistem memberikan hasil yang baik, hal ini dapat diketahui dari uji *noise* dimana sistem masih dapat mempertahankan *setpoint* pada titik 1000 KW.
4. Pengendalian daya menggunakan DIC menghasilkan settling time yang lebih singkat jika dibandingkan dengan pengendalian daya menggunakan PID.

5.2 Saran

Beberapa saran yang perlu disampaikan dalam laporan ini dalam rangka pengembangan penelitian ini antara lain adalah sebagai berikut:

1. Untuk mengembangan penelitian ini maka perlu dilakukan metode pencatatan data yang lebih baik, misalnya dengan merekam data dalam jumlah yang lebih banyak.

2. Untuk mengembangan penelitian ini maka perlu dilakukan penelitian dengan menggunakan struktur model yang lain seperti NNOE, algoritma pembelajaran yang lain, seperti Gauss Newton.
3. Perlu diaplikasikan secara *online* pada *real plant* dalam penelitian selanjutnya.

LAMPIRAN A

DATA INPUT OUTPUT TRAINING
PEMODELAN PLANT DAN KONTROLLER DENGAN JST

POSISI BATANG KENDALI					DAYA
1	2	3	4	5	(KW)
631	632	628	639	630	2000
627	626	629	625	624	2000
714	712	709	708	710	2000
714	713	712	711	711	2000
714	713	712	711	710	2000
724	723	727	724	723	2000
692	708	714	713	678	2000
692	708	714	713	678	1930
708	703	701	686	692	1900
725	729	728	729	728	1900
725	729	728	729	728	1900
752	705	693	759	708	1900
610	614	611	619	613	1800
627	644	623	637	643	1800
639	637	636	645	645	1800
635	618	597	581	605	1800
615	612	614	611	618	1800
748	699	689	755	708	1800
672	676	668	675	678	1800
671	669	668	668	672	1800
661	661	661	660	662	1800
691	692	689	691	691	1750
695	698	692	695	695	1750

693	685	678	679	680	1750
693	685	678	679	680	1750
705	691	692	681	686	1750
611	613	609	611	612	1600
612	613	608	611	612	1600
611	613	611	611	613	1600
620	620	629	621	619	1600
627	628	629	628	629	1600
628	629	629	628	630	1600
628	629	633	636	636	1600
636	643	646	644	644	1600
700	700	700	700	690	1600
614	615	615	615	615	1600
585	567	597	606	587	1500
604	600	598	586	615	1500
601	604	607	601	597	1500
607	608	596	581	604	1500
605	602	604	601	608	1500
604	602	604	602	608	1500
615	615	617	616	616	1500
615	615	617	616	616	1500
628	627	626	626	627	1500
628	627	627	626	627	1500
628	627	627	626	627	1500
646	648	645	648	649	1500
653	648	649	648	649	1500
665	665	654	654	654	1500
453	456	460	437	431	1500
452	454	460	437	430	1500

666	666	697	672	664	1500
644	668	655	670	660	1500
643	646	643	670	660	1500
639	642	640	654	659	1500
639	643	639	654	660	1500
658	656	658	655	659	1500
677	681	692	699	693	1500
626	625	626	625	625	1500
626	625	626	625	625	1500
627	628	627	628	628	1500
634	634	634	633	630	1500
652	652	651	652	651	1500
651	652	651	652	652	1500
644	645	645	647	648	1500
649	648	643	650	645	1500
636	630	626	629	624	1500
628	628	628	628	628	1500
630	630	630	631	630	1500
635	635	635	635	636	1500
642	642	644	646	646	1500
664	663	662	664	665	1500
675	672	673	673	674	1500
652	658	668	642	645	1500
688	687	688	688	687	1500
670	670	671	674	671	1500
663	667	667	668	667	1500
658	659	667	668	667	1500
657	659	658	659	658	1500
667	667	669	667	669	1500

683	682	685	681	681	1500
689	689	685	685	693	1500
681	681	681	680	680	1500
686	684	655	657	650	1500
665	665	655	655	655	1500
665	665	655	655	655	1500
672	673	661	662	660	1500
675	675	675	675	674	1500
675	675	675	675	674	1500
687	691	688	689	689	1500
687	691	688	689	689	1500
735	737	737	735	736	1500
747	746	748	746	748	1500
779	766	761	758	763	1500
779	778	778	780	780	1500
789	788	786	784	785	1500
800	803	803	802	801	1500
800	803	803	802	801	1500
801	802	802	800	801	1500
817	817	817	817	817	1500
818	818	818	818	818	1500
814	815	815	815	815	1500
828	830	830	828	830	1500
830	830	830	829	830	1500
831	833	830	831	832	1500
672	669	668	668	667	1500
666	664	664	666	666	1500
663	661	661	664	664	1500
663	661	672	670	672	1500

676	677	676	675	674	1500
681	682	681	681	681	1500
691	691	692	692	692	1500
703	704	703	701	703	1500
703	703	703	702	703	1500
700	704	703	701	703	1500
697	704	703	702	703	1500
695	703	703	702	701	1500
764	763	761	761	762	1500
763	760	758	761	758	1500
763	760	761	760	761	1500
763	762	763	764	765	1500
770	768	768	767	768	1500
775	773	773	767	774	1500
780	782	778	779	778	1500
788	785	787	787	786	1500
788	790	788	787	794	1500
791	789	787	787	794	1500
791	790	787	787	794	1500
791	789	787	787	794	1500
791	789	788	791	790	1500
791	790	788	787	530	1500
808	808	808	808	808	1500
808	808	807	808	806	1500
808	808	807	808	806	1500
808	808	808	808	808	1500
808	808	808	808	808	1500
808	808	808	808	808	1500
818	818	818	818	818	1500

818	818	818	818	818	1500
818	818	818	818	818	1500
821	820	821	821	821	1500
831	829	829	830	830	1500
840	840	839	839	840	1500
808	808	808	808	808	1500
818	818	818	818	818	1500
818	818	818	818	818	1500
818	818	818	818	818	1500
821	820	821	821	821	1500
831	829	829	830	830	1500
840	840	839	839	840	1500
669	670	670	670	670	1500
679	682	672	687	687	1500
676	681	671	686	686	1500
676	681	671	686	686	1500
677	678	677	678	678	1500
681	681	681	682	681	1500
684	681	681	680	680	1500
684	681	681	680	680	1500
690	692	691	691	687	1500
695	696	702	695	691	1500
691	691	689	692	691	1500
694	694	694	694	693	1500
714	713	714	714	713	1500
716	715	713	714	713	1500
664	663	662	664	665	1500
678	678	678	678	678	1400
689	689	688	689	686	1400

689	689	688	689	686	1400
689	689	688	689	686	1400
588	588	585	586	562	1300
589	589	585	586	563	1300
645	646	646	649	650	1250
640	645	645	648	645	1250
653	651	654	653	651	1250
654	654	655	659	659	1250
640	645	645	648	645	1250
656	656	656	656	656	1200
664	664	664	664	664	1200
805	805	805	805	805	1200
806	806	805	804	805	1200
813	814	815	816	817	1200
813	814	815	816	817	1200
813	814	815	816	817	1200
581	531	569	555	536	1000
607	608	606	606	605	1000
603	603	604	604	605	1000
619	619	617	620	618	1000
546	551	563	564	540	1000
625	627	628	628	627	1000
627	624	625	625	625	1000
627	624	625	625	625	1000
655	659	662	659	660	1000
655	659	662	659	660	1000
667	667	663	669	666	1000
667	667	663	669	667	1000
683	684	689	691	679	1000

683	684	689	691	679	1000
620	622	623	624	620	1000
625	638	633	632	632	1000
625	638	633	632	632	1000
633	631	635	627	627	1000
636	635	635	634	634	1000
640	639	638	642	638	1000
631	630	629	632	632	1000
640	638	639	642	638	1000
641	641	640	639	649	1000
641	641	640	640	642	1000
641	641	640	639	642	1000
641	641	640	639	642	1000
660	663	660	660	660	1000
658	660	657	658	658	1000
658	658	659	658	656	1000
658	658	659	658	658	1000
658	658	659	658	658	1000
636	637	636	635	634	1000
636	637	636	635	635	1000
633	633	632	632	631	1000
641	640	632	632	630	1000
641	644	637	638	637	1000
648	648	648	648	648	1000
653	653	653	654	654	1000
643	646	644	645	643	1000
640	641	641	641	641	1000
640	641	641	641	641	1000
642	641	641	641	641	1000

674	676	677	676	675	1000
640	641	641	641	641	1000
641	641	641	641	641	1000
641	642	641	641	643	1000
624	648	666	660	658	1000
645	645	651	651	652	1000
649	650	651	652	652	1000
653	653	655	658	654	1000
656	655	655	654	654	1000
649	648	641	643	640	1000
643	641	642	640	641	1000
643	641	642	640	641	1000
645	644	643	645	645	1000
645	646	647	647	648	1000
652	655	653	657	647	1000
652	655	653	657	648	1000
636	641	642	646	654	1000
636	638	642	640	642	1000
643	641	645	644	647	1000
648	647	648	647	642	1000
648	647	648	647	649	1000
650	650	651	650	654	1000
619	619	617	620	618	1000
596	598	593	593	593	750
604	602	603	601	604	750
598	599	597	596	596	750
604	604	605	607	607	750
596	598	593	593	593	750
490	486	489	488	489	500

493	496	507	513	485	500
509	500	499	486	514	500
491	479	525	532	534	500
492	479	526	532	534	500
561	563	562	560	561	500
571	568	571	571	569	500
565	565	567	565	570	500
567	566	569	566	570	500
553	553	554	554	554	500
587	589	591	561	567	500
585	588	589	567	567	500
576	576	574	575	575	500
583	577	576	575	577	500
574	565	614	582	578	500
577	579	579	579	579	500
576	577	576	576	575	500
561	563	562	560	561	500
576	576	577	576	575	500
572	572	571	576	573	500
521	521	525	522	522	250
464	462	459	460	462	250
482	464	484	482	475	250
467	460	459	454	460	250
521	521	525	522	522	250
530	530	531	529	530	250
451	462	460	452	447	250
522	522	522	525	524	250
533	533	534	534	534	250
531	531	530	530	531	250

531	531	534	533	531	250
531	531	534	533	531	250
531	531	533	633	531	250
531	531	533	531	531	250
540	537	536	540	534	250
538	538	537	537	538	250
543	541	541	542	543	250
540	542	541	540	541	250
542	542	541	541	543	250
542	542	541	541	541	250
546	543	543	545	544	250
541	545	543	545	541	250
541	545	543	545	541	250
540	540	541	541	541	250
544	542	541	543	541	250
545	544	543	545	545	250
599	542	543	543	543	250
544	542	543	543	543	250
544	542	556	543	543	250
544	542	556	543	542	250
451	451	451	453	452	200
455	453	456	456	455	200
496	497	497	495	498	100
491	495	485	482	485	100
483	486	491	497	490	100
498	499	499	498	499	100
491	495	485	482	485	100
478	476	478	478	477	25
485	481	477	477	478	25

461	460	461	460	459	25
475	475	474	474	474	2.25
475	475	472	474	472	2
451	434	442	442	450	2
473	472	475	474	473	1.75
447	436	443	442	448	1.75
446	440	441	448	444	1.5
472	472	475	474	473	1.5
472	471	471	474	473	1.25
443	445	443	443	444	1
471	476	470	474	473	1
471	472	471	473	473	1
440	431	432	435	442	0.5

LAMPIRAN B
BOBOT MODEL PLANT JST DAN MODEL PENGENDALI JST

Bobot (W1f) Antara Input Layer Ke Hidden Layer Model Plant JST

Hidden Node	Input Node						
	1	2	3	4	5	6	Bias
1	0.9326	0.03195	0.03289	0.025575	0.02768	0.02607	-0.01378

Bobot (W2f) Antara Hidden Layer Ke Output Layer Model Plant JST

Hidden Node	Input Node	
	1	2
1	0.9349	-0.0146

Bobot (W1i) Antara Input Layer Ke Hidden Layer Model Pengendali JST

Hidden Node	Input Node						
	1	2	3	4	5	6	7
1	0.0296	0.0963	-0.3360	0.4507	0.0516	-0.1884	-0.2640
2	0.1524	0.2235	0.1586	-0.0376	0.4589	0.2086	-0.3613
3	0.2782	-0.0919	-0.2444	-0.2124	0.3554	0.0871	-0.1718
4	0.1180	0.0976	-0.3424	0.4264	-0.0233	-0.2608	-0.1536
5	0.0021	-0.0743	-0.2234	-0.0037	0.3778	-0.4359	0.4288
6	-0.0146	-0.3782	-0.2013	0.3773	-0.1101	-0.4966	0.3883
7	-0.3899	-0.1500	-0.4941	-0.2868	-0.4508	-0.3109	0.2900
8	0.4165	0.0467	-0.2713	-0.3856	-0.1926	-0.0088	-0.4255
9	-0.0674	-0.3729	0.1701	-0.3757	-0.1798	0.1165	-0.0802
10	-0.0730	-0.4185	0.4817	0.0741	-0.0387	-0.3772	-0.1033
11	-0.4707	-0.2105	-0.2439	-0.0336	-0.0119	-0.3670	-0.0871
12	0.0104	0.1768	0.1192	-0.1260	0.2067	-0.1858	0.4740
13	0.4637	-0.0430	0.4718	0.2546	-0.1327	-0.4121	-0.2621
14	-0.2246	-0.0008	-0.0078	-0.0444	0.4414	-0.0912	-0.3914
15	0.2434	-0.4415	-0.1990	0.2060	-0.2318	-0.2177	0.3000
16	-0.3832	-0.4264	0.0971	0.4295	0.2454	0.1306	-0.4552
17	0.2693	-0.1621	0.0434	0.3205	0.4091	-0.2802	0.4600
18	0.2657	-0.1639	-0.3702	-0.4602	0.0846	0.1543	0.3016
19	0.0408	0.4850	0.0522	0.4662	0.4065	-0.3065	0.4606
20	-0.3307	0.2517	-0.3509	0.3880	-0.4433	-0.4054	0.2657
21	0.4301	0.1465	-0.2415	0.1388	0.0567	0.4184	0.1082
22	-0.2918	0.3829	0.0287	-0.4890	-0.4970	0.0582	0.4890
23	-0.4802	0.0119	0.3134	0.4773	-0.3468	-0.0824	0.0626
24	0.1016	-0.1238	0.2185	-0.1145	-0.0925	-0.3896	-0.0041
25	0.1074	0.3838	0.0008	-0.1003	-0.2666	0.0527	-0.2470
26	-0.0311	-0.2542	0.3940	0.0103	0.0353	-0.3887	-0.3040

Bobot (W2i) Antara Hidden Layer Ke Output Layer
Model Pengendali JST

Hidden Node	Input Node					
	1	2	3	4	5	6
1	-0.0073	-0.1547	-0.0735	-0.0729	0.0206	-0.4136
2	0.2109	-0.4084	0.3908	0.0889	0.4141	-0.4592
3	0.4876	0.1316	-0.2758	-0.4083	0.1633	0.2516
4	-0.1654	-0.1218	0.2969	-0.2494	-0.0003	-0.1762
5	-0.2798	0.3874	0.3583	0.0938	0.1348	-0.4335
6	-0.0073	-0.1547	-0.0735	-0.0729	0.0206	-0.4136
7	0.2109	-0.4084	0.3908	0.0889	0.4141	-0.4592

Hidden Node	Input Node					
	7	8	9	10	11	12
1	0.4233	-0.4726	-0.0236	0.2306	0.1063	0.1465
2	-0.4456	-0.4503	-0.3900	0.4810	0.2643	-0.3233
3	-0.4321	-0.2614	-0.0646	0.0882	0.1006	0.3765
4	0.0735	-0.3727	0.4019	0.0032	0.2226	-0.2483
5	-0.0608	0.1303	0.0658	0.0316	-0.0648	-0.1435
6	0.4233	-0.4726	-0.0236	0.2306	0.1063	0.1465
7	-0.4456	-0.4503	-0.3900	0.4810	0.2643	-0.3233

Hidden Node	Input Node					
	13	14	15	16	17	18
1	-0.1967	-0.4040	0.1026	-0.4956	0.1469	-0.1173
2	0.4678	-0.3006	0.3625	-0.3014	0.0857	0.4772
3	-0.3077	0.1071	0.1760	-0.0613	0.2663	0.4361
4	-0.4021	0.2097	0.2850	0.0583	-0.3254	0.3927
5	0.1938	0.3905	0.0366	-0.1149	0.0031	-0.0565
6	-0.1967	-0.4040	0.1026	-0.4956	0.1469	-0.1173
7	0.4678	-0.3006	0.3625	-0.3014	0.0857	0.4772

Hidden Node	Input Node					
	19	20	21	22	23	24
1	0.2515	-0.1589	0.1063	0.1465	-0.1967	-0.4040
2	0.4094	0.3494	0.2643	-0.3233	0.4678	-0.3006
3	-0.1267	-0.3138	0.1006	0.3765	-0.3077	0.1071
4	-0.3837	-0.2913	0.2226	-0.2483	-0.4021	0.2097
5	0.0251	0.0304	-0.0648	-0.1435	0.1938	0.3905
6	0.2515	-0.1589	0.1063	0.1465	-0.1967	-0.4040
7	0.4094	0.3494	0.2643	-0.3233	0.4678	-0.3006

Hidden Node	Input Node	
	25	26
1	0.1026	-0.4956
2	0.3625	-0.3014
3	0.1760	-0.0613
4	0.2850	0.0583
5	0.0366	-0.1149
6	0.1026	-0.4956
7	0.3625	-0.3014

LAMPIRAN C
LISTING PROGRAM

1. Program Scalling Data
   Program ini untuk inisialisasi data input, output, kemudian dilakukan normalisasi data dan menampilkan hasilnya pada grafik. Berikut adalah listing program pada matlab;

```
% ------------- SCALLING DATA --------------
% Program Untuk Scalling Data Input Output

%load data
load data
u1A = u1;
u2A = u2;
u3A = u3;
u4A = u4;
u5A = u5;
y1A = y1;

% Scalling Data
u1s = u1A/max(u1A);
u2s = u2A/max(u2A);
u3s = u3A/max(u3A);
u4s = u4A/max(u4A);
u5s = u5A/max(u5A);
y1s = y1A/max(y1A);

% Menampilkan Grafik Data Input Output
figure(1)
subplot (2,1,1);plot(u1);grid;
  title('Input Batang Kendali 1')
 legend('Posisi Batang Kendali 1');
 xlabel('data')
 ylabel('Posisi Batang Kendali')
subplot (2,1,2);plot(u2);grid;
  title('Input Batang Kendali 2')
 legend('Posisi Batang Kendali 2');
 xlabel('data')
```

```
 ylabel('Posisi Batang Kendali')

figure(2)
subplot (2,1,1);plot(u3);grid;
  title('Input Batang Kendali 3')
 legend('Posisi Batang Kendali 3');
 xlabel('data')
 ylabel('Posisi Batang Kendali')
subplot (2,1,2);plot(u4);grid;
  title('Input Batang Kendali 4')
 legend('Posisi Batang Kendali 4');
 xlabel('data')
 ylabel('Posisi Batang Kendali')

 figure(3)
 subplot (2,1,1);plot(u5);grid;
   title('Input Batang Kendali 5')
 legend('Posisi Batang Kendali 5');
 xlabel('data')
 ylabel('Posisi Batang Kendali')

figure(4)
plot(y1);grid;
  title('Kenaikkan Daya')
 legend('Daya');
 xlabel('data')
 ylabel('Daya (KW)')

% Menampilkan Grafik Data Input Output Setelah
di Scalling
figure(5)
 subplot (2,1,1);plot(u1s);grid;
  title('Input Batang Kendali 1')
 legend('Posisi Batang Kendali 1');
 xlabel('data')
 ylabel('Posisi Batang Kendali')
subplot (2,1,2);plot(u2s);grid;
  title('Input Batang Kendali 2')
 legend('Posisi Batang Kendali 2');
```

```
 xlabel('data')
 ylabel('Posisi Batang Kendali')


 figure(6)
 subplot (2,1,1);plot(u3s);grid;
  title('Input Batang Kendali 3')
 legend('Posisi Batang Kendali 3');
 xlabel('data')
 ylabel('Posisi Batang Kendali')
subplot (2,1,2);plot(u4s);grid;
  title('Input Batang Kendali 4')
 legend('Posisi Batang Kendali 4');
 xlabel('data')
 ylabel('Posisi Batang Kendali')

 figure(7)
  subplot (2,1,1);plot(u5s);grid;
   title('Input Batang Kendali 5')
 legend('Posisi Batang Kendali 5');
 xlabel('data')
 ylabel('Posisi Batang Kendali')

figure(8)
plot(y1s);grid;
  title('Kenaikkan Daya')
 legend('Daya');
 xlabel('data')
 ylabel('Daya (KW)')

 save data_scal u1s u2s u3s u4s u5s y1s;
```

2. Program Pemodelan Plant Berbasis JST

   Pada program ini terdapat beberapa m.file matlab yang saling terkait dalam melakukan training dan validasi model JST, khususnya dengan memanggil data hasil scalling yang kemudian data itu akan dilakukan training dan validasi. Dengan program utama **training.m** untuk training dan **validasi.m** untuk validasi dan beberapa m.file yang mendukung program utama yaitu **settrain.m** untuk memberikan parameter-parameter pelatihan, **marq_rev.m** sebagai algoritma pemodelan JST, **rmse.m** sebagai rumus yang digunakan untuk mendapatkan nilai RMSE, **vaf.m** sebagai rumus yang digunakan untuk mendapatkan nilai VAF, dan **pmntanh.m** sebagai fungsi tangent hiperbolik. Berikut adalah listing program pada matlab;

2.1. training.m

```
% -------------------------- PEMODELAN PLANT JST
--------------------------
% Program Jaringan Syaraf Tiruan untuk
mendapatkan Model Plant JST

clear all
close all
clc

% ------------------------------- TRAINING -----
---------------------------
% Load Data Input Output

load data_scal
load data

u11 = u1s(1:282); %sebagai data input 1
u12 = u2s(1:282); %sebagai data input 2
u13 = u3s(1:282); %sebagai data input 3
u14 = u4s(1:282); %sebagai data input 4
u15 = u5s(1:282); %sebagai data input 5
y11 = y1s(1:282); %sebagai data output
```

```
% Menampilkan Grafik Data Input Output Training
figure(1)
subplot (2,1,1);plot(u11);grid;
  title('Input Batang Kendali
1','FontSize',8,'FontWeight','bold')
 legend('Posisi Batang Kendali 1');
 xlabel('data','FontSize',8)
 ylabel('Posisi Batang Kendali','FontSize',8)
subplot (2,1,2);plot(u12);grid;
  title('Input Batang Kendali
2','FontSize',8,'FontWeight','bold')
 legend('Posisi Batang Kendali 2');
 xlabel('data','FontSize',8)
 ylabel('Posisi Batang Kendali','FontSize',8)

figure(2)
subplot (2,1,1);plot(u13);grid;
  title('Input Batang Kendali
3','FontSize',8,'FontWeight','bold')
 legend('Posisi Batang Kendali 3');
 xlabel('data','FontSize',8)
 ylabel('Posisi Batang Kendali','FontSize',8)
subplot (2,1,2);plot(u14);grid;
  title('Input Batang Kendali
4','FontSize',8,'FontWeight','bold')
 legend('Posisi Batang Kendali 4');
 xlabel('data','FontSize',8)
 ylabel('Posisi Batang Kendali','FontSize',8)

 figure(3)
  subplot (2,1,1);plot(u15);grid;
    title('Input Batang Kendali
5','FontSize',8,'FontWeight','bold')
 legend('Posisi Batang Kendali 5');
 xlabel('data','FontSize',8)
 ylabel('Posisi Batang Kendali','FontSize',8)

 figure(4)
  subplot (2,1,1);plot(y11);grid
```

```
  title('Output
Daya','FontSize',8,'FontWeight','bold')
 legend('Daya');
 xlabel('data','FontSize',8)
 ylabel('Daya (KW)','FontSize',8)

   strvcat('Figure 1, 2, 3 dan 4 adalah grafik
data input output training ',...
   'Tekan sembarang tombol untuk melanjutkan!')
pause;

% Konstruksi dari Matrik Input Training
hist = [1 1 1 1 1 1]
[n_rows,n_col] = size(u11);

% Membuat Matrik Regressor Training
data_latih = zeros(n_rows-1,sum(hist));

for i = 1:hist(1),
   data_latih(:,i) = [zeros(-
1+i,1);y11(1:n_rows-1+1-i)];
end

for j = 1:hist(2),
   data_latih(:,sum(hist(1))+j) = [zeros(-
1+j,1);u11(1:n_rows-1+1-j)];
end

for k = 1:hist(3),
   data_latih(:,sum(hist(1:2))+k) = [zeros(-
1+k,1);u12(1:n_rows-1+1-k)];
end

for l = 1:hist(4),
   data_latih(:,sum(hist(1:3))+l) = [zeros(-
1+l,1);u13(1:n_rows-1+1-l)];
end
```

```
for m = 1:hist(5),
   data_latih(:,sum(hist(1:4))+m) = [zeros(-
1+m,1);u14(1:n_rows-1+1-m)];
end

for n = 1:hist(6),
   data_latih(:,sum(hist(1:5))+n) = [zeros(-
1+n,1);u15(1:n_rows-1+1-n)];
end

PHI = data_latih'; % matrik PHI merupakan
regressor bagi model plant JST

Y = zeros(n_rows-1,1);
Y(:,1) = y11(2:end);
Y_target = Y';

NN = [hist(1) hist(2) hist(3) hist(4) hist(5)
hist(6) 1];

% Konstruksi dari Struktur Jaringan
NetDeff = ['L';'L'];
trparms = settrain;
[W1f,W2f,PI_vec,yhat] =
marq_rev(NetDeff,[],[],PHI,Y_target,trparms);

% Root Mean Square Error
RMSE_training = rmse(y11(1:281)',yhat)
display('Nilai di atas adalah RMSE untuk
Training');

% Variance Accounted For
VAF_training = vaf(y11(1:281),yhat')
display('Nilai di atas VAF untuk Training');

% Prediksi Error
error=y11(1,:)-yhat;
```

```
% Descalling
yhat1=yhat*2000;

% Menampilkan Grafik Hasil Training
figure(5)
subplot(2,1,1);
plot(y1(1:282));
hold on
plot(yhat1,'r--');
grid
 title('Output Plant','FontWeight','bold');
legend('Solid : Output Proses','Dash : Output
Model');
xlabel('data');
ylabel('Daya (KW)');
subplot(2,1,2);
plot(error)
grid
title('Prediksi error(y-yhat)');
xlabel('data');
ylabel('Error');

% Menyimpan Data
save forward NN W1f W2f NetDeff hist
```

2.2. validasi.m

```
% -------------------------- PEMODELAN PLANT JST
--------------------------
% Program Jaringan Syaraf Tiruan untuk
mendapatkan Model Plant JST

clear all
close all
clc

% ------------------------------ VALIDASI -----
--------------------------
```

```
% Load Data Input Output

load data
load data_scal
load forward

u21 = u1s(283:326); %sebagai data input 1
u22 = u2s(283:326); %sebagai data input 2
u23 = u3s(283:326); %sebagai data input 3
u24 = u4s(283:326); %sebagai data input 4
u25 = u5s(283:326); %sebagai data input 5
y21 = y1s(283:326); %sebagai data output

% Menampilkan Grafik Data Input Output Training
figure(1)
subplot (2,1,1);plot(u21);grid;
  title('Input Batang Kendali
1','FontSize',8,'FontWeight','bold')
 legend('Posisi Batang Kendali 1');
 xlabel('data','FontSize',8)
 ylabel('Posisi Batang Kendali','FontSize',8)
subplot (2,1,2);plot(u22);grid;
  title('Input Batang Kendali
2','FontSize',8,'FontWeight','bold')
 legend('Posisi Batang Kendali 2');
 xlabel('data','FontSize',8)
 ylabel('Posisi Batang Kendali','FontSize',8)

figure(2)
subplot (2,1,1);plot(u23);grid;
  title('Input Batang Kendali
3','FontSize',8,'FontWeight','bold')
 legend('Posisi Batang Kendali 3');
 xlabel('data','FontSize',8)
 ylabel('Posisi Batang Kendali','FontSize',8)
subplot (2,1,2);plot(u24);grid;
  title('Input Batang Kendali
4','FontSize',8,'FontWeight','bold')
 legend('Posisi Batang Kendali 4');
```

```
 xlabel('data','FontSize',8)
 ylabel('Posisi Batang Kendali','FontSize',8)

 figure(3)
  subplot (2,1,1);plot(u25);grid;
    title('Input Batang Kendali
5','FontSize',8,'FontWeight','bold')
 legend('Posisi Batang Kendali 5');
 xlabel('data','FontSize',8)
 ylabel('Posisi Batang Kendali','FontSize',8)

 figure(4)
  subplot (2,1,1);plot(y21);grid
  title('Output
Daya','FontSize',8,'FontWeight','bold')
 legend('Daya');
 xlabel('data','FontSize',8)
 ylabel('Daya (KW)','FontSize',8)


   strvcat('Figure 1, 2 dan 3 adalah grafik data
input output training ',...
   'Tekan sembarang tombol untuk melanjutkan!')
pause;

% Membuat Matrik Regressor Validasi

[n_rows,n_col] = size(u21);

data_uji = zeros(n_rows-1,sum(hist));

for i = 1:hist(1),
   data_uji(:,i) = [zeros(-1+i,1);y21(1:n_rows-
1+1-i)];
end

for j = 1:hist(2),
```

```
   data_uji(:,sum(hist(1))+j) = [zeros(-
1+j,1);u21(1:n_rows-1+1-j)];
end

for k = 1:hist(3),
   data_uji(:,sum(hist(1:2))+k) = [zeros(-
1+k,1);u22(1:n_rows-1+1-k)];
end

for l = 1:hist(4),
   data_uji(:,sum(hist(1:3))+l) = [zeros(-
1+l,1);u23(1:n_rows-1+1-l)];
end

for m = 1:hist(5),
   data_uji(:,sum(hist(1:4))+m) = [zeros(-
1+m,1);u24(1:n_rows-1+1-m)];
end

for n = 1:hist(6),
   data_uji(:,sum(hist(1:5))+n) = [zeros(-
1+n,1);u25(1:n_rows-1+1-n)];
end



PHI_uji = data_uji';
PHI_ujis = [PHI_uji;ones(1,size(PHI_uji,2))];

Y_uji = zeros(n_rows-1,1);
Y_uji(:,1) = y21(2:end);
Y_target_uji = Y_uji';

L_hidden = find(NetDeff(1,:)=='L')';      % Letak
dari syaraf linier tersembunyi
H_hidden = find(NetDeff(1,:)=='H')';      % Letak
dari syaraf tanh tersembunyi
L_output = find(NetDeff(2,:)=='L')';      % Letak
dari syaraf linier output
```

```
H_output = find(NetDeff(2,:)=='H')';      % Letak
dari syaraf tanh output
hidden = length(L_hidden)+length(H_hidden);

y1f       =
[zeros(hidden,size(data_uji,1));ones(1,size(data
_uji,1))];
y2f       =
zeros(size(Y_target_uji,1),size(data_uji,1));


h1f = W1f*PHI_ujis;
y1f(H_hidden,:) = pmntanh(h1f(H_hidden,:));
y1f(L_hidden,:) = h1f(L_hidden,:);

h2f = W2f*y1f;
y2f(H_output,:) = pmntanh(h2f(H_output,:));
y2f(L_output,:) = h2f(L_output,:);
yhat=y2f;

% Root Mean Square Error
RMSE_validasi = rmse(y21(1:43)',yhat)
display('Nilai di atas adalah RMSE untuk
validasi');

% Variance Accounted For
VAF_validasi = vaf(y21(1:43),yhat')
display('Nilai di atas VAF untuk validasi');

% Prediksi Error
error=y21(1,:)-yhat;

% Descalling
yhat2=yhat*1500;

% Menampilkan Grafik Hasil Validasi
figure(4)
subplot(2,1,1);
```

```
plot(y1(283:326));
hold on
plot(yhat2,'r--');
grid
 title('Output Teras
Reaktor','FontWeight','bold');
legend('Solid : Output Proses','Dash : Output
Model');
xlabel('data');
ylabel('Daya (KW)');

subplot(2,1,2);
plot(error)
grid
 title('Prediksi Error(y-
yhat)','FontWeight','bold');
xlabel('data');
ylabel('Error');
```

2.3. settrain.m

```
function tr = settrain(trparms,varargin)
%==============================================
============================
%SETTRAIN menentukan parameter untuk algoritma
training.
%   Hanya dibutuhkan untuk menentukan parameter
spesifik pada algoritma
%   training yang dipilih.
%
%   TRPARMS = SETTRAIN
%      Menentukan semua parameter pada nilai
default.
%
%   SETTRAIN(TRPARMS)
%      Menyusun semua parameter.
%
```

```
%   TRPARMS =
SETTRAIN(TRPARMS,'field1',value1,'field2',value2
,...)
%      Menentukan parameter spesifik
%         TRPARMS.field1 = nilai1;
%         TRPARMS.filed2 = nilai2;
%         etc.
%      jika niali = 'default', parameter-
parameter diset ke nilai default.
%
%
%   Kriteria henti(untuk semua algoritma)
%    maxiter   - Iterasi maksimum.
%    critmin   - Berhenti jika kriteria dibawah
nilai ini.
%    critterm  - Berhenti jika perubahan
kriteria dibawah nilai ini.
%    gradterm  - Berhenti jika elemen terbesar
pada gradien dibawah nilai ini.
%    paramterm - Berhenti jika parameter
terbesar berubah dibawah nilai ini.
%
%   Peluruhan bobot (semua algoritma dilatih
dengan Levenberg-Marquardt).
%    D         - Vektor baris yang mengandung
parametr peluruhan bobot.
%                Jika D mempunyai satu elemen,
digunakan peluruhan bobot skalar.
%                Jika D mempunyai dua elemen,
elemen pertama akan digunakan
%                sebagai peluruh bobot untuk
lap.tersembunyi-ke-output,
%                sementara elemen yang kedua
akan digunakan sebagai bobot
%                input-ke-lap.tersembunyi. Untuk
peluruh bobot individual,
%                D harus mengandung elemen
sebanyak bobot pada jaringan.
%
%   Parameter-parameter Levenberg-Marquardt
```

```
%    lambda    - Parameter Levenberg-Marquardt
awal .
%
%
%  Diadopsi dari : Magnus Norgaard, IAU/IMM,
dengan perubahan seperlunya.
%=================================================
============================

% >>>>>>>>>>>>>>>>>>>>   MENENTUKAN SEMUA
PARAMETER KE DEFAULT   <<<<<<<<<<<<<<<<<<<<
% Informasi level
trd.infolevel = 0;

% Nilai terminasi
trd.maxiter = 500;
trd.critmin = 0;
trd.critterm = 1e-7;
trd.gradterm = 1e-4;
trd.paramterm = 1e-3;

% Peluruhan bobot
trd.D = 1;

% Parameter Levenberg-Marquardt
trd.lambda = 1;

if nargin==0
  tr = trd;

% >>>>>>>>>>>>>>>>>>   MENUNUJUKAN PROPERTI-
PROPERTI   <<<<<<<<<<<<<<<<<<
elseif nargin==1,
  names = fieldnames(trparms);
  for idx=1:length(names),
    tmp = getfield(trparms,names{idx});
    if ischar(tmp),
      fprintf('%15s  = %s\n',names{idx},tmp);
```

```
    elseif (size(tmp,1)==1 | size(tmp,1)==1)
      if rem(tmp,1)==0,
        fprintf('%15s  = %d\n',names{idx},tmp);
      else
        fprintf('%15s  =
%4.3e\n',names{idx},tmp);
      end
    else
       fprintf('%15s  = [%dx%d
double]\n',names{idx},size(tmp,1),size(tmp,2));
    end
  end

% >>>>>>>>>>>>>>>>>>>>>>>   MENGESET PROPERTI
SPESIFIK   <<<<<<<<<<<<<<<<<<<<
elseif nargin>=2,
  tr = trparms;
  if rem(length(varargin),2),
    error('Harus dispesifikan sebuah angka genap
dari properti.');
  end
  for idx=1:2:length(varargin)

    % Mengecheck apakah objeknya sebuah string
    if ~isstr(varargin{idx})
      error('Nama properti harus dalam bentuk
string.');

    % Mengecek apakah objek dikenal

elseif(isempty(find(strcmp(lower(dnames),lower(v
arargin{idx})))))
       errstr = sprintf('%s ''%s''.','Nama
properti tidak dikenali ',varargin{idx});
       error(errstr);

    % Mengeset ke nilai default jika nilainya
diminta
```

```
elseif(strcmp(lower(varargin{idx+1}),'default'))
      if strcmp(lower(varargin{idx}),'d'),
        tr = setfield(tr,'D',getfield(trd,'D'));
      else

tr=setfield(tr,lower(varargin{idx}),getfield(trd
,lower(varargin{idx})));
      end

    % Mengeset objek ke nilai spesifik
    else
      if strcmp(lower(varargin{idx}),'d'),
        tr = setfield(tr,'D',varargin{idx+1});
      else
        tr =
setfield(tr,lower(varargin{idx}),varargin{idx+1}
);
      end
    end
  end
end
```

2.4. marq_rev.m

```
function
[W1,W2,PI_vector,y2]=marq_rev(NetDef,W1,W2,PHI,Y
,trparms)
%================================================
============================
%  MARQ
%  ----
%          Melatih dua lapis jaringan syaraf
tiruan dengan metode
%          Levenberg-Marquardt
%
%          Diberikan data input-output dari
sistem dan jaringan awal
```

```
%
[W1,W2,critvec,iteration,lambda]=marq(NetDef,W1,
W2,PHI,Y,trparms)
%          melatih jaringan dengan metode
Levenberg-Marquardt.
%
%          Fungsi aktivasi dapat berupa liner
atau tanh.
%          Arsitektur jaringan didefinisikan
oleh matrik 'NetDef' yang
%          mempunyai dua baris. Baris pertama
menunjukkan lapis
%          tersembunyi, dan baris yang kedua
menunjukkan lapis output.
%
%
%          Misal:    NetDef = ['LHHHH'
%                              'LL---']
%          (L = Linier, H = tanh)
%
%
%          Bias terdapat dalam kolom terakhir di
matrik bobot.
%
%
%  INPUT:
%  NetDef : Definisi jaringan .
%  W1     : Bobot input-ke-lapis tersembunyi.
Dimensi dari matriknya adalah
%           [(# unit tersembunyi)-kali-(input +
1)] (1 adalah bias).
%           Gunakan [] untuk inisialisasi acak.
%  W2     : Bobot lap. tersembunyi ke lapis
output. Dimensinya adalah
%           [(output)  *  (# unit tersembunyi +
1)].
%           Gunakan [] untuk inisialisasi acak.
%  PHI    : Vektor input. dim(PHI) = [(input)  *
(# data)].
```

```
%  Y      : Data output. dim(Y) = [(output)  *
(# data)].
%  trparms: Struktur data dengan parameter-
parameter yang berhubungan dengan
%           algoritma pelatihan (opsional).
Gunakan fungsi SETTRAIN jika
%           tidak ingin menggunakan nilai
default.
%
%  OUTPUT:
%  W1, W2   : Matrik bobot setelah pelatihan.
%  critvec:   Vector yang mengandung kriteria
yang dievaluasi pada setiap iterasi
%  iteration: # iterasi
%  lambda   : Nilai akhir dari lambda. Hanya
berguna jika retraining dikehendaki

%  Diadopsi dari: Magnus Norgaard, IAU/IMM,
Tecnical University of Denmark
%
================================================
=========================


%-----------------------------------------------
-----------------------------------
%--------------              INISIALISASI
JARINGAN                     -------------
%-----------------------------------------------
-----------------------------------
[outputs,N] = size(Y);                    % #
output dan # data
[inputs,N] = size(PHI);                   % # unit
tersembunyi
L_hidden = find(NetDef(1,:)=='L')';       % Letak
dari syaraf linier tersembunyi
H_hidden = find(NetDef(1,:)=='H')';       % Letak
dari syaraf tanh tersembunyi
```

```
L_output = find(NetDef(2,:)=='L')';      % Letak
dari syaraf linier output
H_output = find(NetDef(2,:)=='H')';      % Letak
dari syaraf tanh output
hidden = length(L_hidden)+length(H_hidden);
if isempty(W1) | isempty(W2),            %
Inisialisasi bobot jika dibutuhkan
  W1 = rand(hidden,inputs+1)-0.5;
  W2 = rand(outputs,hidden+1)-0.5;
end

y1       = [zeros(hidden,N);ones(1,N)]; % Output
lapisan tersembunyi
y2       = zeros(outputs,N);             % Output
jaringan
index = outputs*(hidden+1) + 1 + [0:hidden-
1]*(inputs+1); % Vektor bantu!
index2 = (0:N-1)*outputs;                % Vektor
bantu yang lain
iteration = 1;                           %
Penghitung variabel
dw       = 1;                            % Flag
yang menyatakan bahwa bobotnya adalah baru
PHI      = [PHI;ones(1,N)];              %
Augment PHI dengan sebuah baris yang nilainya 1
parameters1= hidden*(inputs+1);          % #
bobot input-ke-lap. tersembunyi
parameters2= outputs*(hidden+1);         % #
bobot lap. tersembunyi-ke-output
parameters = parameters1 + parameters2; % Total
# bobot
PSI      = zeros(parameters,outputs*N); %
Turunan dari setiap output w.r.t. setiap bobot
ones_h   = ones(hidden+1,1);             % Vektor
satuan
ones_i   = ones(inputs+1,1);             % Vektor
satuan yang lain
                                         %
Parameter vektor yang mengandung
```

```
                                                % semua
bobot
theta = [reshape(W2',parameters2,1) ;
reshape(W1',parameters1,1)];
theta_index = find(theta);                      % Index
ke bobot<>0
theta_red = theta(theta_index);                 %
Mengurangi vektor parameter
reduced  = length(theta_index);                 % #
parameter-parameter dalam theta_red
index3   = 1:(reduced+1):(reduced^2);           % Vektor
bantu yang lain
lambda_old = 0;
if nargin<6 | isempty(trparms)                  %
Parameter-parameter pelatihan default
  trparms = settrain;
  lambda  = trparms.lambda;
  D       = trparms.D;
else                                            % Nilai
yang ditentukan
  if ~isstruct(trparms),
     error('''trparms'' harus variabel
struktur.');
  end
  if ~isfield(trparms,'infolevel')
     trparms =
settrain(trparms,'infolevel','default');
  end
  if ~isfield(trparms,'maxiter')
     trparms =
settrain(trparms,'maxiter','default');
  end
  if ~isfield(trparms,'critmin')
     trparms =
settrain(trparms,'critmin','default');
  end
  if ~isfield(trparms,'critterm')
     trparms =
settrain(trparms,'critterm','default');
  end
```

```
  if ~isfield(trparms,'gradterm')
     trparms =
settrain(trparms,'gradterm','default');
  end
  if ~isfield(trparms,'paramterm')
     trparms =
settrain(trparms,'paramterm','default');
  end
  if ~isfield(trparms,'lambda')
     trparms =
settrain(trparms,'lambda','default');
  end
  lambda    = trparms.lambda;
  if ~isfield(trparms,'D')
     trparms = settrain(trparms,'D','default');
     D = trparms.D;
  else
    if length(trparms.D)==1,                %
Parameter peluruhan bobot scalar
      D = trparms.D(ones(1,reduced));
    elseif length(trparms.D)==2,            % Dua
bobot parameter peluruan
      D = trparms.D([ones(1,parameters2)
2*ones(1,parameters1)])';
      D = D(theta_index);
    elseif length(trparms.D)>2,             %
Peluruhan bobot individual
      D = trparms.D(:);
    end
  end
end
D = D(:);
critdif  = trparms.critterm+1;              %
Inisialisasi variabel untuk berhenti
gradmax  = trparms.gradterm+1;
paramdif = trparms.paramterm+1;
PI_vector = zeros(trparms.maxiter,1);       %
Vektor untuk menyimpan nilai kriteria
```

```
%-----------------------------------------------
-----------------------------------
%--------------                    PELATIHAN
JARINGAN                        -------------
%-----------------------------------------------
-----------------------------------
clc;
c=fix(clock);
fprintf('Pelatihan jaringan dimulai pada
%2i.%2i.%2i\n\n',c(4),c(5),c(6));


% >>>>>>>>>>>>>>>>>>>  MENGHITUNG OUTPUT
JARINGAN y2(theta)   <<<<<<<<<<<<<<<<<<<<
h1 = W1*PHI;
y1(H_hidden,:) = pmntanh(h1(H_hidden,:));
y1(L_hidden,:) = h1(L_hidden,:);

h2 = W2*y1;
y2(H_output,:) = pmntanh(h2(H_output,:));
y2(L_output,:) = h2(L_output,:);

E        = Y - y2;                          % Error
pelatihan
E_vector = E(:);                            %
Membentuk E dalam vektor yang panjang
SSE      = E_vector'*E_vector;              % Sum of
squared errors (SSE)
PI       =
(SSE+theta_red'*(D.*theta_red))/(2*N); % Indeks
performansi

% Itersai hingga kriteria henti tercapai
while (iteration<=trparms.maxiter &
PI>trparms.critmin & lambda<1e7 & ...
       (critdif>trparms.critterm |
gradmax>trparms.gradterm | ...
```

```
       paramdif>trparms.paramterm))

    if dw==1,
   % >>>>>>>>>>>>>>>>>>>>>>>>>   MENGHITUNG
MATRIK PSI   <<<<<<<<<<<<<<<<<<<<<<<<
   % (Turunan dari setiap output (y2) jaringan
terhadap setiap bobot)

    % ====== Element-elemen yang berhubungan
pada unit-unit output linier ========
    for i = L_output'
      index1 = (i-1) * (hidden + 1) + 1;

      % -- Bagian dari PSI yang berhubungan pada
bobot lap.tersembunyi-ke-output --
      PSI(index1:index1+hidden,index2+i) = y1;
      % ----------------------------------------
----------------------------

      % -- Bagian dari PSI yang berhubungan pada
bobot input-ke-lap. tersembunyi --
      for j = L_hidden',
        PSI(index(j):index(j)+inputs,index2+i) =
W2(i,j)*PHI;
      end

      for j = H_hidden',
        tmp = W2(i,j)*(1-y1(j,:).*y1(j,:));
        PSI(index(j):index(j)+inputs,index2+i) =
tmp(ones_i,:).*PHI;
      end
      % ----------------------------------------
----------------------------
    end

    % ========  Element-elemen yang berhubungan
pada unit-unit output tanh =======
    for i = H_output',
```

```
      index1 = (i-1) * (hidden + 1) + 1;

      % -- Bagian dari PSI yang berhubungan pada
bobot lap.tersembunyi-ke-output --
      tmp = 1 - y2(i,:).*y2(i,:);
      PSI(index1:index1+hidden,index2+i) =
y1.*tmp(ones_h,:);
      % ----------------------------------------
----------------------------

      % -- Bagian dari PSI yang berhubungan pada
bobot input-ke-lap. tersembunyi --
      for j = L_hidden',
        tmp = W2(i,j)*(1-y2(i,:).*y2(i,:));
        PSI(index(j):index(j)+inputs,index2+i) =
tmp(ones_i,:).*PHI;
      end

      for j = H_hidden',
        tmp  = W2(i,j)*(1-y1(j,:).*y1(j,:));
        tmp2 = (1-y2(i,:).*y2(i,:));
        PSI(index(j):index(j)+inputs,index2+i) =
tmp(ones_i,:)...

.*tmp2(ones_i,:).*PHI;
      end
      % ----------------------------------------
----------------------------
    end
    PSI_red = PSI(theta_index,:);

    % -- Gradient --
    G = PSI_red*E_vector-D.*theta_red;

    % -- Means square error bagian Hessian  --
    H = PSI_red*PSI_red';
    H(index3) = H(index3)'+D;
% Meletakkan matrik diagonal
    dw = 0;
```

```
  end


% >>>>>>>>>>>>>>>>>>>>>>>>>>>>        MENGHITUNG
h_k        <<<<<<<<<<<<<<<<<<<<<<<<<<<<
  % -- Hessian  --
  H(index3) = H(index3)'+(lambda-lambda_old);
% Meletakkan matrik diagonal

  % -- Search direction --
  h = H\G;
% Penyelesaian untuk search direction

  % -- Menghitung iterasi 'apriori'  --
  theta_red_new = theta_red + h;
% Update parameter vector
  theta(theta_index) = theta_red_new;

  % -- Meletakkan parameter-parameter kembali ke
matrik bobot --
  W1_new =
reshape(theta(parameters2+1:parameters),inputs+1
,hidden)';
  W2_new =
reshape(theta(1:parameters2),hidden+1,outputs)';


% >>>>>>>>>>>>>>>>>>>>   MENGHITUNG OUTPUT
JARINGAN  y2(theta+h)   <<<<<<<<<<<<<<<<<<<<
  h1 = W1_new*PHI;
  y1(H_hidden,:) = pmntanh(h1(H_hidden,:));
  y1(L_hidden,:) = h1(L_hidden,:);

  h2 = W2_new*y1;
  y2(H_output,:) = pmntanh(h2(H_output,:));
  y2(L_output,:) = h2(L_output,:);
```

```
  E_new        = Y - y2;                  %
Pelatihan error
  E_new_vector = E_new(:);                %
Menyusun kembali E dalam vektor yang panjang
  SSE_new  = E_new_vector'*E_new_vector; % Sum
of squared errors (SSE)
  PI_new   = (SSE_new +
theta_red_new'*(D.*theta_red_new))/(2*N); % PI


% >>>>>>>>>>>>>>>>>>>>>>>>>>>>>       UPDATE
lambda     <<<<<<<<<<<<<<<<<<<<<<<<<<<<<
  L = h'*G + h'*(h.*(D+lambda));
  lambda_old = lambda;

  % Kurangi lambda jika SSE telah 'dengan cukup'
turun
  if 2*N*(PI - PI_new) > (0.75*L),
    lambda = lambda/2;

  % Naikkan lambda jika SSE telah 'dengan cukup'
tumbuh
  elseif 2*N*(PI-PI_new) <= (0.25*L),
    lambda = 2*lambda;
  end

% >>>>>>>>>>>>>>>>>>>>>>       UPDATE UNTUK
ITERASI SELANJUTNYA        <<<<<<<<<<<<<<<<<<<<<
  % Update hanya jika kriteria telah berkurang
  if PI_new < PI,
    critdif  = PI-PI_new;
% Perbedaan kriteria
    gradmax  = max(abs(G))/N;
% Gradien maksimum
    paramdif = max(abs(theta_red_new -
theta_red)); % Turunan parameter maksimum
    W1 = W1_new;
    W2 = W2_new;
    theta_red = theta_red_new;
```

```
    E_vector = E_new_vector;
    PI = PI_new;
    dw = 1;
    lambda_old = 0;
    iteration = iteration + 1;
    PI_vector(iteration-1) = PI;
% Mengumpulkan PI dalam vector
    switch(trparms.infolevel)
% Cetak informasi on-line
       case 1
          fprintf('# %i   W=%4.3e  critdif=%3.2e
maxgrad=%3.2e  paramdif=%3.2e\n',...

iteration-1,PI,critdif,gradmax,paramdif);
       otherwise
          fprintf('iterasi ke %i   W =
%4.3e\r',iteration-1,PI);
    end
  end
end
%-------------------------------------------------
%--------------            AKHIR DARI PELATIHAN
JARINGAN             -------------
%-------------------------------------------------
iteration = iteration-1;
PI_vector = PI_vector(1:iteration);
c=fix(clock);
fprintf('\n\nPelatihan jaringan berakhir pada
%2i.%2i.%2i\n',c(4),c(5),c(6));
```

2.5. rmse.m

```
function [e]=r_m_s_e(y,yhat);

% Fungsi ini untuk menghitung root mean squared
error
% dari data hasil identifikasi

% y      : data dari output proses
% yhat   : data dari output model
```

```
l1=length(y);
l2=length(yhat);

if l1==l2
   e=sqrt(sum((y-yhat).^2)/l1);
else
error ('Dimensi data tidak sama')
end
```

2.6. vaf.m

```
function v = vaf(y,ym)
%=================================================
% Vaf adalah Variance accounted for (Penghitung
variansi).
% V = VAF(Y,Ym) mencari presentasi terukur
antara dua sinyal. Ketika
% digunakan untuk memvalidasi model, Y adalah
data terukur dan Ym adalah
% output model. Ketika Y dan Ym adalah matrik,
VAF menghitung seluruh kolom
% yang sama.
% Diadopsi dari: Robert Babuska, 1996
%=================================================

v=diag(100*(1-(cov(y-ym)/cov(y))));
```

2.7. pmntanh.m

```
function t=pmntanh(x)
% PMNTANH
% -------
% Fungsi tangent hiperbolik di gunakan dalam
jaringan syaraf tiruan sebagai
% pengganti dari tangent yang disediakan oleh
MATLAB

t=1-2./(exp(2*x)+1);
```

3. Program pemodelan pengendali
   Pada umunya program pemodelan pengendali dengan pemodelan plant hampir sama, hanya yang membedakan adalah pada tahap training dan validasi. Berikut listing training.m dan validasi.m pada program pemodelan controller pada matlab;

3.1. training.m

```
% ------------------------ PEMODELAN KONTROLLER
JST ------------------------
% Program Jaringan Syaraf Tiruan untuk
mendapatkan Model Kontroller JST

clear all
close all
clc

% ------------------------------- TRAINING -----
--------------------------
% Load Data Input Output

load data
load data_scal

y11 = y1s(1:282); %sebagai data input
u11 = u1s(1:282); %sebagai data output 1
u12 = u2s(1:282); %sebagai data output 2
u13 = u3s(1:282); %sebagai data output 3
u14 = u4s(1:282); %sebagai data output 4
u15 = u5s(1:282); %sebagai data output 5

% Menampilkan Grafik Data Input Output Training

figure (1)
plot(y11);grid
  title('Input
Daya','FontSize',8,'FontWeight','bold')
 legend('Daya');
```

```
 xlabel('data','FontSize',8)
 ylabel('Daya (KW)','FontSize',8)

figure(2)
subplot(2,1,1);plot(u11);grid;
 title('Output Batang Kendali
1','FontSize',8,'FontWeight','bold')
legend('Posisi Batang Kendali 1');
xlabel('data','FontSize',8)
ylabel('Posisi Batang Kendali','FontSize',8)
subplot(2,1,2);plot(u12);grid;
 title('Output Batang Kendali
2','FontSize',8,'FontWeight','bold')
legend('Posisi Batang Kendali 2');
xlabel('data','FontSize',8)
ylabel('Posisi Batang Kendali','FontSize',8)

figure(3)
subplot(2,1,1);plot(u13);grid;
 title('Output Batang Kendali
3','FontSize',8,'FontWeight','bold')
legend('Posisi Batang Kendali 3');
xlabel('data','FontSize',8)
ylabel('Posisi Batang Kendali','FontSize',8)
subplot(2,1,2);plot(u14);grid;
 title('Output Batang Kendali
4','FontSize',8,'FontWeight','bold')
legend('Posisi Batang Kendali 4');
xlabel('data','FontSize',8)
ylabel('Posisi Batang Kendali','FontSize',8)

figure(4)
  subplot (2,1,1);plot(u15);grid;
    title('Output Batang Kendali
5','FontSize',8,'FontWeight','bold')
 legend('Posisi Batang Kendali 5');
 xlabel('data','FontSize',8)
 ylabel('Posisi Batang Kendali','FontSize',8)
```

```
strvcat('Figure 1, 2, 3 dan 4 adalah grafik data
input output Training',...
   'Tekan sembarang tombol untuk melanjutkan!')
pause;

% Konstruksi dari Matrik Input
hist = [1 1 1 1 1 1];
[n_rows,n_col] = size(y11);

% Membuat Matrik Regressor
data_latih = zeros(n_rows-1,sum(hist));

for i = 1:hist(1),
   data_latih(:,i) = [zeros(-
1+i,1);y11(1:n_rows-1+1-i)];
end

for j = 1:hist(2),
   data_latih(:,sum(hist(1))+j) = [zeros(-
1+j,1);u11(1:n_rows-1+1-j)];
end

for k = 1:hist(3),
   data_latih(:,sum(hist(1:2))+k) = [zeros(-
1+k,1);u12(1:n_rows-1+1-k)];
end

for l = 1:hist(4),
   data_latih(:,sum(hist(1:3))+l) = [zeros(-
1+l,1);u13(1:n_rows-1+1-l)];
end

for m = 1:hist(5),
   data_latih(:,sum(hist(1:4))+m) = [zeros(-
1+m,1);u14(1:n_rows-1+1-m)];
end

for n = 1:hist(6),
```

```
   data_latih(:,sum(hist(1:5))+n) = [zeros(-
1+n,1);u15(1:n_rows-1+1-n)];
end

PHI = data_latih'; % matrik PHI merupakan
regressor bagi model JST

U = zeros(n_rows-1,5);
U(:,1) = u11(2:end);
U(:,2) = u12(2:end);
U(:,3) = u13(2:end);
U(:,4) = u14(2:end);
U(:,5) = u15(2:end);
U_target = U';

NN = [hist(1) hist(2) hist(3) hist(4) hist(5)
hist(6) 1];

% Konstruksi dari Struktur Jaringan
NetDefi = ['LLLLLLLLLLLLLLLLLLLLLLLLL';'LLLLL--
-------------------'];
trparms = settrain;
[W1i,W2i,PI_vec,uhat] =
marq_rev(NetDefi,[],[],PHI,U_target,trparms);

% Root Mean Square Error
error_training_1 = rmse(u11(1:281)',uhat(1,:))
error_training_2 = rmse(u12(1:281)',uhat(2,:))
error_training_3 = rmse(u13(1:281)',uhat(3,:))
error_training_4 = rmse(u14(1:281)',uhat(4,:))
error_training_5 = rmse(u15(1:281)',uhat(5,:))
display('Nilai di atas adalah RMSE untuk
Training');

% Variance Accounted For
VAF_training_1 = vaf(u11(1:281),uhat(1,:)')
VAF_training_2 = vaf(u12(1:281),uhat(2,:)')
VAF_training_3 = vaf(u13(1:281),uhat(3,:)')
```

```
VAF_training_4 = vaf(u14(1:281),uhat(4,:)')
VAF_training_5 = vaf(u15(1:281),uhat(5,:)')
display('Nilai di atas VAF untuk Training');

%Prediksi Error
error1=u11(1,:)-uhat(1,:);
error2=u12(1,:)-uhat(2,:);
error3=u13(1,:)-uhat(3,:);
error4=u14(1,:)-uhat(4,:);
error5=u15(1,:)-uhat(5,:);

%Descaling
uhat1=uhat(1,:)*840;
uhat2=uhat(2,:)*840;
uhat3=uhat(2,:)*839;
uhat4=uhat(2,:)*839;
uhat5=uhat(2,:)*840;

% Menampilkan Grafik Hasil Pelatihan

figure(5)
subplot(2,1,1);
plot(u1(1:282));
hold on
plot(uhat1,'r--');
grid
 title('Output Batang Kendali
1','FontWeight','bold')
legend('Solid : Output Proses','Dash : Output
Model');
xlabel('data','FontSize',8)
ylabel('Posisi Batang Kendali 1','FontSize',8)

subplot(2,1,2);
plot(error1);
grid
 title('Prediksi error(u-
uhat)','FontWeight','bold');
xlabel('data');
```

```
ylabel('error');

figure(6)
subplot(2,1,1);
plot(u2(1:282));
hold on
plot(uhat2,'r--');
grid
 title('Output Batang Kendali
2','FontWeight','bold')
legend('Solid : Output Proses','Dash : Output
Model');
xlabel('data','FontSize',8)
ylabel('Posisi Batang Kendali 2','FontSize',8)

subplot(2,1,2);
plot(error2);
grid
 title('Prediksi error(u-
uhat)','FontWeight','bold');
xlabel('data');
ylabel('error');

figure(7)
subplot(2,1,1);
plot(u3(1:282));
hold on
plot(uhat3,'r--');
grid
 title('Output Batang Kendali
3','FontWeight','bold')
legend('Solid : Output Proses','Dash : Output
Model');
xlabel('data','FontSize',8)
ylabel('Posisi Batang Kendali 3','FontSize',8)

subplot(2,1,2);
plot(error3);
grid
```

```
 title('Prediksi error(u-
uhat)','FontWeight','bold');
xlabel('data');
ylabel('error');

figure(8)
subplot(2,1,1);
plot(u4(1:282));
hold on
plot(uhat4,'r--');
grid
 title('Output Batang Kendali
4','FontWeight','bold')
legend('Solid : Output Proses','Dash : Output
Model');
xlabel('data','FontSize',8)
ylabel('Posisi Batang Kendali 4','FontSize',8)

subplot(2,1,2);
plot(error4);
grid
 title('Prediksi error(u-
uhat)','FontWeight','bold');
xlabel('data');
ylabel('error');

figure(9)
subplot(2,1,1);
plot(u5(1:282));
hold on
plot(uhat5,'r--');
grid
 title('Output Batang Kendali
5','FontWeight','bold')
legend('Solid : Output Proses','Dash : Output
Model');
xlabel('data','FontSize',8)
ylabel('Posisi Batang Kendali 5','FontSize',8)
```

```
subplot(2,1,2);
plot(error5);
grid
 title('Prediksi error(u-
uhat)','FontWeight','bold');
xlabel('data');
ylabel('error');

% Menyimpan Data
save inverse NN W1i W2i NetDefi hist
```

3.2. validasi.m

```
% ------------------------ PEMODELAN KONTROLLER
JST ----------------------
% Program Jaringan Syaraf Tiruan untuk
mendapatkan Model Kontroller JST

clear all
close all
clc

% ------------------------------ VALIDASI -----
--------------------------
% Load Data Input Output

load data
load data_scal
load inverse

y1_uji = y1s (283:326); %sebagai data input
u1_uji = u1s (283:326); %sebagai data output 1
u2_uji = u2s (283:326); %sebagai data output 2
u3_uji = u3s (283:326); %sebagai data output 3
u4_uji = u4s (283:326); %sebagai data output 4
u5_uji = u5s (283:326); %sebagai data output 5
```

```
% Menampilkan Grafik Data Input Output Validasi
figure (1)
plot(y1_uji);grid
  title('Input
Daya','FontSize',8,'FontWeight','bold')
legend('Daya');
xlabel('data','FontSize',8)
ylabel('Daya (KW)','FontSize',8)

figure(2)
subplot (2,1,1);plot(u1_uji);grid;
 title('Output Batang Kendali
1','FontSize',8,'FontWeight','bold')
legend('Posisi Batang Kendali 1');
xlabel('data','FontSize',8)
ylabel('Posisi Batang Kendali','FontSize',8)
subplot(2,1,2);plot(u2_uji);grid;
 title('Output Batang Kendali
2','FontSize',8,'FontWeight','bold')
legend('Posisi Batang Kendali 2');
xlabel('data','FontSize',8)
ylabel('Posisi Batang Kendali','FontSize',8)

figure(3)
subplot (2,1,1);plot(u3_uji);grid;
 title('Output Batang Kendali
3','FontSize',8,'FontWeight','bold')
legend('Posisi Batang Kendali 3');
xlabel('data','FontSize',8)
ylabel('Posisi Batang Kendali','FontSize',8)
subplot(2,1,2);plot(u4_uji);grid;
 title('Output Batang Kendali
4','FontSize',8,'FontWeight','bold')
legend('Posisi Batang Kendali 4');
xlabel('data','FontSize',8)
ylabel('Posisi Batang Kendali','FontSize',8)

figure(4)
  subplot (2,1,1);plot(u5_uji);grid;
```

```
 title('Output Batang Kendali
5','FontSize',8,'FontWeight','bold')
legend('Posisi Batang Kendali 5');
xlabel('data','FontSize',8)
ylabel('Posisi Batang Kendali','FontSize',8)

 strvcat('Figure 1, 2, 3 dan 4 adalah grafik
data input output validasi',...
    'Tekan sembarang tombol untuk melanjutkan!')
 pause;

% Membuat Matrik Regressor Validasi
[n_rows,n_col] = size(y1_uji);

data_uji = zeros(n_rows-1,sum(hist));

for i = 1:hist(1),
   data_uji(:,i) = [zeros(-
1+i,1);y1_uji(1:n_rows-1+1-i)];
end

for j = 1:hist(2),
   data_uji(:,sum(hist(1))+j) = [zeros(-
1+j,1);u1_uji(1:n_rows-1+1-j)];
end

for k = 1:hist(3),
   data_uji(:,sum(hist(1:2))+k) = [zeros(-
1+k,1);u2_uji(1:n_rows-1+1-k)];
end

for l = 1:hist(4),
   data_uji(:,sum(hist(1:3))+l) = [zeros(-
1+l,1);u3_uji(1:n_rows-1+1-l)];
end

for m = 1:hist(5),
```

```
   data_uji(:,sum(hist(1:4))+m) = [zeros(-
1+m,1);u4_uji(1:n_rows-1+1-m)];
end

for n = 1:hist(6),
   data_uji(:,sum(hist(1:5))+n) = [zeros(-
1+n,1);u5_uji(1:n_rows-1+1-n)];
end

PHI_uji = data_uji';
PHI_uji_bias =
[PHI_uji;ones(1,size(PHI_uji,2))];

U_uji = zeros(n_rows-1,2);
U_uji(:,1) = u1_uji(2:end);
U_uji(:,2) = u2_uji(2:end);
U_uji(:,3) = u3_uji(2:end);
U_uji(:,4) = u4_uji(2:end);
U_uji(:,5) = u5_uji(2:end);
U_target_uji = U_uji';

L_hiddeni = find(NetDefi(1,:)=='L')';     %
Location of linear hidden neurons
H_hiddeni = find(NetDefi(1,:)=='H')';     %
Location of tanh hidden neurons
L_outputi = find(NetDefi(2,:)=='L')';     %
Location of linear output neurons
H_outputi = find(NetDefi(2,:)=='H')';     %
Location of tanh output neurons
hiddeni = length(L_hiddeni)+length(H_hiddeni);

u1i       =
[zeros(hiddeni,size(data_uji,1));ones(1,size(dat
a_uji,1))]; % Hidden layer outputs
u2i       =
zeros(size(U_target_uji,1),size(data_uji,1));
% Network output
```

```
h1i = W1i*PHI_uji_bias;
u1i(H_hiddeni,:) = pmntanh(h1i(H_hiddeni,:));
u1i(L_hiddeni,:) = h1i(L_hiddeni,:);

h2i = W2i*u1i;
u2i(H_outputi,:) = pmntanh(h2i(H_outputi,:));
u2i(L_outputi,:) = h2i(L_outputi,:);
uhat_uji=u2i;

% Root Mean Square Error
error_uji_1 = rmse(u1_uji(1:43)',uhat_uji(1,:))
error_uji_2 = rmse(u2_uji(1:43)',uhat_uji(2,:))
error_uji_3 = rmse(u3_uji(1:43)',uhat_uji(3,:))
error_uji_4 = rmse(u4_uji(1:43)',uhat_uji(4,:))
error_uji_5 = rmse(u5_uji(1:43)',uhat_uji(5,:))
display('Nilai di atas adalah RMSE untuk
pelatihan');

% Variance Accounted For
VAF_uji_1 = vaf(u1_uji(1:43),uhat_uji(1,:)')
VAF_uji_2 = vaf(u2_uji(1:43),uhat_uji(2,:)')
VAF_uji_3 = vaf(u3_uji(1:43),uhat_uji(3,:)')
VAF_uji_4 = vaf(u4_uji(1:43),uhat_uji(4,:)')
VAF_uji_5 = vaf(u5_uji(1:43),uhat_uji(5,:)')
display('Nilai di atas VAF untuk training');

%Prediksi Error
error1=u1_uji(1,:)-uhat_uji(1,:);
error2=u2_uji(1,:)-uhat_uji(2,:);
error3=u3_uji(1,:)-uhat_uji(3,:);
error4=u4_uji(1,:)-uhat_uji(4,:);
error5=u5_uji(1,:)-uhat_uji(5,:);

%Descaling
uhat1_uji=uhat_uji(1,:)*840;
uhat2_uji=uhat_uji(2,:)*840;
uhat3_uji=uhat_uji(3,:)*839;
uhat4_uji=uhat_uji(4,:)*850;
uhat5_uji=uhat_uji(5,:)*840;
```

```
% Menampilkan Grafik Hasil Pelatihan
figure(5)
subplot(2,1,1);
plot(u1(283:326));
hold on
plot(uhat1_uji,'r--');
grid
 title('Output Batang Kendali
1','FontWeight','bold');
legend('Solid : Output Proses','Dash : Output
Model');
xlabel('data');
ylabel('Posisi Batang Kendali');

subplot(2,1,2);
plot(error1);
grid
 title('Prediksi error(u-
uhat)','FontWeight','bold');
xlabel('data');
ylabel('error');

figure(6)
subplot(2,1,1);
plot(u2(283:326));
hold on
plot(uhat2_uji,'r--');
grid
 title('Output Batang Kendali
2','FontWeight','bold');
legend('Solid : Output Proses','Dash : Output
Model');
xlabel('data');
ylabel('Posisi Batang Kendali');

subplot(2,1,2);
plot(error2);
grid
```

```
 title('Prediksi error(u-
uhat)','FontWeight','bold');
xlabel('data');
ylabel('error');

figure(7)
subplot(2,1,1);
plot(u3(283:326));
hold on
plot(uhat3_uji,'r--');
grid
 title('Output Batang Kendali
3','FontWeight','bold');
legend('Solid : Output Proses','Dash : Output
Model');
xlabel('data');
ylabel('Posisi Batang Kendali');

subplot(2,1,2);
plot(error1);
grid
 title('Prediksi error(u-
uhat)','FontWeight','bold');
xlabel('data');
ylabel('error');

figure(8)
subplot(2,1,1);
plot(u4(283:326));
hold on
plot(uhat4_uji,'r--');
grid
 title('Output Batang Kendali
4','FontWeight','bold');
legend('Solid : Output Proses','Dash : Output
Model');
xlabel('data');
ylabel('Posisi Batang Kendali');
```

```
subplot(2,1,2);
plot(error2);
grid
 title('Prediksi error(u-
uhat)','FontWeight','bold');
xlabel('data');
ylabel('error');

figure(9)
subplot(2,1,1);
plot(u5(283:326));
hold on
plot(uhat5_uji,'r--');
grid
 title('Output Batang Kendali
5','FontWeight','bold');
legend('Solid : Output Proses','Dash : Output
Model');
xlabel('data');
ylabel('Posisi Batang Kendali');

subplot(2,1,2);
plot(error2);
grid
 title('Prediksi error(u-
uhat)','FontWeight','bold');
xlabel('data');
ylabel('error');
```

4. Program simulasi DIC

   Pada program inilah bagaimana kita mendapatkan simulasi pengendali DIC. Dimana pada program ini terdapat 2 program utama yaitu **invinit.m** sebagai inisialisasi pengendali DIC dan **invicon.m** sebagai program yang akan menjalankan simulasi. Namun, terdapat beberapa program yang mendukung jalanya program simulasi ini yaitu **progress.m** sebagai display jalannya program, **shift.m** untuk mendapatkan vector selanjutnya dari data, **siggener.m** sebagai sinyal referensi dan **pmntanh.m**.

4.1. invinit.m

```
% ------------------------------>  INVINIT.M  <-
-----------------------------

% ----------      Switches        -----------
regty      ='dic';           % Controller type
(dic, pid, none)
refty      ='myref';       % Reference signal
(siggener/<var. name>)
simul      ='nnet';       % Control object spec.
(simulink/matlab/nnet)


% ------    General Initializations  -------
Ts = 60;                    % Sampling period (in
seconds)
samples = 1000 ;              % Number of
samples to be simulated
u_0_1 = 0;                     % Initial control
input
u_0_2 = 0;
u_0_3 = 0;
u_0_4 = 0;
u_0_5 = 0;
y_0 = 0;                      % Initial output
ulim_min = -Inf;              % Minimum control
input
```

```
ulim_max = Inf;                % Maximum control
input
c = 1;                         % gain proses


% --  System to be Controlled (SIMULINK) --
% integrator= 'ode45';         % Name of dif.
eq. solver (f. ex. ode45 or ode15s)
% sim_model = 'spm1';          % Name of
SIMULINK model


% ---  System to be Controlled (MATLAB)  --
% mat_model = 'springm';       % Name of MATLAB
model
% model_out = 'smout';         % Output equation
(function of the states)
% x0        = [0;0];           % Initial states


% ----- Inverse Network Specification ------
% The file must contain the variables:
% NN, NetDefi, W1i, W2i
% (i.e. regressor structure, architecture
definition, and weight matrices)
nninv = 'inverse';             % Name of file


% ----- Forward Network Specification ------
% Only used if simul='nnet' (no Simulink or
Matlab model available)
% The file must contain: NN, NetDeff, W1f, W2f
nnforw = 'forward';            % Name of file


% ------------ Reference Filter ---------------
% Am = [1 -0.7];               % Filter
denominator
```

```
% Bm = [0.3];                    % Filter
numerator
 Am=1;Bm=1;


% ------------ Reference signal ------------
dc      = 1000;                  % DC-level
sq_amp  = [-40];                  % Amplitude of
square signals (row vector)
sq_freq = [10];                 % Frequency of
square signals (column vector)
sin_amp = [0];                 % Amplitude of sine
signals  (row vector)
sin_freq= [0]';                % Frequency of sine
signals   (column vector)
Nvar  = 0.009;                    % Variance of
white noise signal
G = 0.75;                       % Gain factor
C = 0.05;                      %Corecction factor
D = 1;                      %Discrit Conversion

myref=[0*ones(100,1);2000*ones(400,1)];

% --------  Linear Controller Parameters  ------
---
% Kp=1;                            % PID parameters
% Ti=0;                          % PID
% alf=0;                         % PID
% Wi=0;                          % PID (1/Ti)


% ------- Specify data vectors to plot --------
% plot_a and plot_b must be cell structures
containing the vector names in strings
plot_a = {'ref_data','y_data'};
plot_b = {'u_data_1'};
plot_c = {'u_data_2'};
plot_d = {'u_data_3'};
```

```
plot_e = {'u_data_4'};
plot_f = {'u_data_5'};
```

4.2. invcon.m

```
% -----    INVCON     -----
%
%  Program for simulating direct inverse control
of nonlinear processes.
%
%  All design parameters must be defined in the
file 'invinit.m'
%
%  Programmed by Magnus Norgaard IAU, Technical
University of Denmark.
%  LastEditDate: Jan. 23, 2000

%-----------------------------------------------
-----------------------------------
%-------------------         >>>
INITIALIZATIONS  <<<        ---------------------
-
%-----------------------------------------------
-----------------------------------

%>>>>>>>>>>>>>>>>>>>>>>      READ VARIABLES FROM
FILE       <<<<<<<<<<<<<<<<<<<<<<<

clear plot_a plot_b
invinit
eval(['load ' nninv]);                    % Load
inverse neural model


% >>>>>>>>>>>>>>>>>>>>>>>   DETERMINE REGRESSOR
STRUCTURE   <<<<<<<<<<<<<<<<<<<<<<
na         = NN(1);          % # jumlah y' lampau
yang digunakan dalam TDL
```

```
nb1        = NN(2);          % # jumlah u' lampau
yang digunakan dalam TDL
nb2        = NN(3);
nb3        = NN(4);
nb4        = NN(5);
nb5        = NN(6);
nb         =(nb1+nb2+nb3+nb4+nb5);
nk         = NN(7);              % Waktu tunda
dalam sistem
nab        = na+sum(nb1:nb5);          % Jumlah
input masing-masing jaringan
outputs    = 1;              % # output
inputs     = nab-1;          % # input
phi = zeros(inputs,1);      % Inisialisasi vektor
regressor

% >>>>>>>>>>>>>>>>>   DETERMINE STRUCTURE OF
FORWARD MODEL    <<<<<<<<<<<<<<<<<<
if strcmp(simul,'nnet'),
  eval(['load ' nnforw]);                % Load
forward neural model of system
  hiddenf    = length(NetDeff(1,:));       %
Number of hidden neurons
  L_hiddenf = find(NetDeff(1,:)=='L')';   %
Location of linear hidden neurons
  H_hiddenf = find(NetDeff(1,:)=='H')';   %
Location of tanh hidden neurons
  L_outputf = find(NetDeff(2,:)=='L')';   %
Location of linear output neurons
  H_outputf = find(NetDeff(2,:)=='H')';   %
Location of tanh output neurons
  y1f        = [zeros(hiddenf,1);1];       %
Hidden layer outputs

end

% >>>>>>>>>>>>>>>>>>   DETERMINE STRUCTURE OF
INVERSE MODEL    <<<<<<<<<<<<<<<<<<
```

```
hiddeni   = length(NetDefi(1,:));         %
Number of hidden neurons
L_hiddeni = find(NetDefi(1,:)=='L')';     %
Location of linear hidden neurons
H_hiddeni = find(NetDefi(1,:)=='H')';     %
Location of tanh hidden neurons
L_outputi_1 = find(NetDefi(2,1)=='L')';     %
Location of linear output neurons
H_outputi_1 = find(NetDefi(2,1)=='H')';     %
Location of tanh output neurons
L_outputi_2 = find(NetDefi(2,2)=='L')';     %
Location of linear output neurons
H_outputi_2 = find(NetDefi(2,2)=='H')';     %
Location of tanh output neurons
L_outputi_3 = find(NetDefi(2,3)=='L')';     %
Location of linear output neurons
H_outputi_3 = find(NetDefi(2,3)=='H')';     %
Location of tanh output neurons
L_outputi_4 = find(NetDefi(2,4)=='L')';     %
Location of linear output neurons
H_outputi_4 = find(NetDefi(2,4)=='H')';     %
Location of tanh output neurons
L_outputi_5 = find(NetDefi(2,5)=='L')';     %
Location of linear output neurons
H_outputi_5 = find(NetDefi(2,5)=='H')';     %
Location of tanh output neurons
y1i       = [zeros(hiddeni,1);1];         %
Hidden layer outputs


%>>>>>>>>>>>>>>>>>>>>>>>>        INITIALIZE
VARIABLES        <<<<<<<<<<<<<<<<<<<<<<<
% Determine length of polynomials
nam = length(Am);
nbm = length(Bm);



% Initialization of past signals
```

```
maxlength = 5;                                  % MIGHT
BE NECESSARY TO INCREASE maxlength
ref_old    = repmat(y_0,maxlength,1);     % FOR
HIGH-ORDER SYSTEMS!!!
y_old      = repmat(y_0,maxlength,1);
u_old_1      = repmat(u_0_1,maxlength,1);
u_old_2      = repmat(u_0_2,maxlength,1);
u_old_3      = repmat(u_0_3,maxlength,1);
u_old_4      = repmat(u_0_4,maxlength,1);
u_old_5      = repmat(u_0_5,maxlength,1);

% Miscellanous initializations
t     = -Ts;
u_1     = u_0_1;
u_2     = u_0_2;
u_3     = u_0_3;
u_4     = u_0_4;
u_5     = u_0_5;
y     = y_0;
% yhat = y_0;

% Initialization of Simulink system
% if strcmp(simul,'simulink')
%   simoptions =
simset('Solver',integrator,'MaxRows',0); % Set
integrator
%   eval(['[sizes,x0] = ' sim_model
'([],[],[],0);']);    % Get initial states
% end


%>>>>>>>>>>>>>>>>>>    CALCULATE REFERENCE SIGNAL
& FILTER IT     <<<<<<<<<<<<<<<<<<
if strcmp(refty,'siggener'),
  ref = zeros(samples+1,1);
  for i = 1:samples+1,
    ref(i) = siggener(Ts*(i-
1),sq_amp,sq_freq,sin_amp,sin_freq,dc,sqrt(Nvar)
);
```

```
    ref_old  = shift(ref_old,ref(i));
  end
elseif strcmp(refty,'none'),
  ref = zeros(samples+1,1);
else
  eval(['ref = ' refty ';']);
  ref=ref(:);
  i=length(ref);
  if i>samples+1,
    ref=ref(1:samples+1);
  else
    ref=[ref;ref(i)*ones(samples+1-i,1)];
  end
end
ref=filter(Bm,Am,ref);

% Initialization of data vectors

u_data_1    = zeros(samples,1);
u_data_2    = zeros(samples,1);
u_data_3    = zeros(samples,1);
u_data_4    = zeros(samples,1);
u_data_5    = zeros(samples,1);
y_data      = zeros(samples,1);
t_data      = zeros(samples,1);
ref_data    = ref(1:samples);
fighandle=progress;

%-----------------------------------------------
-------------------------------
%-------------------          >>>   MAIN LOOP
<<<           ------------------
%-----------------------------------------------
-------------------------------
for i=1:samples,
  t = t + Ts;
```

```
% %>>>>>>>>>>>>>>>>>>>>  READ OUTPUT FROM THE
PHYSICAL SYSTEM   <<<<<<<<<<<<<<<<<<
%   if strcmp(simul,'simulink')
%     utmp=[t-Ts,u_old(1);t,u_old(1)];
%     simoptions.InitialState=x0;
%     [time,x0,y] = sim(sim_model,[t-Ts
t],simoptions,utmp);
%     x0 = x0(size(x0,1),:)';
%     y  = y(size(y,1),:)';
%   elseif strcmp(simul,'matlab')
%     ugl = u_old(1);
%     [time,x] = ode45(mat_model,[t-Ts t],x0);
%     x0 = x(length(time),:)';
%     eval(['y = ' model_out '(x0);']);
%   if
      strcmp(simul,'nnet')
    phi =
[y_old(1:na);u_old_1(1:nb1);u_old_2(1:nb2);u_old
_3(1:nb3);u_old_4(1:nb4);u_old_5(1:nb5);1];
    h1f = W1f*phi;
    y1f(H_hiddenf)  = pmntanh(h1f(H_hiddenf));
    y1f(L_hiddenf)  = h1f(L_hiddenf);
    h2f = W2f*y1f;
    y(H_outputf) = pmntanh(h2f(H_outputf));
    y(L_outputf) = h2f(L_outputf)*c;
%   end


%>>>>>>>>>>>>>>>>>>>>>>     DETERMINE CONTROL
SIGNAL      <<<<<<<<<<<<<<<<<<<<<<
  e = ref(i) - y;

  % Control using the inverse model

  if strcmp(regty,'dic'),
    phii = [ref(i+1);y;y_old(1:na-
1);u_old_1(1:nb4-1);u_old_2(1:nb3-
1);u_old_3(1:nb2-1);u_old_4(1:nb1-
1);u_old_5(1:nb-1);1];
```

```
    h1  = W1i*phii;
    y1i(H_hiddeni) = pmntanh(h1(H_hiddeni));
    y1i(L_hiddeni) = h1(L_hiddeni);
    h2 = W2i*y1i;
     u_1(H_outputi_1)  =
pmntanh(h2(H_outputi_1));
    u_1(L_outputi_1)  = h2(L_outputi_1)*G;
    u_2(H_outputi_2)  =
pmntanh(h2(H_outputi_2));
    u_2(L_outputi_2)  = h2(L_outputi_2)*G;
    u_3(H_outputi_3)  =
pmntanh(h2(H_outputi_3));
    u_3(L_outputi_3)  = h2(L_outputi_3)*G;
    u_4(H_outputi_4)  =
pmntanh(h2(H_outputi_4));
    u_4(L_outputi_4)  = h2(L_outputi_4)*G;
    u_5(H_outputi_5)  =
pmntanh(h2(H_outputi_5));
    u_5(L_outputi_5)  = h2(L_outputi_5)*G;

  else
     u = ref(i);
  end

%   % Make sure control inputs is within limits
  if u_1>ulim_max,
     u_1=ulim_max;
     u_2>ulim_max;
     u_2=ulim_max;
     u_3>ulim_max;
     u_3=ulim_max;
     u_4>ulim_max;
     u_4=ulim_max;
     u_5>ulim_max;
     u_5=ulim_max;
  elseif u_1<ulim_min
     u_1=ulim_min;
     u_2<ulim_min
     u_2=ulim_min;
```

```
    u_3<ulim_min
    u_3=ulim_min;
    u_4<ulim_min
    u_4=ulim_min;
    u_5<ulim_min
    u_5=ulim_min;

  end


  %>>>>> COPY DATA INTO THE DATA VECTORS <<<<<
  u_data_1(i,:)        = u_1;
  u_data_2(i,:)        = u_2;
  u_data_3(i,:)        = u_3;
  u_data_4(i,:)        = u_4;
  u_data_5(i,:)        = u_5;
  y_data(i,:)        = y;
  t_data(i,:)        = t;


%>>>>>>      TIME OPDATES   <<<<<<<<<<
  y_old      = shift(y_old,y);
  u_old_1    = shift(u_old_1,u_1);
  u_old_2    = shift(u_old_2,u_2);
  u_old_3    = shift(u_old_3,u_2);
  u_old_4    = shift(u_old_4,u_2);
  u_old_5    = shift(u_old_5,u_2);
  ref_old    = shift(ref_old,ref(i));


%>>>>> PRINT %-AGE OF SIMULATION COMPLETED
<<<<<<<
  progress(fighandle,floor(100*i/samples));
end
%----------------         >>>   END OF MAIN LOOP
<<<        ----------------

%>>>>    DRAW PLOTS   <<<<
```

```
figure(gcf);clf
set(gcf,'DefaultTextInterpreter','none');
% Plot A
  if(exist('plot_a')==1),
   a_plots=length(plot_a);             % Number
of plots in plot A
   plmat = zeros(samples,a_plots);     % Collect
vectors in plmat
   for nn = 1:a_plots,
      plmat(:,nn) = eval(plot_a{nn});
   end
   subplot(2,1,1);
   plot([0:samples-1],plmat);            % Plot
plmat
   xlabel('Samples');
   set(gca,'Xlim',[0 samples-1]);        % Set x-
axis
   if regty(1)=='d',
     title('Direct inverse control');
   elseif regty(1)=='p',
     title('Constant gain PID controller');
   else
     title('Open-loop simulation');
   end
   grid on
   legend(plot_a{:});
  end

 % Plot B
  if(exist('plot_b')==1),
    b_plots=length(plot_b);               %
Number of plots in plot B
    plmat = zeros(samples,b_plots);       %
Collect vectors in plmat
    for nn = 1:b_plots,
      plmat(:,nn) = eval(plot_b{nn});
    end
   subplot(2,1,2);
```

```
   plot([0:samples-1],plmat);            % Plot
plmat
   xlabel('Waktu (menit)');
   set(gca,'Xlim',[0 samples-1]);        % Set x-
axis
   grid on
   legend(plot_b{:});
  end

set(gcf,'DefaultTextInterpreter','tex');
subplot(111)
```

4.3. progress.m

```
function cf=progress(cf,elapsed)
%  fighandle=progress(fighandle,elapsed)
%
%     Displays the completed fraction of a
simulation.
%
%     fighandle=progress;
%       initializes the figure window and
returns a handle.
%
%     progress(fighandle,elapsed)
%       updates the plot (0<=elapsed<=100)
%
%     progress(fighandle,100)
%       removes the plot
%
% Magnus Norgaard, Department of Automation,
Technical University of Denmark
% LastEditDate: June 5, 1997.

% ----- Initialize figure window -----
if nargin==0,

cf=figure('Units','Centimeters','Position',[1.5
1.5 10 1.5]);
```

```
  ca=gca;
  set(gca,'Units','Normalized');
  set(gca,'Position',[0.05 0.35 0.9 0.5])
  axis([0 100 0 1]);
  set(cf,'Numbertitle','off','Name','% of
simulation completed');
  set(ca,'box','on','YTickLabel',[]);

elseif nargin==2,
% ----- First time it is called with a % -----
  if isempty(get(gca,'Children')),
    patch([0 0 elapsed elapsed]',[0 1 1 0]','r')

set(get(gca,'Children'),'EraseMode','None','Edge
color','r');

% ----- When it is called again -----
  else

set(get(gca,'Children'),'EraseMode','None','Edge
color','r');
    ax2=get(gca,'Children');
    xdat=get(ax2,'XData');
    set(ax2,'XData',[xdat(3) xdat(3) elapsed
elapsed])
  end
  drawnow
  if elapsed>=100, pause(1),close(cf); end
end
```

4.4. shift.m

```
function out_vector = shift(in_vector,
new_element)
% Shifts all elements of a vector and inserts a
new value
```

```
% in the first element. The last element is
lost.
%
% Inputs:   in_vector   : Vector to be shifted
%           new_element : New element to be
inserted
%
% Outputs:  out_vector  : The updated vector

out_vector = [new_element ;
in_vector(1:length(in_vector)-1)];
```

4.5. siggener.m

```
function
signal=siggen(t,sqr_amp,sqr_frq,sin_amp,sin_frq,
dc,Nlevel,stp_lvl,stp_t)
%
% Signal generator
%
% The function combines square signals with sine
wawes and
% adds a DC-level and a step function.
%
% INPUT:    t       - The time in seconds
%           sqr_amp - Amplitude of square
signals (row-vector)
%           sqr_frq - Frequency of square
signals (column-vector)
%           sin_amp - Amplitude of sine signals
(row-vector)
%           sin_frq - Frequency of sine signals
(column-vector)
%           dc      - [DC off-set level]
(optional)
%           stp_lvl - Step level
%           stp_t   - Step time
% OUTPUT:   signal  - The combined signal
%
```

```
if nargin<=7, stp_lvl=0.0; stp_t=0.0; end;
if nargin<=6, Nlevel = 0.0;end
if nargin<=5, dc=0.0; end;
if nargin<=4, sin_amp=0.0; sin_frq=0.0; end;

signal=0.0;
if t>=stp_t, signal=stp_lvl; end;
signal=signal +
sqr_amp*sign(sin(2*pi*sqr_frq*t));
signal=signal + sin_amp*sin(2*pi*sin_frq*t) + dc
+ Nlevel*randn;
```

4.6. pmntanh.m

```
function t=pmntanh(x)
% PMNTANH
% -------
% Fast hyperbolic tangent function to be used in
% neural networks instead of the tanh provided
by MATLAB
t=(1-2./(exp(2*x)+1));
```