ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR RF1483

SISTEM KONTROL OPTIMAL NON-LINER BERBASIS JARINGAN SYARAF TIRUAN PADA REAKTOR *VAN DER VUSSE*

NURI DIANSYAH
NRP 2402 100 070

Dosen Pembimbing
Dr. Bambang Lelono Widjiantoro, ST.MT

JURUSAN TEKNIK FISIKA
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2006

ITS
Institut
Teknologi
Sepuluh Nopember

FINAL PROJECT - RF 1483

NON-LINEAR OPTIMAL CONTROL SYSTEM BASED ON NEURAL NETWORK AT VAN DER VUSSE REACTOR

NURI DIANSYAH
NRP 2402 100 070

Supervisor
Dr.Bambang Lelono Widjiantoro ST.MT.

ENGINEERING PHYSICS DEPARTMENT
Faculty of Industrial Technology
Sepuluh Nopember Institute of Technology
Surabaya 2006

SISTEM KONTROL OPTIMAL NON-LINIER BERBASIS JARINGAN SYARAF TIRUAN PADA REAKTOR *VAN DER VUSSE*

Nama Mahasiswa : NURI DIANSYAH
NRP : 2402 100 070
Jurusan : Teknik Fisika FTI-ITS
Dosen Pembimbing : Dr. Bambang Lelono W. ST.MT.

Abstrak

Kemampuan Jaringan syaraf tiruan (JST) dalam memetakan hubungan non-linier antara input–output dimanfaatkan dalam pengembangan algoritma sistem kontrol optimal. JST digunakan secara langsung sebagai kontroller, dengan algoritma pelatihan optimal. Sinyal kontrol dibangkitkan berdasarkan optimasi suatu fungsi kriteria J oleh kontroller JST. Sistem yang ditinjau adalah reaktor Van Der Vusse yang merupakan Continous Stirred Tank Reactor (CSTR). Reaktor Van Der Vusse merupakan suatu sistem yang memiliki karakteristik sebagai sistem yang non-linier dan non-minimum phase. Dari penelitian yang telah dilakukan, sistem stabil pada semua set-point, 0.4 mol/liter, 0.3 mol/liter, 0.5 mol/liter, 0.35 mol/liter, 0.6 mol/liter, 0.45 mol/liter dan 0.8 mol/liter dengan masing-masing error steady state sebesar 3.5%, 4%, 3.2%, 4%, 3.8%, 3.11% dan 5%.

***Kata kunci:* Jaringan syaraf tiruan, sistem kontrol optimal, fungsi kriteria, Reaktor Van Der Vusse, CSTR, non-linier, non-minimum phase**

NON-LINEAR OPTIMAL CONTROL SYSTEM BASED ON NEURAL NETWORK AT VAN DER VUSSE REACTOR

Student Name : **NURI DIANSYAH**
NRP : **2402 100 070**
Department : **Engineering Physics Department FTI-ITS**
Supervisor : **Dr. Bambang Lelono Widjiantoro ST.MT.**

Abstract

The ability of Neural Network for mapping non-linear connection between an input-output was used in the development of optimal control system algorithm. Neural Networks were used directly as a controller with optimal training algorithm. Control signal were generated based on a cost function (J) optimization by Neural Network controller. This method was applied on Van Der Vusse reactor, a kind of continuous stirred tank reactor (CSTR). Van Der Vusse reactor has a characteristic as non-linear and non-minimum phase system. From the research has done, it can be seen that the system stable at all set-point, 0.4 mol/liter, 0.3 mol/liter, 0.5 mol/liter, 0.35 mol/liter, 0.6 mol/liter, 0.45 mol/liter dan 0.8 mol/liter where each error steady state were 3.5%, 4%, 3.2%, 4%, 3.8%, 3.11% and 5%.

Key words: ***Neural Networks, optimal control system, cost function, Van Der Vusse Reactor, CSTR, non-linear, non-minimum phase.***

SISTEM KONTROL OPTIMAL NON-LINIER BERBASIS JARINGAN SYARAF TIRUAN PADA REAKTOR *VAN DER VUSSE*

TUGAS AKHIR

OLEH:

NURI DIANSYAH
Nrp. 2402 100 070

Surabaya, Desember 2006
Mengetahui/Menyetujui

Pembimbing,

DR. Bambang Lelono W, ST.MT.
NIP.132 137 895

Ketua Jurusan
Teknik Fisika FTI-ITS

DR.Ir. Totok Suhartanto, DEA
NIP. 131 879 399

SISTEM KONTROL OPTIMAL NON-LINIER BERBASIS JARINGAN SYARAF TIRUAN PADA REAKTOR *VAN DER VUSSE*

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Teknik
pada
Bidang Studi Rekayasa Instrumentasi
Program Studi S-1 Jurusan Teknik Fisika
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember

Oleh:
NURI DIANSYAH
NRP. 2402 100 070

Disetujui oleh Tim Penguji Tugas Akhir:

1. Dr. Bambang Lelono W, ST.MT. ..... (Pembimbing)
2. Ir. Ya'umar, MT. ..... (Penguji I)
3. Ir. Yerri Susatio, MT. ..... (Penguji II)
4. Lizda Johar, ST. MT. ..... (Penguji III)
5. Fitri Adi I. ST. ..... (Penguji IV)

SURABAYA
DESEMBER, 2006

KATA PENGANTAR

Alhamdulillahi Robbil'alamin penulis ucapkan kehadirat Allah SWT atas segala rahmat dan hidayah-Nya akhirnya penulis dapat menyelesaikan Tugas Akhir dengan judul: **SISTEM KONTROL OPTIMAL NON-LINIER BERBASIS JARINGAN SYARAF TIRUAN PADA REAKTOR VAN DER VUSSE.**

Dengan terselesaikannya kegiatan dan laporan Tugas Akhir ini, penyusun juga mengucapkan terimakasih kepada pihak-pihak yang telah membantu selama pelaksanaan dan penyelesaian penelitian.

1. **Bapak Dr. Ir. Totok Soehartanto, DEA,** selaku Ketua Jurusan Teknik Fisika - ITS.
2. **Ayah, Ibu, kakak dan adik-adikku** atas segala dukungan, semangat dan doa yang telah diberikan.
3. **Bapak DR. Bambang Lelono Widjiantoro ST.MT**, yang telah memberikan banyak wawasan mengenai jaringan syaraf tiruan untuk identifikasi dan kontrol serta dengan sabar memotivasi dan membimbing penulis untuk menyelesaikan tugas akhir ini.
4. **Bapak Suyanto ST.MT.** atas segala kesabaran dan perhatiannya selama menjadi dosen wali penulis.
5. Seluruh **Dosen Teknik Fisika ITS** atas kesabarannya dan semua ilmu yang telah diberikan kepada penulis, sehingga penulis bisa menjadi seperti saat ini.
6. Seluruh **staf dan karyawan Teknik Fisika ITS** atas pelayanan yang diberikan.
7. Teman-Teman Teknik Fisika : **"Larins Team", "Boys Don't Cry", "Green House Family"** dan seluruh **angkatan 2002**. Dan juga tak lupa kepada **Wuri Dianasari** atas dukungannya selama ini.

Penulis mengharapkan agar semua proses penyusunan laporan tugas akhir membawa manfaat bagi banyak pihak. Penulisan tugas akhir ini masih banyak kekurangan, oleh karena

itulah segala saran dan kritik diharapkan penulis demi sempurnanya laporan ini.

Surabaya, Desember 2006

Penulis.

DAFTAR ISI

BAB **HALAMAN**

LEMBAR JUDUL i
ABSTRAK iii
LEMBAR PENGESAHAN v
KATA PENGANTAR vii
DAFTAR ISI ix
DAFTAR GAMBAR xi
DAFTAR TABEL xiii
I. PENDAHULUAN 1
1.1 Latar Belakang Permasalahan 1
1.2 Rumusan Permasalahan 2
1.3 Batasan Masalah 2
1.4 Tujuan 3
1.5 Metodologi Penelitian 3
1.6 Sistematika 4
1.7 Manfaat 5
II. TEORI PENUNJANG 7
2.1 Jaringan Syaraf Tiruan 7
2.1.1 Identifikasi Sistem Berbasis JST *(Neural Networks Based System Identification)*. 11
2.1.2 Sistem Kontrol Berbasis JST *(Neural Networks Based Control Systems)* 18
2.2 Sistem Kontrol Optimal (*Optimal Control System*) 21
2.3 Reaktor *Van Der Vusse* 25
III. PERANCANGAN SISTEM 31
3.1 Pembangkitan Data *Input-output* Pada Reaktor *Van Der Vusse* 32
3.2 Pemodelan Reaktor *Van Der Vusse* Dengan JST 33
3.3 Pelatihan Kontroller Optimal. 35
3.4 Simulasi Optimal Kontrol Proses 38

IV. HASIL dan ANALISA. 41
4.1 Data *Input-output* Reaktor. 41
4.2 Pemodelan Sistem Dengan Jaringan Syaraf Tiruan . . 42
4.3 Pelatihan Kontroller Optimal 47
4.4 Simulasi Kontroller Optimal 59
V. KESIMPULAN DAN SARAN 67
5.1 Kesimpulan 67
5.2 Saran 67
DAFTAR PUSTAKA 69
LAMPIRAN A Listing program Untuk Pengambilan Data. . A-1
LAMPIRAN B Listing Program Pemodelan Reaktor Dengan JST B-1
LAMPIRAN C Listing Program Pelatihan Kontroller Dengan Algoritma Kontrol Optimal C-1

DAFTAR GAMBAR

GAMBAR **HALAMAN**

Gambar 2.1 : a. Sel Otak Manusia 7
b. Struktur Jaringan Syaraf Tiruan 7
Gambar 2.2 : Pelatihan JST 9
Gambar 2.3 : Dua lapis *MLP* umpan maju terkoneksi penuh dengan 3 *inputs*, 2 lapis tersembunyi dan 2 lapis *output* 10
Gambar 2.4 : *Series-Paralel* Model 12
Gambar 2.5 : *Paralel / NOE* Model 13
Gambar 2.6 : Sebuah unit *input* dimasukkan ke sistem dan diamati bagaimana pengaruhnya terhadap *output*..... 14
Gambar 2.7 : *Direct Control System* 19
Gambar 2.8 : *Indirect Control System* 20
Gambar 2.9 : *Control System* 20
Gambar 2.10: *Direct Inverse Model* 20
Gambar 2.11: *Continous Stirred Tank Reactor CSTR* 25
Gambar 2.12: Reaktor *Van Der Vusse* 26
Gambar 3.1 : Alur Penelitian 31
Gambar 3.2 : Diagram Blok Pengendalian 32
Gambar 3.3 : Struktur JST untuk pemodelan reaktor *Van Der Vusse* 34
Gambar 3.4 : Struktur JST untuk proses pelatihan dengan algoritma kontrol optimal serta simulasi kontrol optimal 37
Gambar 4.1 : Pasangan data *Input-Output* reaktor 42
Gambar 4.2 : *Output* model dan *error* prediksi pada proses pelatihan JST 45
Gambar 4.3 : *Output* model dan *error* prediksi pada proses validasi JST 46

Gambar 4.4 : Respon sistem terhadap perubahan *set-point* dan sinyal kontrol yang dihasilkan pada iterasi ke-1 49
Gambar 4.5 : Respon sistem terhadap perubahan *set-point* dan sinyal kontrol yang dihasilkan pada iterasi ke-2 50
Gambar 4.6 : Respon sistem terhadap perubahan *set-point* dan sinyal kontrol yang dihasilkan pada iterasi ke-3 51
Gambar 4.7 : Respon sistem terhadap perubahan *set-point* dan sinyal kontrol yang dihasilkan pada iterasi ke-4 52
Gambar 4.8 : Respon sistem terhadap perubahan *set-point* dan sinyal kontrol yang dihasilkan pada iterasi ke-5 53
Gambar 4.9 : Respon sistem terhadap perubahan *set-point* dan sinyal kontrol yang dihasilkan pada iterasi ke-6 54
Gambar 4.10 : Respon sistem terhadap perubahan *set-point* dan sinyal kontrol yang dihasilkan pada iterasi ke-7 55
Gambar 4.11 : Respon sistem terhadap perubahan *set-point* dan sinyal kontrol yang dihasilkan pada iterasi ke-8 56
Gambar 4.12 : Respon sistem terhadap perubahan *set-point* dan sinyal kontrol yang dihasilkan pada iterasi ke-9 57
Gambar 4.13 : Performansi sistem dan sinyal kontrol yang dihasilkan dari simulasi proses kontrol optimal 60

DAFTAR TABEL

TABEL **HALAMAN**

Tabel 2.1 : Parameter model Reaktor Van Der Vusse 29
Tabel 4.1 : Matrik bobot dari lapis *input* ke lapis tersembunyi pada pemodelan sistem..... 43
Tabel 4.2 : Matrik bobot dari lapis tersembunyi ke lapis *output* pada pemodelan sistem..... 44
Tabel 4.3 : Matrik bobot awal dari lapis *input* ke lapis tersembunyi untuk pelatihan kontroller. 47
Tabel 4.4 : Matrik bobot awal lapis tersembunyi ke lapis *output* untuk pelatihan kontroller. 48
Tabel 4.5 : Matrik bobot lapis *input* ke lapis tersembunyi pada jaringan kontroller setelah pelatihan 58
Tabel 4.6 : Matrik bobot lapis tersembunyi ke lapis output pada jaringan kontroller setelah pelatihan 58
Tabel 4.7 : Performansi sistem pada setiap *set-point* pada proses pelatihan 64
Tabel 4.8 : Sinyal kontrol yang dihasilkan pada setiap *set-point* pada proses pelatihan 64
Tabel 4.9 : Performansi sistem pada setiap *set-point* saat simulasi kontrol 65
Tabel 4.10: Sinyal kontrol yang dihasilkan pada setiap *set-point* saat simulasi kontrol 65

BAB I
PENDAHULUAN

1.1 Latar Belakang Permasalahan

Optimalisasi sebuah kontroller pada sebuah proses industri menawarkan berbagai keuntungan, antara lain adalah penghematan sumberdaya yang berarti pula penghematan biaya produksi. Akan tetapi, untuk sistem dengan non-linieritas yang tinggi, optimalisasi kontroller sulit untuk diaplikasikan. Algoritma sistem kontrol optimal pada umumnya adalah algoritma sistem kontrol linier yang didasarkan pada penggunaan model non-linier. Penerapan sistem kontrol linier pada suatu sistem yang non-linier kurang menghasilkan performansi yang bagus, sehingga perlu dikembangkan sistem kontrol optimal non-linier berbasiskan algoritma yang juga non-linier.

Salah satu algoritma non-linier yang handal dalam pemodelan sistem dengan non-linieritas yang tinggi adalah jaringan syaraf tiruan (JST). Selain dapat digunakan sebagai pemodelan sistem, JST juga dapat dimanfaatkan sebagai kontroller proses non-linier, baik secara langsung maupun tidak langsung. Secara tidak langsung, berarti bahwa JST digunakan bersama kontroller lain, dimana fungsi dari JST adalah sebagai *tuning* kontroller utama. Secara langsung, berarti bahwa JST digunakan langsung sebagai sebuah kontroller, tanpa bantuan dari kontroller lainnya. Penggunaan jaringan syaraf tiruan sebagai kontroller secara langsung, menawarkan keuntungan pada *real-time platform*. JST dapat bertingkah laku sebagai *black bock processing*, artinya bahwa JST tidak memerlukan suatu persamaan matematis dari sistem yang akan dikontrol. JST hanya memerlukan serangkaian data *input-output* dari sistem. Sehingga sangat cocok untuk sistem dengan non-linieritas yang tinggi.

Dalam tugas akhir ini sistem yang ditinjau adalah reaktor *Van Der Vusse* yang merupakan jenis *Continous Stirred Tank Reactor (CSTR)*. Reaktor *Van Der Vusse* merupakan suatu sistem yang memiliki karakteristik sebagai sistem yang non-linier dan *non-minimum phase*, sehingga sangat sesuai dengan metode kontrol yang akan dikembangkan.

1.2 Rumusan Permasalahan

Jaringan syaraf tiruan (JST) merupakan salah satu metode untuk memodelkan dan mengontrol suatu sistem non-linier. Dengan sejumlah data input-output dari suatu sistem, maka sistem tersebut dapat dimodelkan dengan akurasi tinggi sekaligus dikontrol dengan metode ini. Oleh karena itu, permasalahan yang diangkat dalam penelitian ini adalah bagaimana mengaplikasikan JST untuk memodelkan sekaligus mengontrol dengan algoritma kontrol optimal reaktor *Van Der Vusse* yang memiliki non-linieritas tinggi.

1.3 Batasan Masalah

Batasan masalah yang digunakan dalam tugas akhir ini antara lain:

- Struktur jaringan syaraf tiruan yang digunakan adalah struktur NNARX atau model *series-paralel.*
- Algoritma pembelajaran menggunakan algoritma orde dua *Lavenberg-Marquardt.*
- Produk yang dianalisa dari reaktor *Van Der Vusse* adalah produk B *(cyclopentenol)*, dengan mengatur flow dari produk A *(cyclopentaddiene)*
- Suhu dan volume reaktor *Van Der Vusse* dianggap konstan.

- Optimalisasi yang dimaksud adalah optimalisasi *unconstraint* dari kontroller JST berdasarkan fungsi kriteria tertentu.
- Pelatihan kontroller optimal menggunakan algoritma *recursive Gauss-Newton* dengan metode *forgetting factor.*

1.4 Tujuan

Tujuan yang ingin dicapai dalam tugas akhir ini adalah mengembangkan algoritma kontrol optimal berbasis jaringan syaraf tiruan dan menganalisa performasi yang dihasilkan.

1.5 Metodologi Penelitian

Metodologi yang direncanakan untuk mengerjakan tugas akhir ini adalah:

- Studi literatur mengenai konsep:
  - *Reaktor Van Der Vusse,* khususnya mengenai karakteristik dan persamaan *input-output* reaktor.
  - Konsep dasar Jaringan syaraf tiruan.
  - Jaringan syaraf tiruan untuk pemodelan sistem non-linier.
  - Algoritma kontrol optimal.
  - Jaringan syaraf tiruan sebagai kontroller optimal.

- Perancangan sistem, yang meliputi:
  - Penurunan rumus dinamik reaktor *Van Der Vusse.*
  - Pembuatan blok diagram dari sistem kontrol.

- Pembuatan software simulasi, yang meliputi:
  - Pembangkitan data *input-output* reaktor *Van Der Vusse.*
  - Pemodelan reaktor dengan jaringan syaraf tiruan secara offline.
  - Pelatihan kontroller dengan algortima kontrol optimal

- Simulasi proses kontrol optimal.

➢ Analisa hasil simulasi

➢ Penulisan laporan

1.6 Sistematika Laporan

Sistematika laporan yang digunakan dalam penyusunan laporan tugas akhir ini adalah sebagai berikut :

Bab I PENDAHULUAN
Berisi tentang latar belakang, permasalahan, batasan masalah, tujuan, metodologi penelitian, dan sistematika laporan.

Bab II TEORI PENUNJANG
Berisi tentang dasar teori, Jaringan syaraf tiruan, *Opimal Control based on Neural Network*, Reaktor Van Der Vusse.

Bab III PERANCANGAN SISTEM
Berisi tentang langkah-langkah analisa yang akan dilakukan selama tugas akhir, diantaranya adalah pembangkitan data *input-output*, pemodelan reaktor dengan JST, pelatihan kontroller JST dengan algoritma kontrol optimal dan simulasi proses kontrol optimal.

Bab IV HASIL DAN ANALISA
Berisi tentang hasil pembangkitan data, hasil pemodelan reaktor dan performansi sistem kontrol optimal.

Bab V KESIMPULAN DAN SARAN
Berisi tentang hasil yang diperoleh dari analisa sistem, analisa data, dan saran.

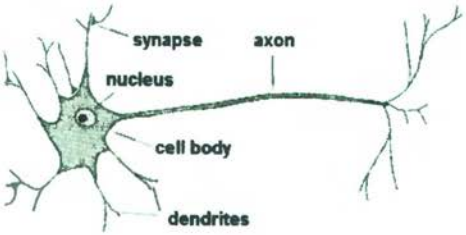
1.7 Manfaat

Manfaat yang dapat dari penelitian ini adalah dikembangkannya sistem kontrol optimal sebagai salah satu alternatif dari bermacam-macam metode kontrol yang ada. Dengan sistem kontrol optimal dapat menghemat sumber daya yang berarti pula penghematan biaya produksi, sehingga diharapkan harga produk dapat ditekan.

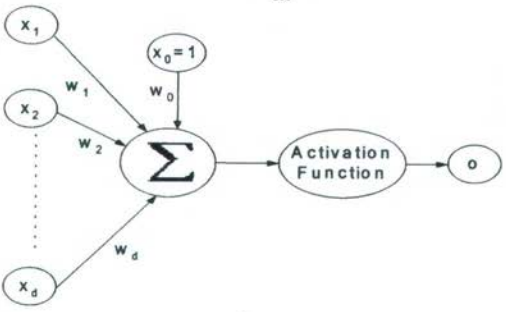
BAB II
DASAR TEORI

2.1 Jaringan Syaraf Tiruan

Jaringan syaraf tiruan (JST) merupakan algoritma yang menirukan cara berpikir otak manusia yang selalu mencoba untuk mensimulasikan proses pembelajaran. Seperti halnya otak manusia, jaringan syaraf tiruan juga terdiri dari beberapa *neuron*, dan terdapat suatu hubungan antara *neuron-neuron* tersebut. *Neuron* akan mentransformasikan informasi yang diterima melalui sambungan keluarnya menuju ke *neuron-neuron* yang lain.



a.



b.

Gambar 2.1 a. Sel otak manusia. b. Struktur Jaringan Syaraf Tiruan

Bagian-bagian JST antara lain:

- *neuron* atau *node*
- bobot *(weight)*
- fungsi aktivasi
- lapis *(layer)*

Neuron (node)

Merupakan unit pemrosesan sederhana. Dalam neuron ini terdapat mekanisme-mekanisme pengolahan data, antara lain: perkalian input dengan bobot dan fungsi aktivasi.

Bobot (wight)

Bobot adalah nilai pemberat dari suatu input yang masuk pada JST. Pada JST, bobot akan mengalami proses adaptasi agar didapatkan suatu fungsi JST yang sesai dengan yang diinginkan.

Fungsi aktivasi Jaringan Syaraf Tiruan

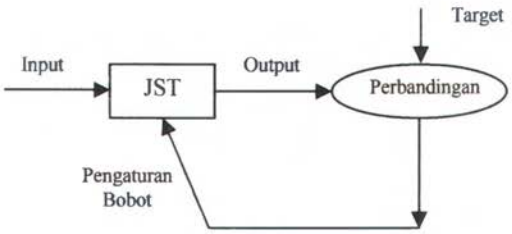
Hasil penjumlahan dari setiap input yang telah dikalikan matrik pembobot akan dibandingkan dengan suatu nilai ambang *(treshold)* tertentu melalui fungsi aktivasi setiap *neuron*. Beberapa fungsi aktivasi antara lain:

1. Fungsi undak *biner*
2. Fungsi bipolar
3. Fungsi *linear*
4. Fungsi *saturating linear*
5. Fungsi *simetric saturating linier*
6. Fungsi *sigmoid biner*
7. Fungsi *sigmoid bipolar*

Lapis (layer)

Merupakan sekumpulan *neuron* yang menjalankan fungsi yang sama. Dalam penerapannya, JST terdiri atas beberapa lapis, diantaranya lapis input, lapis tersembunyi *(hidden layer)* dan lapis *output*.

Proses Pelatihan Jaringan Syaraf Tiruan



Gambar 2.2 Pelatihan JST

Tujuan dari pelatihan adalah untuk mengatur matrik pembobot sehingga JST dapat melaksanakan suatu fungsi. Jenis pelatihan JST antara lain adalah:

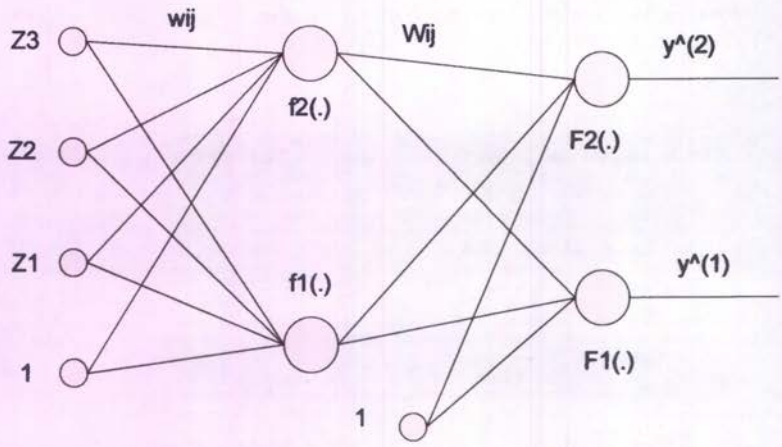
- *supervised learning* (berlatih dengan guru)
- *unsupervised learning* (berlatih tanpa guru)
- *reinforcement learning* (berlatih dengan feedback)

Beberapa tipe algoritma belajar antara lain:

- orde satu: melibatkan turunan pertama *(gradien) error* thd bobot. Contoh: algoritma belajar *backpropagation.*
- orde dua: melibatkan turunan kedua *(hessian) error* terhadap bobot.
  Contoh: algoritma belajar *Lavenberg Marquat.*

Multilayer Percepteron

Multylayer Percepteron (MLP) adalah jaringan yang paling sering mempertimbangkan anggota dari keluarga jarigan syaraf tiruan. Alasan utamanya adalah kemampuannya untuk memodelkan secara sederhana dari hubungan fungsional yang kompleks.



Gambar 2.3 Dua lapis *MLP* umpan maju yang terkoneksi penuh dengan 3 *inputs*, 2 lapis tersembunyi dan 2 lapis *output*

Rumus matematik yang mengekspresikan apa yang terjadi pada jaringan-MLP diambil dari:

$$\hat{y}_i(t) = g_i[\varphi,\theta] = F_i\left(\sum_{j=1}^{n_h} W_{i,j} f_i\left(\sum_{l=1}^{n_\varphi} w_{j,l}\varphi_l + w_{j,0}\right) + W_{i,0}\right) \quad ....(2.1)$$

θ menunjukkan vektor parameter yang didalamnya terdapat semua parameter JST yang dapat diatur (bobot dan bias).

Untuk mendapatkan nilai bobot, terlebih dahulu harus didapatkan suatu contoh bagaimana hubungan antara output dengan input. Suatu cara untuk mendapatkan bobot disebut sebagai pelatihan *(training)* atau pembelajaran *(learning)*, yang pada dasarnya merupakan sebuah masalah optimasi.

2.1.1 Identifikasi sistem berbasis jaringan syaraf tiruan *(Neural Networks Based System Identification)*

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:

- *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.
- *Empirical mode*: didasarkan pada hubungan input-output sistem.
  Keuntungan: detail proses yang terjadi tidak perlu dicari terlebih dahulu dan dapat digunakan untuk model yang sangat kompleks.

Sesuai dengan karakteristik yang dimiliki oleh JST, maka model yang dihasilkan oleh JST merupakan *empirical model* serta *non-parametric* model. Fokus utama dari sistem identifikasi dengan JST hanya untuk sistem non-linier yang dinyatakan sebagai berikut:

$$y(k) = f(x(k)) \qquad ...(2.2)$$

$$x(k) = \left[u(k-1)..u(k-n_u) \quad y(k-1)..y(k-n_y)^T\right] \qquad ...(2.3)$$

dimana:

f : fungsi nonlinier
$x(k)$: *regressor*
$y(k)$: *output* sistem
$u(k)$: *input* sistem

n_u : *history length* untuk *input* sistem
n_y : *history length* untuk *output* sistem

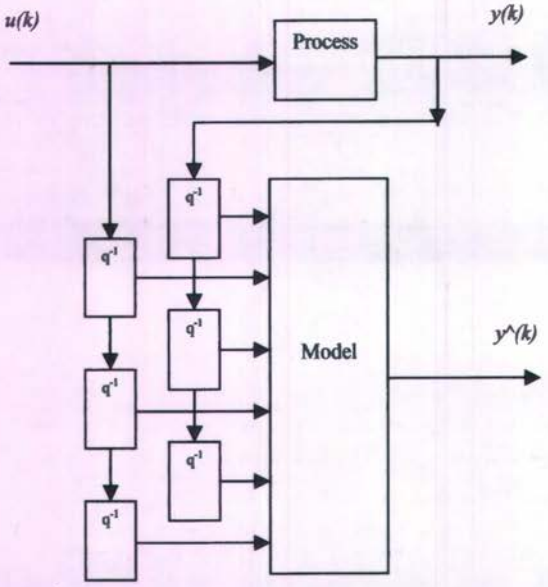
Pada prinsipnya, sistem identifikasi non linier dapat dibedakan menjadi 2, yaitu:

- *Series-parallel / NARX (Non-linier Auto Regresive with eXogenous input) model*:
  Persamaannya:

$$\hat{y}(k) = \hat{f}(x(k)) \quad ...(2.4)$$

$$x(k) = [u(k-1)..u(k-n_u) \quad y(k-1)..y(k-n_y)] \quad ...(2.5)$$

Gambar 2.3 memperlihatkan diagram dari model serial-parallel atau *NARX*.



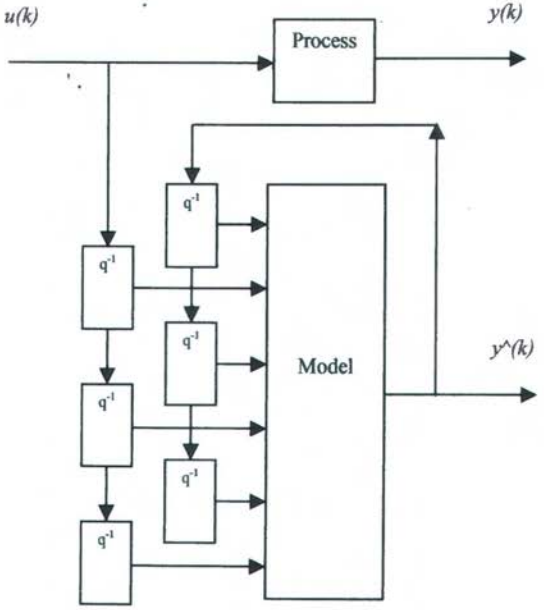
Gambar 2.4 *Series-parallel* model

- *Parallel / NOE (Non-linear Output Error) model*
  Persamaannya:

$$\hat{y}(k) = \hat{f}(x(k)) \quad ...(2.6)$$

$$x(k) = [u(k-1)..u(k-n_u) \quad \hat{y}(k-1)..\hat{y}(k-n_y)] \quad ...(2.7)$$

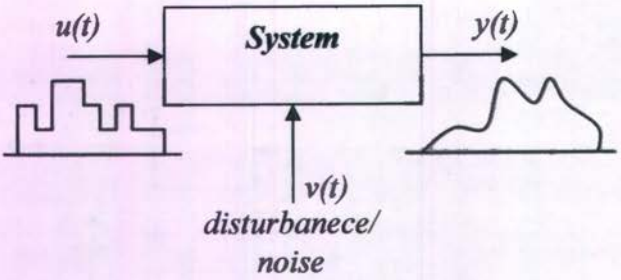
Gambar 2.4 memperlihatkan diagram dari model paralel atau *NOE*



Gambar 2.5 *Parallel / NOE* model

Tahapan dalam sistem identifikasi:

- *Experiment*, 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 *experiment* adalah untuk memasukkan *input* yang bervariasi, *u*, dan mengamati akibatnya pada *output,y*.



Gambar 2.6 Sebuah *input* dimasukkan ke sistem dan diamati bagaimana pengaruhnya terhadap *output*

Pasangan data yang berhubungan dengan input dan output:

$$Z^N = \{[u(t), y(t)], T = 1,...., N\} \quad ...(2.8)$$

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 pemrossan data.

- *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. Struktur model adalah pasangan kandidat model. Masalah utama dalam pemilihan model struktur adalah:
  a. 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.*

b. 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 satu periode sampling dan *output* saat ini tergantung dari dua *output* masa lampau dan tiga *input* masa lampau.

- *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 kriteria 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 *one-step a head prediction* paling bagus dengan *squared error* terkecil antara *output* sistem dengan *output* prediksi. Dalam tahap ini, proses yang paling penting adalah penentuan bobot jaringan atau proses pelatihan. Pasangan data diberikan oleh persamaan (2.7) dan pasangan model kandidat adalah:

$$y(t) = \hat{y}(t \mid \theta) + e(t) = g[t, \theta] + e(t) \qquad ...(2.9)$$

Tujuan dari pelatihan adalah untuk mendapatkan sebuah pemetaan dari pasangan data ke pasangan kandidat model

$$Z^N \rightarrow \hat{\theta} \qquad ...(2.10)$$

sehingga didapatkan model yang menyediakan prediksi mendekati *output* sistem yang sebenarnya. Metode yang paling sering digunakan untuk mengukur kemiripan antara model output dengan model sebenarnya adalah tipe kriteria *mean square error.*

$$V_N(\theta, Z^N) = \frac{1}{2N}\sum_{t=1}^{N}[y(t) - \hat{y}(t \mid \theta)]^2 = \frac{1}{2N}\sum_{t=1}^{N}\varepsilon^2(t, \theta) \qquad ...(2.11)$$

Pola ini disebut sebagai *Prediction Error Methode (PEM)*, dimana tujuannya adalah untuk meminimasi jumlah dari error prediksi. Fitur utama dari kriteria *mean square error* adalah kesederhanann pemakaiannya, dimana aturan update bobot dapat diperoleh dan pengetahuan tentang distribusi *noise* biasanya tidak diperhitungkan. Beberapa metode untuk mencari *prediction error* antara lain:

a. *Metode Lavenberg-Marquardt.*

Serch direction yang dicari dari metode *Gauss-Newton* perlu dioptimalisasi. *Search direction* dicari melalui sebuah kriteria perkiraan, $L^{(i)}(\theta)$ yang diharapkan dapat *valid* hanya pada iterasi saat ini. Jika minimum $L^{(i)}(\theta)$ jauh dari iterasi saat ini, $(\theta)^{(i)}$, maka akan didapatkan *search direction* yang kurang baik. Masalah minimasi dapat dirumuskan sebagai:

$$\theta^{(i+1)} = \arg\min_{\theta} L^{(i)}(\theta) \text{ dimana } \left|\theta - \theta^{(i)}\right| \le \delta^{(i)} \quad ...(2.12)$$

Aturan update yang terjadi ketika menyelesaikan masalah kendala optimasi dapat diselesaikan pada Marquardt (1963):

$$\theta^{(i+1)} = \theta^{(i)} + f^{(i)} \quad ...(2.13)$$

$$\left[R(\theta^{(i)}) + \lambda^{(i)} I\right] f^{(i)} = -G(\theta^{(i)}). \quad ...(2.14)$$

Sebuah bola dengan jari-jari $\delta^{(i)}$ akan diinterpretasikan sebagai wilayah sekitar $\theta^{(i)}$ dimana perkiraan $L^{(i)}(\theta)$ dapat dipercaya sebagai perkiraan yang tepat terhadap kriteria yang sebenarnya $V_N(\theta, Z^N)$.

Untuk menghitung akurasi dari perkiraan, digunakan rasio sebagai berikut:

$$r^{(i)} = \frac{V_N(\theta^{(i)}, Z^N) - V_N(\theta^{(i)} + f^{(i)}, Z^N)}{V_N(\theta^{(i)}, Z^N) - L^{(i)}(\theta^{(i)} + f^{(i)})}. \quad ...(2.15)$$

Apabila rasionya mendekati satu, maka $L^{(i)}(\theta^{(i)} + f)$ merupakan perkiraan ke $V^{N,}$ dan λ harus dikurangi oleh

suatu faktor. Sebaiknya, apabila rasionya cukup kecil atau negatif, λ juga harus ditingkatkan.

Definisi dari $L^{(i)}(\theta)$, $L^{(i)}(\theta^{(i)}+f)$ dapat dituliskan:

$$L^{(i)}(\theta^{(i)}+f)=V_N(\theta^{(i)},Z^N)+f^T G(\theta^{(i)})$$

$$+\frac{1}{2}f^T H(\theta^{(i)})f \qquad ...(2.16)$$

dengan menata kembali ekspresi yang digunakan untuk mendapatkan *search direction*:

$$R(\theta^{(i)})f^{(i)}=-G(\theta^{(i)})-\lambda f^{(i)}, \qquad ... (2.17)$$

Dan dengan memasukkannya kembali ke persamaan (2.16), maka:

$$V_N(\theta^{(i)},Z^N)-L^{(i)}(\theta^{(i)}+f^{(i)}=$$

$$\frac{1}{2}\left[-(f^{(i)})^T G(\theta^{(i)})+\lambda^{(i)}\left|f^{(i)}\right|^2\right] \qquad ...(2.18)$$

Yang membuat rasionya, *r(i)*, menjadi kecil dan siap digunakan dalam algoritma.

b. *Exponential Forgetting*

Suatu strategi yang memungkinkan diabaikannya informasi masa lampau adalah untuk memunculkan suatu *forgetting factor* pada suatu kriteria:

$$V_t(\theta,Z^t)=\frac{1}{2t}\sum_{k=1}^{t}\lambda^{t-k}\varepsilon^T(k,\theta)\varepsilon(k,\theta). \qquad ...(2.19)$$

Pada kasus ini, mekanisme updatenya diekspresikan sebagai berikut:

$$K(t)=\frac{P(t-1)\psi(t)}{\lambda+\psi^T(t)P(t-1)\psi(t)} \qquad ...(2.20)$$

$$\theta(t)=\theta(t-1)+K(t)[y(t)-\hat{y}(t\mid\theta(t-1))] \qquad ...(2.21)$$

$$P(t)=[P(t-1)-K(t)\psi^T(t)P(t-1)]\frac{1}{\lambda} \qquad ...(2.22)$$

Forgetting factor λ harus dipilih pada interval [0,1], dan dalam prakteknya direkomendasikan bahwa suatu nilai yang dekat dengan kesatuan. Jika λ terlalu kecil, akan terjadi suatu fenomena yang disebut *covariance blow-up*. Hal itu terjadi ketika pada arah tertentu dari ruang parameter, informasi dari *input* masa lampau banyak yang dilupakan dari pada informasi baru yang didapatkan. Hal ini akan meningkatkan *eigenvalues* tertentu pada *covariance matrik* sebanding dengan kenaikan variansi pada parameter tertentu. Untuk mengatasi *covariance blow-up* adalah dengan menentukan batas atas pada *eigenvalues* di *covariance* matrik.

- *Model validation*, diperlukan untuk mengetahui apakah model yang telah diperoleh mampu memenuhi kebuthuan yang diperlukan. [Norgaard, Magnus, 2000].

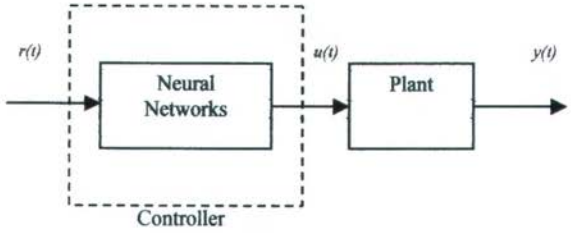
2.1.2 Sistem kontrol berbasis jaringan syaraf tiruan *(Neural Network Based Control System)*

Contoh lain penerapan JST adalah penerapannya dalam suatu sistem kontrol. Karakteristik JST sebagai sistem non-linier sangatlah sesuai bagi penerapan pada sistem kontrol non linier. Secara garis besar, penerapan JST pada sistem kontrol non-linier dapat dibedakan menjadi:

- *Direct Control System Design*
- *Indirect Control System Design*

Direct Control System Design

Pada *direct control system design*, JST digunakan sebagai non-linier kontroller secara langsung. Hal itu berarti JST akan membangkitkan sinyal kontrol yang daplikasikan pada *plant*.



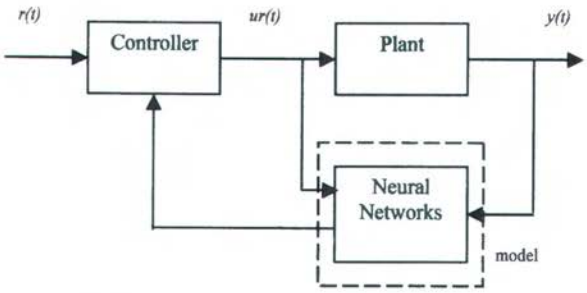
Gambar 2.7 *Direct Control System*

Beberapa contoh *direct control design* antara lain:

- *Direct Inverse Control*
- *Internal Model Control (IMC)*
- *Optimal Control*

Indirect Control System Design

Pada *indirect control system design*, JST akan digunakan sebagai model proses non-linier dari pada kontroller 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*.



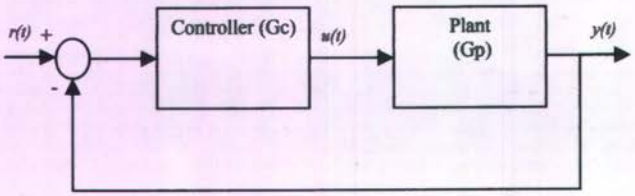
Gambar 2.8 *Indirect Control System*

Idirect Control System Design terdiri atas:

- *Non Linier Model Predictive Control*
- *Model Reference Adaptive Control (MRAC).*

Direct Inverse Control

Diberikan sistem dengan diagram seperti gambar dibawah ini:



Gambar 2.9 *Control System*

Diasumsikan bahwa kita dapat memperoleh model plant dengan menggunakan sistem identifikasi sebagai berikut:

Gm: model plant / prosess

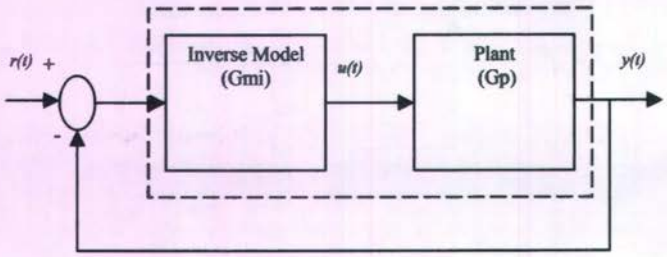
$$Gm \approx Gp \text{ atau } Gm = \hat{G}\, p \qquad ...(2.23)$$

dan diasumsikan bahwa juga terdapat invers dari model:

$$Gmi = G_m^{-1} \qquad ...(2.24)$$

Gmi adalah invers model

Jika *Gc* digantikan dengan inverse model, akan didapatkan:



Gambar 2.10 *Direct Inverse Model*

$$Gmi * Gp = G_m^{-1} * Gp \Rightarrow Gm = Gp \text{ atau } G_m^{-1} = G_p^{-1}$$
$$= G_p^{-1} * Gp$$
$$= 1$$

Konsekuensinya : $y(t) = 1r(t)$, output proses akan sama dengan set point

2.2 Sistem Kontrol Optimal *(Optimal Control System)*

Ide dari kontrol optimal adalah untuk merancang kontroller berdasarkan pada sebuah kriteria tertentu, dimana proses *tracking* referensi didapatkan saat terjadi *penalty* pada magnitudo (beberapa fungsi) dari input kontrol. Perancangan kontrol optimal diimplementasikan dengan melatih Jaringan Syaraf Tiruan agar dapat meminimasi fungsi kriteria tersebut. Kontroller diimplementasikan sebagai sebuah pengembangan dari *specialized training* dari *invers model control*. Pengembangan tersebut adalah penambahan sebuah faktor *penalty* pada *squared control input*. Hal ini akan membuat sinyal kontrol menjadi lebih halus dan mengurangi magnitudo dari sinyal. Harga ini dibayar dengan berkurangnya performasi sistem terhadap *set point*.

Kekurangan utama dari metode ini adalah bahwa dibutuhkannya pelatihan jaringan kembali setiap terjadi perubahan dari *penalty factor*. Hal ini juga merupakan kekurangan dengan menggunakan jaringan syaraf tiruan sebagai kontroller secara langsung.

Pelatihan Kontroller Optimal

Pada algoritma pembelajaran khusus bertujuan untuk meminimasi fungsi kriteria sebagai berikut:

$$J(\theta) = \sum_t [r(t) - y(t)]^2 \qquad ...(2.25)$$

secara *on-line* dengan algoritma pembelajaran *recursive*. Dengan menambahkan kondisi untuk kontrol kuadratik:

$$J(\theta) = \sum_{t} [r(t) - y(t)]^2 + \rho u^2 \quad \rho \geq 0 \qquad ...(2.26)$$

maka jenis sederhana dari kriteria kontrol optimal dapat dicapai. Jika $\rho = 0$ persamaan *(26)* akan serupa dengan algoritma pembelajaran khusus konvensional, dan kontroller akan menjadi *invers* dari sistem. Seiring peningkatan ρ, sebuah invers *'detuned'* juga didapatkan. Deviasi dari kontroller akan meningkat dari kontroller *dead-beat* murni, sementara sinyal kontrol menjadi lebih halus dan mencapai nilai yang lebih kecil. Pada persamaan 2.26, digunakan kriteria *sum square error* dari persamaan 2.25, antara *set point* dengan *output* proses. *Output* dari pemodelan sistem dengan JST $(\hat{y})$ digunakan untuk memprediksi *output* sistem *one-steap ahead*. Prediksi *output* selanjutnya digunakan untuk membangkitkan sinyal kontrol berdasarkan *error* yang didapatkan.

Seperti pada algoritma pembelajaran khusus, kontroller optimal dilatih secara *on-line*, contohnya dengan algoritma pembelajaran *recursive back-propagation* atau *Gauss-Newton*. Untuk konvergensi yang relatif cepat, algritma *Gauss-Newton* lebih disarankan. Sayangnya, keadaan yang baru berakibat pada kriteria *error* bukan lagi *mean-square error*, dimana *mean-square error* adalah kriteria *error* dari algoritma *Gauss-Newton*. Meskipun demikian, dapat dimungkinkan untuk diperoleh semacam modifikasi *adhoc* dari algoritma yang memenuhi kriteria modifikasi. Asumsi dasar yang digunakan adalah (jika *forgetting* tidak digunakan):

$$J_t(\theta, Z^t) = J_{t-1}(\theta, Z^{t-1}) + [r(t) - y(t)]^2 + \rho u^2(t-1) \qquad ...(2.27)$$

dimana J_{t-1} diasumsikan sudah diminimasi. Perlu diperhatikan bahwa perbedaan antara J_t dan J_{t-1} juga mengandung *u(t-1)*, bukan *u(t)*. Hal ini disebabkan karena jeda waktu dari satu periode cuplik telah diasumsikan, dimana pada kasus *u(t-1)* adalah input kontrol yang paling sering yang mempengaruhi *y(t)*.

Persamaan berikut ini digunakan untuk mendapatkan turunan kedua pada sisi sebalah kanan, dengan mempertimbangkan beban JST:

$$G(\theta) \approx \frac{du(t-1)}{d\theta}\left[-\frac{\partial \hat{y}(t)}{\partial u(t-1)}e(t)+\rho u(t-1)\right] \quad ...(2.28)$$

agar lebih singkat, maka:

$$e_u(t)=\frac{\partial \hat{y}(t)}{\partial u(t-1)}e(t) \quad ...(2.29)$$

$$\psi_u(t)=\frac{\partial u(t-1)}{\partial(\theta)} \quad ...(2.30)$$

$$\psi(t)=\frac{\partial \hat{y}(t)}{\partial u(t-1)}\psi_u(t) \quad ...(2.31)$$

Dengan persamaan (2.29), (2.30), (2.31), maka persamaan (2.28) dapat direduksi menjadi:

$$G(\theta) \approx \psi_u(t)[-e_u(t)+\rho u(t-1)] \quad ..(2.32)$$

Tidak begitu jelas bagaimana *update Hessian* harus dimodifikasi, tetapi pengalaman telah menunujukkan bahwa hal ini tidak terlalu penting. Jenis *update* yang sama yang digunakan pada algoritma pembelajaran khusus dapat digunakan disini. Apabila *constant forgeting factor* digunakan, bobot JST akan di-*update* menurut:

$$P(t)=\left[P(t-1)-\frac{P(t-1)\psi(t)\psi^T(t)P(t-1)}{\lambda+\psi^T(t)P(t-1)\psi(t)}\right]\frac{1}{\lambda} \quad ...(2.33)$$

$$\hat{\theta}(t)=\hat{\theta}(t-1)+P(t)\psi_u(t)[e_u(t)-\rho u(t-1)] \quad ...(2.34)$$

$\psi_u(t)$ dapat diperkirakan dengan turunan parsial dalam hubungan regresi *pseudo-linear*:

$$\psi_u(t) \approx \frac{\partial u(t-1)}{\partial(\theta)} \qquad ...(2.35)$$

Kriteria yang dipertimbangkan adalah kriteria jenis *infinite horizon*. Kontroller untuk referensi trayektori yang spesifik *(finite)* dapat diselesaikan dengan mengulang trayektori hingga bobot mencapai *error* yang konvergen. Hal ini berkaitan dengan minimasi dari:

$$J(\theta) = E\left\{\sum_{t=1}^{N}[r(t) - y(t)]^2 + \rho u^2(t)\right\} \qquad ...(2.36)$$

dimana N adalah panjang dari trayektori referensi. Keuntungan dari pelatihan jaringan secara optimal adalah bahwa konvergensi dari algoritma lebih mudah di-supervisi.

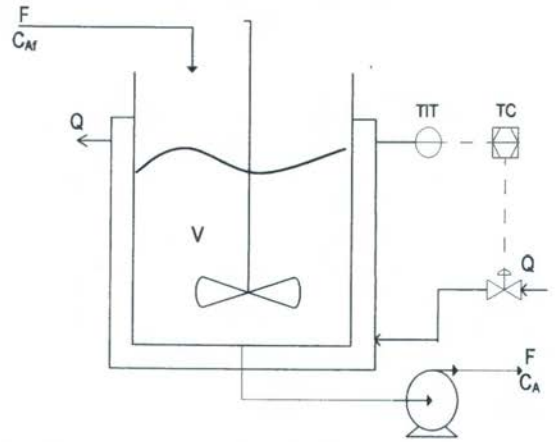
Fitur utama dari optimal kontrol antara lain:

- ❖ Keuntungan:
  - Mudah dituning.
  - Dapat diaplikasikan untuk banyak sistem (apabila dibandingkan dengan *direct inverse control*).
  - *Goal directed.*
  - Cocok untuk mendesain sebuah kontroller untuk sebuah trayektori spesifik.
- ❖ Kekurangan:
  - Jaringan kontroller dilatih secara *on-line.*
  - Jaringan harus dilatih kembali ketika p*enalty factor* dirubah.
  - Inisialisasi bobot dari jaringan termasuk sulit.

[Norgaard, Magnus, 2000].

2.3 Reaktor *Van Der Vusse*

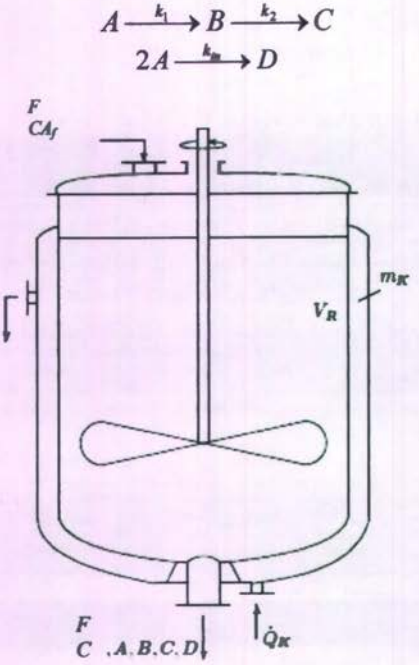
Reaktor kimia merupakan unit paling penting pada *plant* kimia. Jenis dari reaktor kimia bermacam-macam, tetapi dua yang paling umum digunakan adalah *Continous Stirred Tank Reactor (CSTR)* dan *The Plug Flow Reactor (PFR)*. Dua tipe servis reaktor ini membatasi perilaku dari banyak reaktor yang beroperasi. *CSTR* sering digunakan dalam pemodelan dinamik, karena *CSTR* dapat dimodelkan sebagai sebuah *lumped parameter system*. Model dinamik dari *PFR* terdiri dari persamaan turunan parsial (juga dikenal sebagai sistem parameter terdistribusi).



Gambar 2.11 *Continous Stirred Tank Reactor CSTR*

Reaktor *Van Der Vusse* merupakan reaktor *CSTR isothermal*. Untuk menyederhanakan reaksi yang terjadi didalamnya, dalam reaksi terdapat substansi A sebagi input, serangkaian reaksi kimia dan substansi B sebagai output.

Model reaktor ini mensimulasikan reaksi orde satu untuk A → B dengan dua reaksi yang terjadi B → C dan 2A → D. Semua reaksi tersebut dapat dideskripsikan dalam skema reaksi sebagai berikut:



Gambar 2.12 Reaktor *Van Der Vusse*

Reaksi utama pada reaktor *Van Der Vusse* adalah perubahan dari *cyclopentaddiene* (komponen A) menjadi produk *cyclopentenol* (komponen B) dengan katalis asam electrophilic. Raksi paralel menghasilkan *by-product dicyclopentadiene* (komponen D) melalui reaksi Diels-Alder. Lebih jauh lagi, *cyclopentanol* bereaksi lagi dan menghasilkan produk yang tidak diinginkan *cyclopentanediol* (komponen C) sebagai akibat dari penambahan molekul air. [Antonelly, Rita. Astolfi, Alessandro, 2003]. Reaksinya dapat dituliskan sebagai berikut [Conradie, Alex Van Eck, 2000]:

$$C_5H_6 \xrightarrow{+H_3O^+} C_5H_7OH \xrightarrow{+H_3O^+} C_5H_8(OH)_2$$
$$2C_5H_6 \rightarrow C_{10}H_{12}$$

Keseimbangan Massa Keseluruhan

Diasumsikan bahwa sistem dalam keadaan density dan volume yang konstan:

$$\frac{dV}{dt} = 0 \text{ dan } F = F_i \qquad ...(2.37)$$

Keseimbangan Komponen A

Akumulasi = masuk – keluar oleh aliran – keluar oleh reaksi 1 – keluar oleh reaksi 2

$$\frac{d(VC_A)}{dt} = F(C_{Af} - C_A) - Vk_1C_A - Vk_3C_A^2 \qquad ...(2.38)$$

Selama V konstan, maka persamaan (2.38) dapat ditulis :

$$\frac{d(C_A)}{dt} = \frac{F}{V}(C_{Af} - C_A) - k_1C_A - k_3C_A^2 \qquad ...(2.39)$$

Keseimbangan Komponen B

Dengan cara yang sama, didapatkan:

$$\frac{d(C_B)}{dt} = -\frac{F}{V}C_B + k_1C_A - k_2C_B \qquad ...(2.40)$$

Keseimbangan Komponen C

Begitu juga untuk komponen C:

$$\frac{d(C_C)}{dt} = -\frac{F}{V}C_C + k_2C_B \qquad ...(2.41)$$

Keseimbangan Komponen D

Dan juga untuk komponen D:

$$\frac{d(C_D)}{dt} = -\frac{F}{V}C_D + \frac{1}{2}k_3C_A^2 \qquad ...(2.42)$$

Pada persamaan (2.38) sampai dengan persamaan (2.42), persamaan (2.39) dan (2.42) tidak tergantung pada komponen C

dan D. Apabila kita hanya mempertimbangkan tentang komponen A dan B, maka keseimbangan massa untuk komponen A dan B diberikan oleh persamaan sebagai berikut [Waynebequette, B, 1991]:

$$\dot{C}_A = \frac{F}{V}\left(C_{Af} - C_A\right) - k_1 C_A - k_3 C_A^2 \qquad ...(2.43)$$

$$\dot{C}_B = -\frac{F}{V} C_B + k_1 C_A - k_2 C_B \qquad ...(2.44)$$

Dengan output y sebagai berikut:

$$y = C_B \qquad ...(2.45)$$

Dengan menyelesaikan persamaan (2.43) dan (2.44) pada keadaan *steady state*, akan didapatkan persamaan kuadratik dalam C_{As} sebagai berikut:

$$-k_3 C_A^2 + \left(-k_1 - \frac{F_s}{V}\right) C_{As} + \frac{F_s}{V} C_{Afs} = 0 \qquad ...(2.46)$$

Penyelesaian persamaan (2.46) dengan akar positif akan didapatkan:

$$C_{As} = \frac{-\left(k_1 + \frac{F_s}{V}\right)}{2k_3} + \frac{\sqrt{\left(k_1 + \frac{F_s}{V}\right)^2 + 4k_3\left(\frac{F_s}{V}\right)}}{2k_3} \qquad ...(2.47)$$

Sedangkan untuk C_{Bs} didapatkan:

$$C_{Bs} = \frac{k_1 C_{As}}{\frac{F_s}{V} + k_2} \qquad ...(2.48)$$

Tabel 2.1 Parameter model Reaktor *Van Der Vusse* [Kashiwagi, Hiroshi. Li, Yun. 2003].

Simbol	Parameter	Nilai	Satuan
k_1	*rate of reaction 1*	50	h^{-1}
k_2	*rate of reaction 2*	100	h^{-1}
k_3	*rate of reaction 3*	10	h^{-1}
C_A	konsentrasi A *output*	x_1	*mol/liter*
C_{Af}	konsentrasi A *input*	10	*mol/liter*
C_B	konsentrasi B *output*	x_2	*mol/liter*
F/V	*space velocity*	u	h^{-1}

Pada tabel 2.1 terdapat 4 parameter tetap yang sudah diketahui, yaitu k_1, k_2, k_3 dan C_{Af}, sedangkan parameter yang berubah adalah C_A ,C_B dan F/V. *Input* sistem adalah *flow* dari reaktan A *(F/V)*, sedangkan *output* sistem adalah konsentrasi B *(C_B)*. Dengan parameter model pada tabel 2.1, maka persamaan (2.43), (2.44) dan (2.45) dapat ditulis sebagai berikut:

$$\frac{dx_1}{dt} = -50x_1 - 10x_1^2 + (10 - x_1)u \qquad ...(2.49)$$

$$\frac{dx_2}{dt} = 50x_1 - 100x_2 - x_2u \qquad ...(2.50)$$

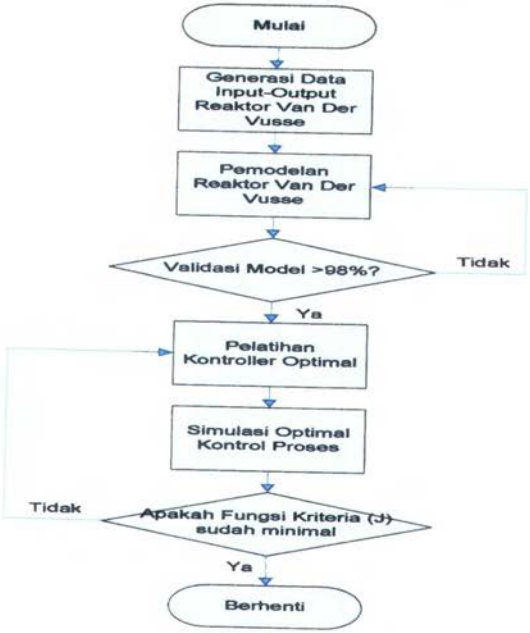
$$y = x_2 \qquad ...(2.51)$$

Semua model reaksi menggunakan kecepatan reaksi *Arrhenius* yang bergantung pada temperature. Model tersebut mempunyai empat bentuk: konsentrasi A, konsentrasi B, temperature reaktor dan jaket pendingin. Dengan proses tresebut diatas, diharapkan agar didapatkan produk B sebanyak mungkin. Sistem pada reaktor ini banyak mempunyai karakteristik yang sangat tidak linear, yang terdiri atas: perkalian input, perubahan *gain sign*, respon *asymetric* dan sifat fase yang *minimum* ke *non-minimum*. Karakteristik yang non-linier tersebut sangat lazim

pada kebanyakan *operating point* yang diinginkan. [Gatzke, Edward P. Doyle, Francis J, 1998]

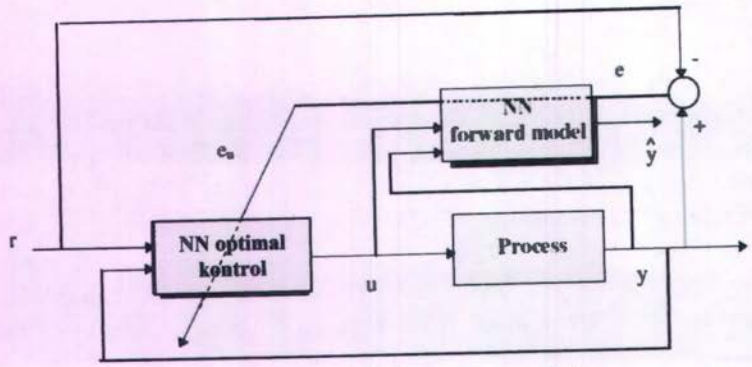
BAB III
PERANCANGAN SISTEM

Pada bab ini akan dibahas mengenai simulasi sistem kontrol optimal berbasis jaringan syaraf tiruan pada reaktor *Van Der Vusse*, yang hasilnya berupa respon dinamik sistem pada berbagai *set point* serta sinyal kontrol yang dihasilkan oleh kontroller JST optimal. Dalam melakukan simulasi ini ada beberapa tahapan, antara lain: eksitasi sinyal *input* untuk mendapatkan *output* pada reaktor *Van Der Vusse*, pemodelan reaktor *Van Der Vusse* dengan menggunakan jaringan syaraf tiruan, pelatihan kontroller optimal, dan simulasi optimal proses kontrol. Berikut ini adalah alur penelitian dari tugas akhir ini.



Gambar 3.1 Alur Penelitian

Berikut ini adalah diagram blok dari sistem.



Gambar 3.2 Diagram Blok Pengendalian

3.1 Pembangkitan Data *Input-output* Pada Reaktor *Van Der Vusse*

Untuk mendapatkan data *input-output* pada reaktor *Van Der Vusse*, maka perlu dilakukan suatu simulasi dengan menggunakan model matematik reaktor *Van Der Vusse* yang telah diketahui pada persamaan (2.49), (2.50) dan (2.51) yaitu:

$$\frac{dx_1}{dt} = -50x_1 - 10x_1^2 + (10 - x_1)u$$

$$\frac{dx_2}{dt} = 50x_1 - 100x_2 - x_2 u$$

$$y = x_2$$

Dengan menggunakan persamaan (2.46), (2.47) dan (2.48), maka persamaan diatas dapat diselesaikan sebagai berikut:

$$-10x_1^2 + (-50 - u)x_1 + 10u = 0 \qquad ...(4.1)$$

$$x_1 = \frac{-50 + u}{20} + \frac{\sqrt{(50+u)^2 + 40u}}{20} \qquad ...(4.2)$$

dan untuk C_{Bs} didapatkan:

$$x_2 = \frac{50x_1}{u+100} \qquad ...(4.3)$$

Data *input* yang digunakan berupa sinyal *APRBS (Amplitude Pseudo Random Binary Signal)*, yaitu sinyal acak dengan amplitudo dan frekuensi yang berubah setiap saat. Sinyal *input* dibatasi pada amplitudo 4 sampai 20, dimana batas tersebut mewakili sinyal kontrol 4-20 mA, yang merupakan sinyal kontrol untuk membuka dan menutup *kontrol valve* pada posisi 0-100%. Posisi dari *kontrol valve* mewakili *mass flow rate* dari reaktan, yaitu *cyclopentaddiene* (komponen A). Sinyal *output* yang dihasilkan mewakili konsentrasi dari *cyclopentenol* (produk B) yang diproduksi oleh reaktor *Van Der Vusse*, dimana produk B adalah produk yang diharapkan dari reaktor *Van Der Vusse*. Amplitudo dari produk B ini berkisar antara 0 sampai 1 *mol/liter*.

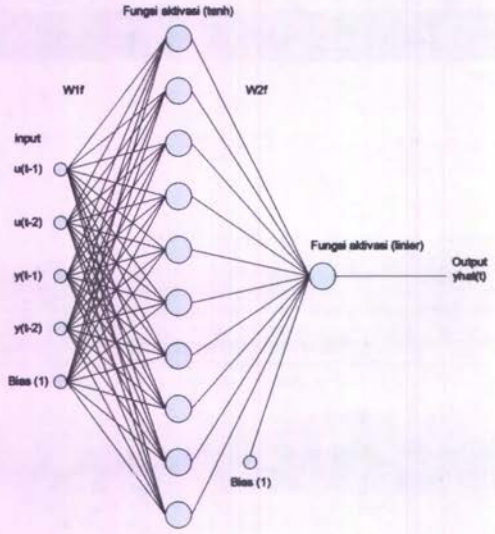
3.2 Pemodelan Reaktor *Van Der Vusse* Dengan JST

Untuk memperoleh *one-step ahead prediction* dari sistem, maka dilakukan suatu pemodelan dari sistem yaitu reaktor *Van Der Vusse* dengan menggunakan jaringan syaraf tiruan. Data *input-output* sistem untuk melakukan pemodelan diperoleh dari proses pembangkitan data. Struktur JST yang digunakan untuk pemodelan adalah model *NNARX (Neural Network Auto Regressive eXogenous)* atau model *series-parallel* dengan jumlah *regressor* sebanyak 2 *input* masa lampau *(u(t-1)* dan *u(t-2))* dan 2 *output* masa lampau *(y(t-1))* dan *(y(t-2))*.

Jumlah *layer* atau lapisan dari JST untuk memodelkan sistem adalah 3 lapis, yaitu lapis *input*, satu lapis tersembunyi dan lapis *output*. Pada lapis *input* terdapat 5 *node* (titik) *input*, yaitu 4 *regressor* dari *input* dan *output* masa lampau dan satu bias yang nilainya adalah 1. Lapis tersembunyi *(hidden layer)* terdiri atas 10

neurons dengan fungsi aktivasi *tangent hyperbolic* dan satu buah bias yang nilainya juga 1. Sedangkan pada lapis *output* terdapat satu buah *neuron* dengan fungsi aktivasi linier.

Untuk mempercepat konvergensi pada proses pelatihan, sebelum data *input-output* dilatih dengan JST dengan struktur seperti diatas, maka data terlebih dahulu di-*scaling* pada *zero mean* dan *variance one*. Setelah data di *scaling*, dilakukan proses pelatihan JST melalui suatu proses iterasi, sehingga didapatkan bobot antara lapis *input* dengan lapis tersembunyi *(W1f)* dan bobot antara lapis tersembunyi dengan lapis *output (W2f)* yang merepresentasikan sifat dari reaktor *Van Der Vusse*. Setelah didapatkan *W1f* dan *W2f* maka bobot tersebut di-*rescaling* untuk mendapatkan bobot asli. Parameter-parameter dari proses pemodelan yaitu struktur JST, jumlah *regressor*, bobot *W1f* dan *W2f* disimpan untuk digunakan dalam kontroller JST optimal. Berikut ini susunan JST untuk pemodelan reaktor.



Gambar 3.3 Struktur JST untuk pemodelan reaktor *Van Der Vusse*

Dalam pemodelan menggunakan JST, ada dua tahapan yang dilakukan, yaitu proses pelatihan data dan proses validasi. Data yang diperoleh dari proses pembangkitan data *input-output* dibagi menjadi dua dengan jumlah yang sama. Pasangan data yang pertama digunakan untuk proses pelatihan jaringan. Tujuan dari proses pelatihan adalah untuk mendapatkan bobot jaringan dari masing-masing lapisan agar jaringan dapat memodelkan sistem dengan tepat. Algoritma pembelajaran untuk mendapatkan bobot menggunakan algoritma *Lavenberg Marquardt* yang merupakan pengembangan dari algoritma pembelajaran orde dua *Quasy-Newton*. Berikut ini alur pembelajaran dengan algoritma *Lavenberg-Marquardt:*

- Memilih sebuah vektor parameter awal $\theta^{(0)}$ dan nilai awal $\lambda^{(0)}$.
- Mencari *search direction* dari: $\left[R(\theta^{(i)} + \lambda^{(i)} I\right] f^{(i)} = -G(\theta^{(i)})$.
- $r^{(i)} > 0.75 \rightarrow \lambda^{(i)} = \lambda^{(i)}/2$
- $r^{(i)} < 0.25 \rightarrow \lambda^{(i)} = 2\lambda^{(i)}$
- Jika $V_N(\theta^{(i)} + f^{(i)}, Z^N) < V_N(\theta^{(i)}, Z^N)$ maka terima $\theta^{(i+1)} = \theta^{(i)} + f^{(i)}$ sebagai iterasi yang baru dan $\lambda^{(i+1)} = \lambda^{(i)}$.
- Jika *stoping* kriteria tidak memuaskan, maka ulangi lagi langkah 2.

Sementara itu, pasangan data *input* yang terakhir digunakan untuk validasi jaringan. Validasi dilakukan dengan struktur dan bobot yang diperoleh dari proses pelatihan. Apabila dari proses validasi diperoleh tingkat $VAV > 98\%$, maka model sistem dapat diterima.

3.3 Pelatihan Kontroller Optimal

Sebelum simulasi kontrol optimal dilakukan, maka perlu dicari bobot dari tiap lapis jaringan. Kontroller JST dilatih agar

dapat bertingkah laku sebagai model *detuned inverse* proses dari reaktor. Jaringan syaraf tiruan dilatih untuk meminimasi fungsi kriteria pada persamaan (2.26).

$$J(\theta) = \sum_t [r(t) - y(t)]^2 + \rho u^2 \quad \rho \geq 0$$

Algoritma pelatihan yang digunakan adalah algoritma *recursive Gauss-Newton* dengan menggunakan metode *forgetting factor*. Berikut ini langkah-langkah update bobot dengan metode *forgetting factor:*

- Inisialisasi nilai $\lambda^{(0)}$ dan elemen diagonal dari covariance matrik
- Inisialisasi covariance matrix dengan mengalikan $p^{(0)}$ dengan matrix satuan
- Membaca *output* dari sistem fisik dan menghitung perkiraan errornya
- Menghitung nilai ψ
- Menghitung K(t) berdasarkan persamaan (2.20)

$$K(t) = \frac{P(t-1)\psi(t)}{\lambda + \psi^T(t)P(t-1)\psi(t)}$$

- Update bobot berdasarkan persamaan (2.21)

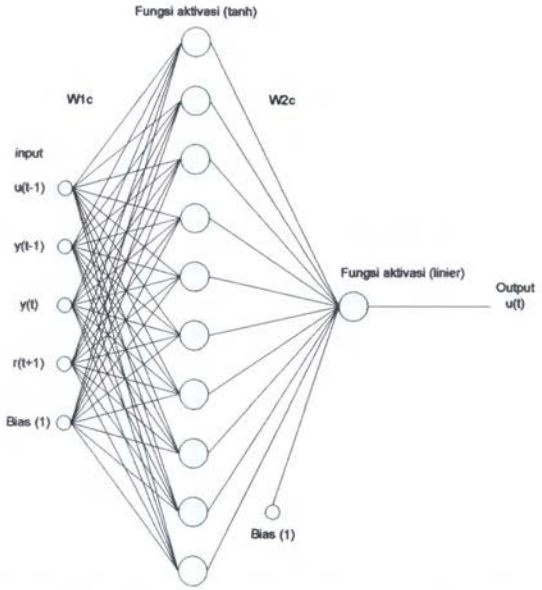
$$\theta(t) = \theta(t-1) + K(t)[y(t) - \hat{y}(t \mid \theta(t-1))]$$

- Update matrik P berdasarkan persamaan (2.22)

$$P(t) = [P(t-1) - K(t)\psi^T(t)P(t-1)]\frac{1}{\lambda}$$

Model sistem yang telah didapatkan dalam pemodelan JST semua parameter-parameternya digunakan disini. Inisialisasi bobot untuk kontroller JST adalah secara acak *(random)* dengan jumlah *regressor* yaitu 1 *input* masa lalu *(u(t-1))*, 1 *output* masa lalu *(y(t-1))* dan 1 output saat ini *(y(t))* serta 1 referensi saat *t+1*. Sedangkan struktur JSTnya yaitu terdiri atas 3 lapis *(layer)*, yaitu lapis *input* yang terdiri atas 5 *node*, lapis tersembunyi *(hidden layer)* yang terdiri atas 10 *neurons* dengan fungsi aktivasi *tangent*

hyperbolic dan satu *node bias* dengan nilai 1. Lapis terakhir adalah lapis *output*, dengan 1 *neuron* dan mempunyai fungsi aktivasi linier. Susunan JST yang dipakai dalam proses pelatihan kontroller sama dengan susunan JST yang dipakai dalam pemodelan sistem. Gambar 3.4 menunjukkan susunan JST yang digunakan.



Gambar 3.4 Struktur JST untuk proses pelatihan kontroller dengan algortima kontrol optimal serta simulasi kontrol optimal

Pada pelatihan kontroller optimal, bobot dari masing-masing lapis akan diupdate melalui suatu iterasi, agar *output* proses dapat meminimalisasi fungsi kriteria *(J)*. Setelah mencapai *J* minimal, parameter-parameter JST seperti bobot dari lapis *input* ke lapis tersembunyi *(W1c)* dan bobot dari lapis tesembunyi ke lapis *output (W2c)* serta struktur dan jumlah *regressor* disimpan untuk digunakan dalam simulasi *kontrol optimal*

3.4 Simulasi Optimal Kontrol Proses

Setelah di dapatkan nilai fungsi kriteria *(J)* minimal pada pelatihan kontrol optimal, tahap selanjutnya adalah mensimulasi kontrol proses optimal pada keadaan lup tertutup *(closed-loop)*. Parameter-parameter dari JST yang didapatkan melalui pelatihan disimulasi pada kontroller JST optimal untuk mendapatkan performanya. Simulasi dilakukan dengan mengubah-ubah set point secara bertahap dari perubahan kecil hingga perubahan yang besar. Respon system serta sinyal kontrol yang dihasilkan dianalisa, apakah sudah memenuhi kriteria atau belum.

Parameter-parameter yang didapatkan dalam proses pelatihan kontroller optimal antara lain:

- *Network Deffinition*, yang berisi informasi mengenai jumlah *neuron* pada setiap lapisan serta fungsi aktivasinya. Jumlah *neuron* yang digunakan adalah 10 *neuron* pada lapis tersembunyi dengan fungsi aktivasi *tangent hyperbolic* dan satu *neuron* pada lapis *output* dengan fungsi aktivasi linier.
- Struktur regressor, yang berisi jumlah regressor atau *history length* dari setiap data *input* dan *output* serta *delay time* setiap *regressor*. *History length* yang digunakan adalah 1 *input* masa lampau *(u(t-1))*, 1 *output* masa lampau *(y(t-1))* dan 1 output saat ini *y(t)*, serta 1 sinyal referensi saat *r+1*, dengan *delay time* sebesar 1 sampel.
- Bobot setiap lapisan, yang berisi mengenai matrik bobot setiap lapisan, baik itu bobot dari lapis *input* ke lapis tersembunyi *(W1c)* dan bobot dari lapis tersembunyi ke lapis *output (W2c)*

digunakan dalam simulasi proses kontrol optimal. Susunan jaringan pada proses simulasi kontrol optimal dapat dilihat pada gambar 3.4.

Informasi mengenai susunan model JST juga digunakan dalam simulasi kontrol. Model yang diperoleh dari proses

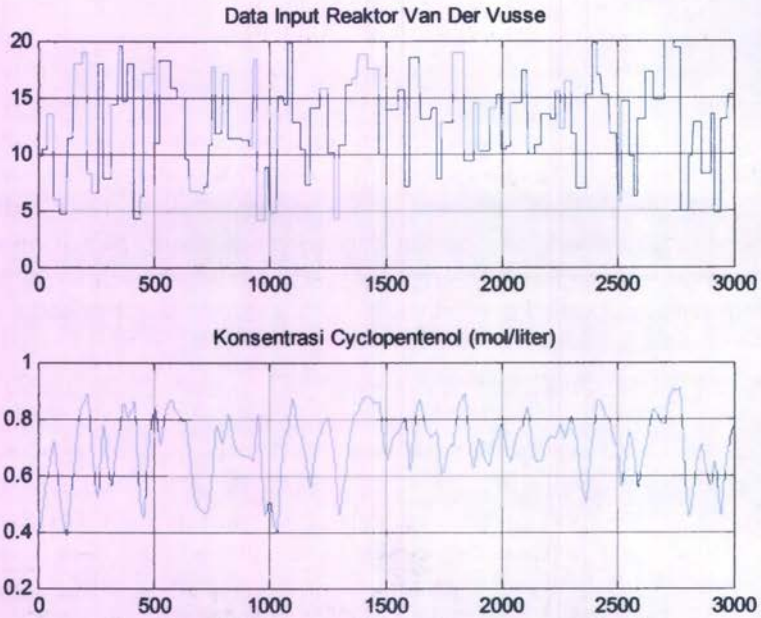
pelatihan JST digunakan untuk mendapatkan prediksi *error* saat *one steap a head*. Parameter-parameter yang digunakan disini sama dengan parameter-parameter pada proses pelatihan kontroller, yaitu: *Network Deffinition*, struktur *regressor* dan bobot setiap lapisan *(W1f* dan *W2f)*.

BAB IV
HASIL DAN ANALISA

Pada bab ini akan diterangkan mengenai data-data yang diperoleh, yaitu data *input* dan *output* pada reaktor *Van Der Vusse*. Juga akan diterangakan mengenai data hasil dari pemodelan reaktor dan respon dari sistem kontrol optimal yang diperoleh, baik itu respon dinamik sistem terhadap perubahan *set point* maupun sinyal kontrol yang dihasilkan dari poses pelatihan.

4.1 Data *Input-output* Reaktor

Data *output* dari reaktor diperoleh melalui proses generasi sinyal. Sinyal *input* berupa sinyal *APRBS* dengan *range* sebesar 4-20 dimasukkan ke dalam persamaan dinamik reaktor *Van Der Vusse* dengan jumlah sampel sebanyak 3000 sampel data *input output*. APRBS diatur agar minimum perubahan amplitudo-nya sebesar 20 sampel dan maksimumnya sebesar 80 sampel. Data *output* dari proses generasi merupakan konsentrasi produk B *(cyclopentenol)* dalam *mol/liter* yang merupakan produk yang diharapkan dari serangkaian reaksi kimia didalam reaktor *Van Der Vusse*. Dengan masukkan sinyal sebesar 4-20, maka *output* produk B berkisar diantara 0-1 *mol/liter*. Serangkaian data *input-output* reaktor yang dibangkitkan dalam proses pembangkitan data ditunjukkan pada gambar 4.1.



Gambar 4.1 Pasangan Data *Input-output* Reaktor

Pada gambar 4.1 dapat dilihat bahwa sifat sistem adalah non-linier. Non-linieritas dapat di lihat dari respon sistem terhadap input yang diberikan, dimana dengan nilai data input yang sama, dihasilkan nilai output yang berbeda. Data *input-output* sistem dapat dilihat pada lampiran D dan E.

4.2 Pemodelan Sistem Dengan Jaringan Syaraf Tiruan

Untuk mendapatkan *one-step ahead prediction* dari sistem, maka sistem dimodelkan dengan jaringan syaraf tiruan. Dari prediksi sistem akan didapatkan *error* prediksi. Error yang dihasilkan digunakan untuk membangkitkan sinyal kontrol. Algoritma pembelajaran yang digunakan adalah algoritma pembelajaran orde dua *L. Marquardt*. Struktur JST yang digunakan adalah model *NNARX* atau model *series-paralel*. Data

yang didapatkan dari proses pembangkitan data dibagi menjadi dua sama banyak. Untuk pelatihan diambil 1500 pasangan data *input-output* pada saat sampel 1 sampai sampel ke 1500. Sedangkan 1500 sisanya, yaitu dari sampel 1501 sampai ke sampel 3000 digunakan untuk proses validasi model yang telah dilatih.

Itersi dibatasi sampai 500 iterasi, sedangkan kriteria minimum adalah 0, perubahan kriteria minimum $1x10^{-7}$, elemen terbesar dalam gradien minimum $1x10^{-4}$ dan perubahan parameter terbesar minimum $1x10^{-3}$. Sedangkan lamda *(λ)* pada metode *L. Marquardt* sebesar 1 dan peluruhan bobotnya adalah 0.

Berikut ini adalah bobot jaringan yang diperoleh dari proses pelatihan. Bobot dari lapis *input* ke lapis tersembunyi *(W1f)*:

Tabel 4.1 Matrik bobot dari lapis *input* ke lapis tersembunyi pada pemodelan sistem

		Bobot dari input ke-				
		1	**2**	**3**	**4**	**Bias**
Neuron lapis tersembunyi ke-	**1**	-4.5075	7.1843	0.015404	0.014393	-2.5563
	2	-4.5341	-0.13498	-0.02494	-0.00741	2.8685
	3	9.0819	-6.9006	0.011952	-0.00608	-1.9411
	4	0.50981	-4.8021	-0.00189	-0.01154	3.0818
	5	-4.181	-1.6173	-0.05878	-0.01287	5.1224
	6	3.2902	3.7557	0.077389	0.018628	-6.3006
	7	-0.56005	-3.7085	-0.01389	-0.01096	4.3229
	8	-8.5938	5.5012	-0.03189	0.001975	2.7541
	9	-1.2408	2.1929	-0.11129	0.002415	1.244
	10	3.6618	-0.80943	0.000568	0.002112	-0.7889

Pada matrik bobot jaringan *W1f* terdapat 10 baris dan 5 kolom matrik. Dapat dilihat bahwa jumlah baris menunjukkan banyaknya jumlah *neuron* pada lapisan tersembunyi, sedangkan jumlah kolom menunjukkan jumlah node pada lapis *input*. Dua kolom pertama adalah bobot dari dua *input* masa lampau, sedangkan dua kolom berikutnya merupakan bobot dari dua *output* masa lampau. Sedangkan satu kolom terakhir adalah bobot dari *bias* ke lapisan tersembunyi.
Berikut ini adalah bobot dari lapis tersembunyi ke lapis *output (W2f)*:

Tabel 4.2 Matrik bobot dari lapis tersembunyi ke lapis *output* pada pemodelan sistem

Kolom 1 sampai kolom 5

-0.16934	-0.06447	0.26314	-0.07755	-0.15186

Kolom 6 sampai kolom 10

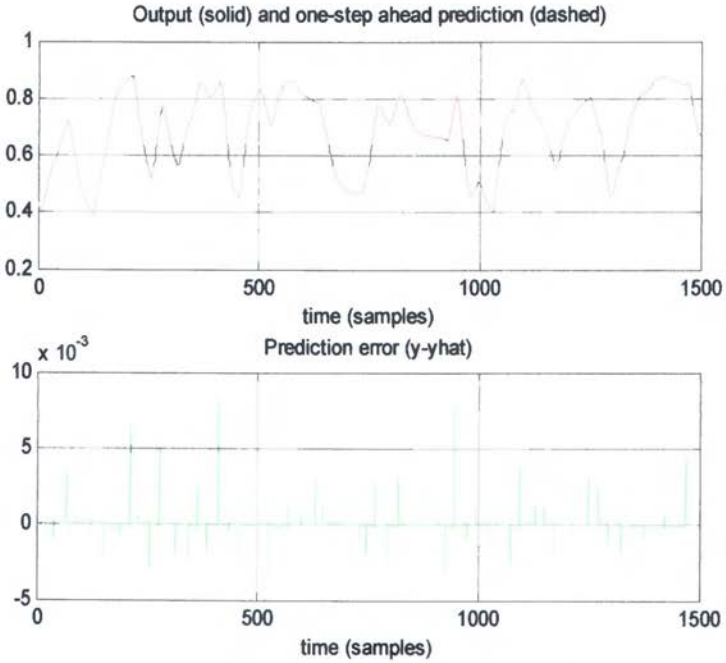
-0.05702	-0.22504	0.2386	-0.0048	0.30256

Kolom 11

0.57092

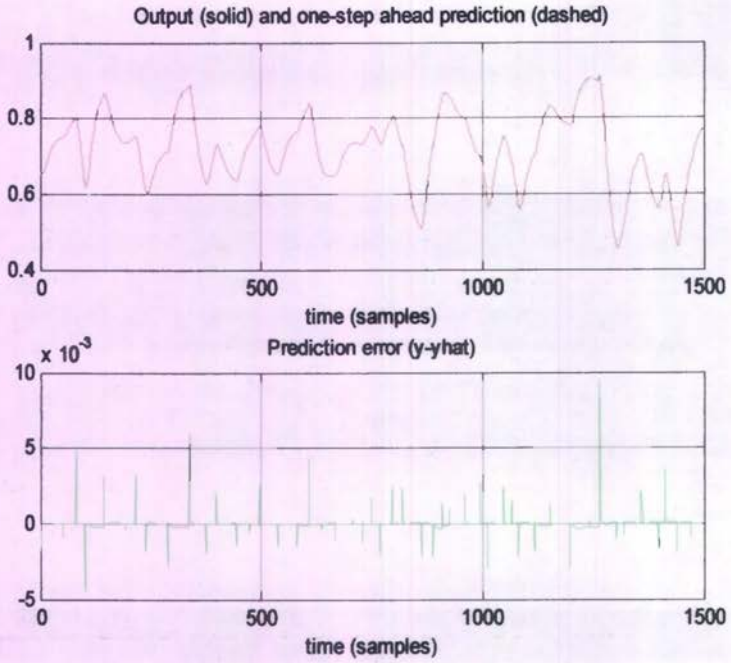
Pada tabel 4.2, terdapat 1 baris matrik dengan 11 kolom. Sama halnya dengan *W1f*, jumlah baris menunjukkan banyaknya *neuron* pada lapis *output*, sedangkan banyaknya kolom menunjukkan jumlah *neuron* pada lapisan sebelumnya (lapisan tersembunyi). Kolom ke-1 sampai ke-10 merupakan bobot dari *neuron* lapisan tersembunyi, sedangkan kolom ke-11 merupakan bobot dari *bias* ke lapisan *output*.

Dengan bobot jaringan pada tabel 4.1 dan tabel 4.2, maka didapatkan *output* prediksi yang hampir mendekati *output* proses dengan masukan yang sama. Berikut ini hasil proses identifikasi sistem dengan menggunakan JST.



Gambar 4.2 *Output* Model dan *Error* Prediksi Pada Proses Pelatihan JST

Pada gambar 4.2 dapat kita amati bahwa model JST dapat mengikuti pola data *output* 1500 sampel pertama dari sistem. Grafik warna merah putus-putus merupakan *output* JST dan biru lurus adalah *output* reaktor *Van Der Vusse*. Dari grafik *prediction error* pelatihan dapat diketahui bahwa *sum square error* pelatihan didapat sebesar 0.0121 dengan *error* tertinggi sebesar 8.35×10^{-3} saat sampel ke 411. *VAV* pelatihan didapat sebesar 99.1719 %.



Gambar 4.3 *Output* Model dan *Error* Prediksi Pada Proses Validasi JST

Validasi dilakukan tanpa pelatihan. Bobot, struktur jaringan dan definisi jaringan yang telah diperoleh dari proses pelatihan digunakan dengan sisa data yang diperleh. Pada gambar 4.3 dapat dilihat bahwa validasi jaringan sudah dapat mengikuti *output* sistem. Garis merah putus-putus merupakan *output* jaringan, dan garis biru lurus merupakan *output* reaktor. Dari grafik *prediction error* pada gambar (10) didapatkan *sum square error* validasi sebesar 0.0098, dimana *error* terbesar terjadi saat sampel ke 1264 sebesar 9.46×10^{-3}. Sedangkan nilai *VAV*-nya sebesar 98.9205%. Dengan validasi lebih besar dari 98%, maka model dapat diterima.

4.3 Pelatihan Kontroller Optimal

Sebelum simulasi proses optimal, maka terlebih dahulu perlu dicari parameter-parameter kontroller JST melalui proses pelatihan kontroller JST dengan algoritma kontrol optimal. Algoritma pelatihan yang digunakan adalah algoritma *recursive Gauss-Newton* dengan menggunakan metode *forgetting factor*. Parameter-parameter jaringan yang didapat melalui proses ini nantinya akan digunakan dalam simulasi proses kontrol optimal.

Bobot awal jaringan diinisialisasi secara acak dengan dimensi yang sama dengan matrik bobot model. Hal itu dilakukan karena sistem sulit mencapai suatu kestabilan apabila inisialisasi bobot dilakukan dengan meng-*inverse* sistem *(general training)*. Tabel 4.3 dan tabel 4.4 menunjukkan bobot awal jaringan untuk proses kontrol.

Tabel 4.3 Matrik bobot awal dari lapis *input* ke lapis tersembunyi untuk pelatihan kontroller

		Bobot dari input ke-				
		1	**2**	**3**	**4**	**Bias**
Neuron lapis tersembunyi ke-	**1**	3.0817	-0.50159	0.10297	0.95261	0.28514
	2	0.93327	0.69898	0.3789	1.1084	0.25229
	3	1.1479	0.34659	0.64885	0.20084	0.027529
	4	2.2145	-0.29436	-0.5259	0.24148	0.24265
	5	0.81025	0.61207	0.88524	0.72236	0.10349
	6	0.73589	0.34924	0.065567	0.24847	0.28746
	7	0.064924	0.51743	0.32929	-0.45287	0.84558
	8	0.46145	0.69476	0.876	0.13139	0.54918
	9	0.52434	0.66834	0.78517	0.72303	0.79236
	10	0.92684	0.70651	0.043365	0.46251	0.15582

Tabel 4.4 Matrik bobot awal dari lapis tersembunyi ke lapis *output* untuk pelatihan kontroller

Kolom 1 sampai 5

3.5864	1.3314	1.125	3.2764	1.0198

Kolom 6 sampai 10

1.0819	-0.93386	0.41828	0.44458	1.067

Kolom 11

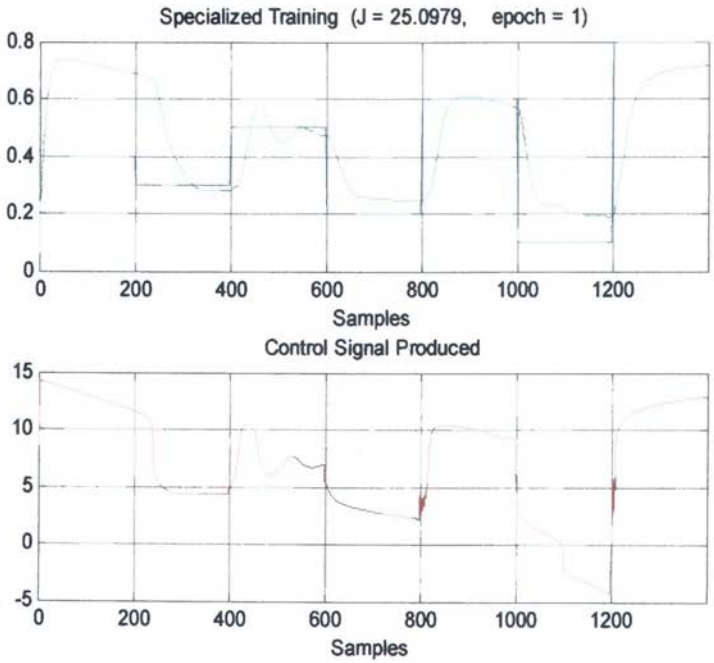
-0.09592

Sama halnya dengan susunan JST untuk proses pemodelan sistem, susunan matrik bobot pada tabel 4.3 dan 4.4 menunjukkan susunan JST yang digunakan. Jumlah baris menunjukkan banyaknya *neuron* dan jumlah kolom menunjukkan banyaknya *input* yang melalui *neuron* tersebut. Pada tabel 4.3 terdapat 10 baris bobot yang berarti bahwa jumlah *neuron* pada lapis tersembunyi berjumlah 10 dan 5 kolom bobot yang berarti bahwa terdapat 5 input yang terdiri atas 4 buah *regressor* dan 1 *bias* yang nilainya adalah 1. *Neuron* pada lapis tersembunyi mempunyai fungsi aktivasi *tangent hyperbolic*. Sedangkan pada tabel 4.4 hanya terdapat satu *neuron* dan 11 *input* dari lapis tersembunyi ke lapis *output*. *Input* tersebut terdiri atas 10 *output* dari lapisan tersembunyi dan 1 *bias* dengan nilai 1. *Neuron* pada lapis *output* mempunyai fungsi aktivasi linier.

Jumlah iterasi yang digunakan untuk mengupdate bobot sebanyak 7 iterasi dengan jumlah sampel sebanyak 1400 sampel. *Time sampling* yang digunakan adalah 0.02 detik dan *penalty factor* (ρ) pada sinyal kontrol sebesar 1×10^{-6}. Parameter pada algoritma *forgetting factor* yang digunakan antara lain λ sebesar 0,999 dan $p^{(0)}$ sebesar 10.

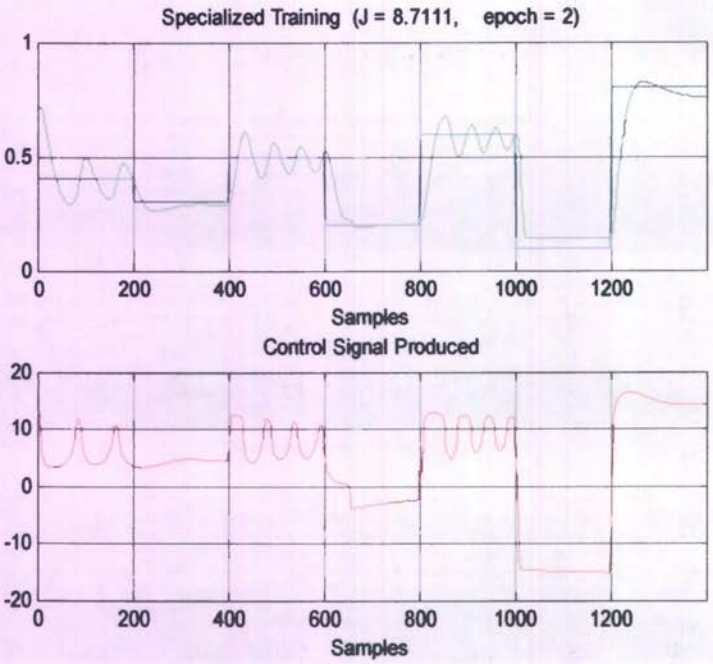
Pada proses pelatihan ini, sinyal kontrol dibebaskan berosilasi sampai dengan tak hingga. Tujuan dari dibebaskannya sinyal kontrol adalah agar sistem mendapatkan bobot sistem

terbaiknya. Berikut ini hasil proses iterasi jaringan dari itersi 1 sampai 9.



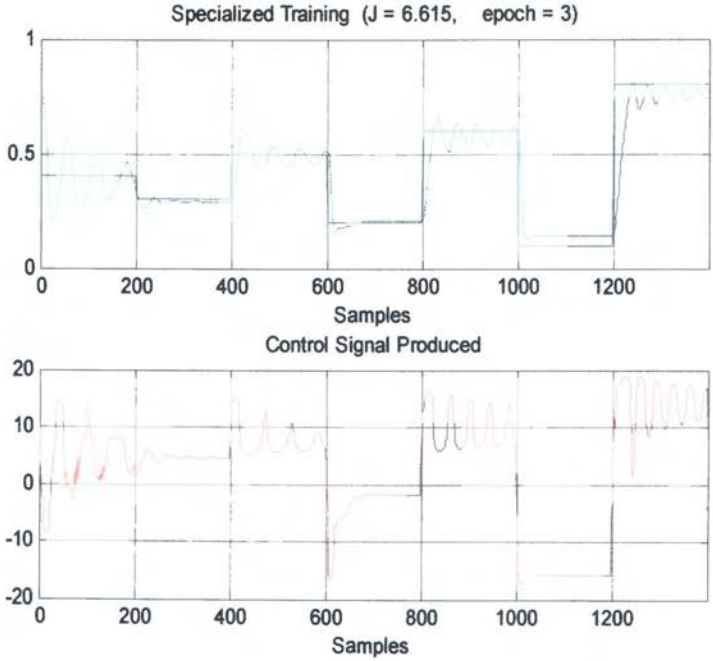
Gambar 4.4 Respon sistem terhadap perubahan *set point* dan sinyal kontrol yang dihasilkan pada iterasi ke-1.

Pada iterasi pertama, terlihat bahwa respon sistem dan sinyal kontrol yang dihasilkan masih jauh dari referensi. Fungsi kriteria *(J)* bernilai 25.0975. Iterasi dilanjutkan dan dibandingkan performansinya.



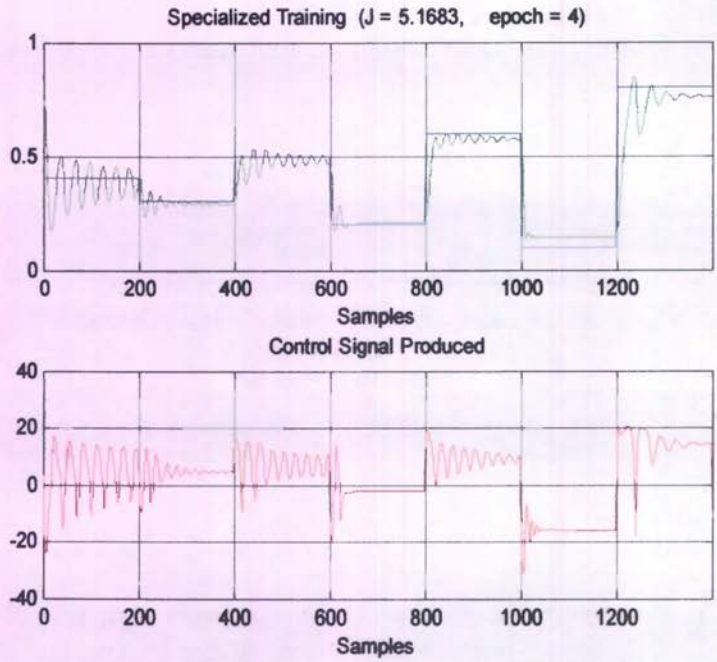
Gambar 4.5 Respon sistem terhadap perubahan *set point* dan sinyal kontrol yang dihasilkan saat iterasi ke-2.

Pada gambar 4.5, dapat dilihat, bahwa saat iterasi ke-2, output sistem mulai mengikuti sinyal referensi, walaupun masih sangat kasar dan dengan *error steady state* yang besar. Sinyal kontrol juga mulai menyesuaikan dengan output dari sistem. Fungsi kriterianya *(J)* bernilai 8.7111. Terlihat disini bahwa nilai J sudah mulai berkurang.



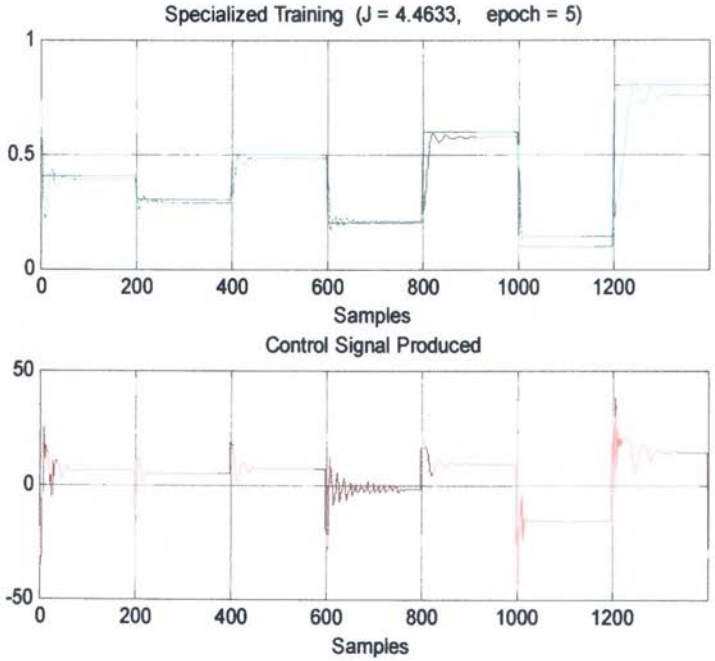
Gambar 4.6 Respon sistem terhadap perubahan *set point* dan sinyal kontrol yang dihasilkan saat iterasi ke-3

Saat iterasi ke-3, didapatkan nilai fungsi kriteria *(J)* lebih kecil dari pada saat iterasi ke-2 yang bernilai 6.615. Hal itu ditunjukkan oleh respon sistem yang lebih baik terhadap referensi yang telah ditentukan. Begitu pula dengan sinyal kontrol yang dihasilkannya.



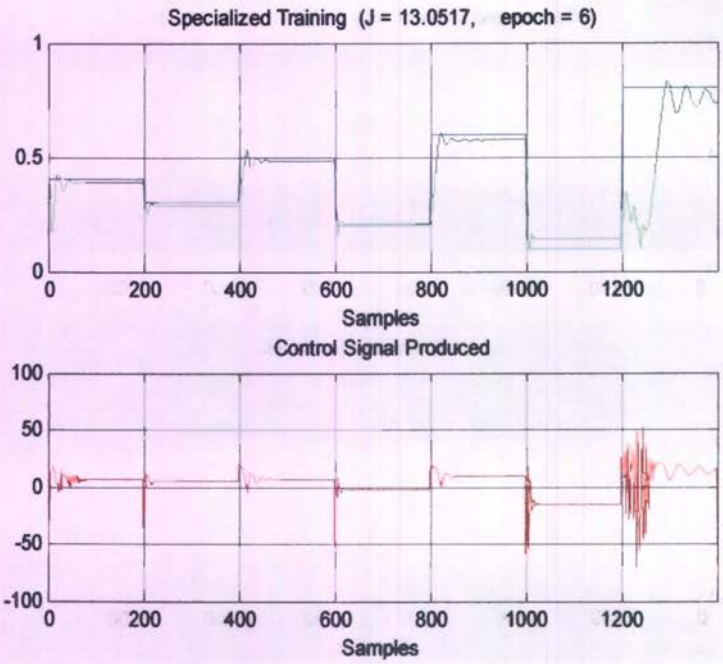
Gambar 4.7 Respon sistem terhadap perubahan *set point* dan sinyal kontrol yang dihasilkan saat iterasi ke-4

Saat iterasi ke-4, didapatkan fungsi kriteria *(J)* sebesar 5.1683, lebih baik saat iterasi sebelumnya. Output sistem pada beberapa set point terlihat sudah mulai stabil, walaupun di referensi yang lain masih terdapat osilasi, terutama disaat-saat awal pada referensi 0.4 mol/liter. Sinyal kontrol yang dihasilkan juga menyesuaikan dengan output sistem, dimana osilasi paling sering terjadi saat 200 sampel pertama.



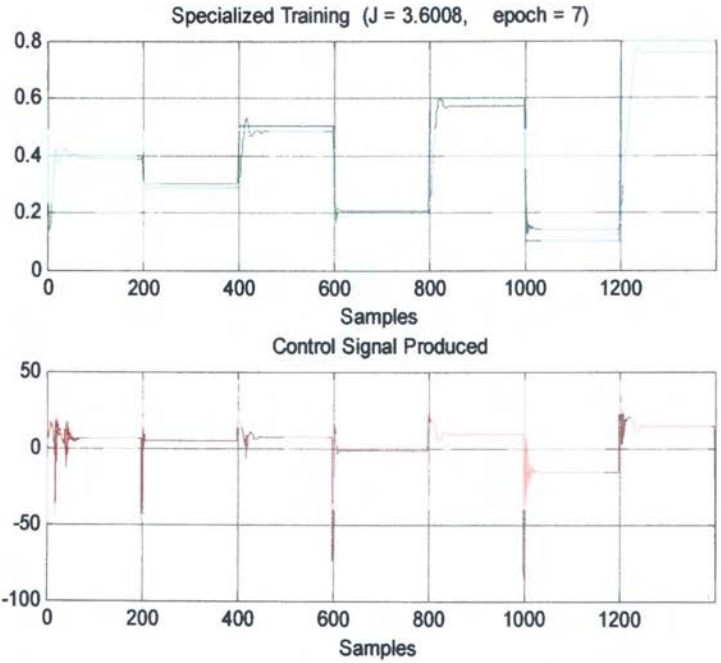
Gambar 4.8 Respon sistem terhadap perubahan *set point* dan sinyal kontrol yang dihasilkan saat iterasi ke-5

Saat itersai ke-5, output sistem hampir sempurna. Nilai J yang didapat sebesar 4.4633. Output sistem sudah dapat mengikuti referensi, walaupun masih terdapat error steady state. Namun output saat iterasi ke-5 masih terdapat osilasi pada tiap perubahan referensi. Begitu pula dengan sinyal kontrol, terutama saat referensi ke 0.2 mol/liter, dimana osilasi terjadi terus-menerus.



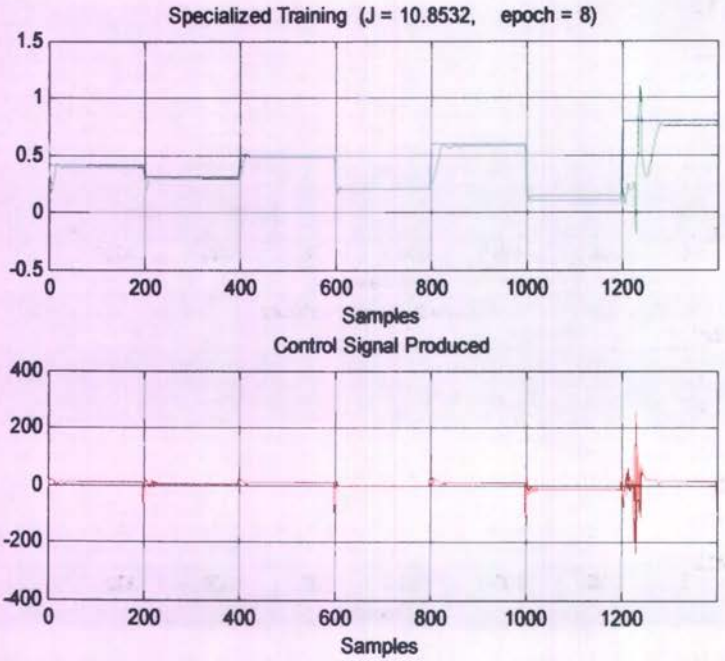
Gambar 4.9 Respon sistem terhadap perubahan *set point* dan sinyal kontrol yang dihasilkan saat iterasi ke-6

Saat iterasi ke-6, ternyata output sistem lebih buruk daripada itersai sebelumnya. Hal itu dapat dilihat saat referensi 0.8 mol/liter, dimana output sistem tidak dapat mengikuti referensi, walaupun untuk referensi yang lain responnya lebih baik. Sinyal kontrol saat itu juga sangat berosilasi. Nilai J-nya lebih besar, yaitu bernilai 13.0517.



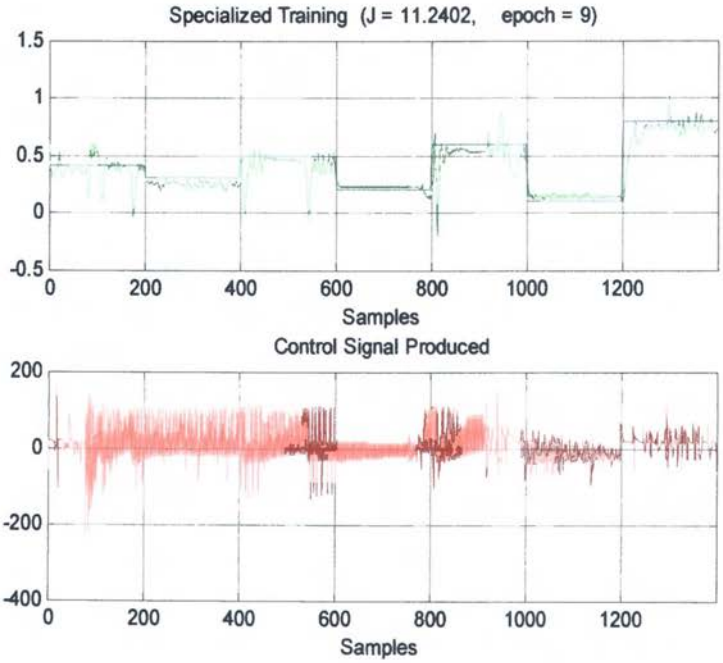
Gambar 4.10 Respon sistem terhadap perubahan *set point* dan sinyal kontrol yang dihasilkan saat iterasi ke-7

Pada iterasi ke-7 didapatkan nilai J paling minimal, yaitu sebesar 3.6008. Respon sistemnya pun paling baik bila dibandingkan dengan saat iterasi-iterasi sebelumnya. Begitu pula dengan sinyal kontrol yang dihasilkan, dimana osilasi sudah jauh berkurang. Untuk memastikan bahwa saat iterasi ke-7 menghasilkan performa terbaik, maka dilakukan iterasi selanjutnya.



Gambar 4.11 Respon sistem terhadap perubahan *set point* dan sinyal kontrol yang dihasilkan saat iterasi ke-8

Pada iterasi ke-8 dihasilkan performa yang lebih buruk. Nilai J yang didapat sebesar 10.8532, lebih besar dari pada iterasi ke-7. Hal itu dapat dilihat dari respon sistem, terutama saat referensi sebesar 0.8 mol/liter. Sinyal kontrol saat itu juga mengalami osilasi yang lebih sering.



Gambar 4.12 Respon sistem terhadap perubahan *set point* dan sinyal kontrol yang dihasilkan saat iterasi ke-9

Saat iterasi ke-9, didapatkan respon paling buruk, dimana output sistem semakin berosilasi pada setiap set point-nya. Begitu pula dengan sinyal kontrol yang dihasilkan, sangat berosilasi. Nilai J yang didapat sebesar 11.2402.

Dari grafik performa sistem yang dihasilkan setiap iterasi, maka diambil iterasi saat J paling minimal, yaitu saat iterasi ke-7, dengan J bernilai 3.6008. Tabel 4.5 dan 4.6 berikut menunjukkan bobot jaringan yang telah dilatih dengan algoritma kontrol optimal saat iterasi ke-7.

Tabel 4.5 Matrik bobot lapis *input* ke lapis tersembunyi pada jaringan kontroller setelah pelatihan

		Bobot dari input ke-				
		1	2	3	4	Bias
Neuron lapis tersembunyi ke-	1	16.734	1.264	1.8568	-51.83	11.58
	2	-4.3114	-2.083	-2.2281	-43.821	-7.1565
	3	0.60705	-1.381	-1.081	7.4609	-2.4877
	4	11.469	-23.712	12.484	-0.00468	0.65392
	5	2.3155	-0.96699	-0.67271	-47.288	0.65201
	6	1.1602	-0.80246	-1.0931	20.288	-2.3964
	7	-12.077	8.8797	9.8626	-3.8409	4.888
	8	-1.7584	0.60002	0.75473	46.22	-1.2955
	9	0.91589	0.021108	0.14577	-4.8326	0.61755
	10	1.8591	-0.26516	-0.92362	9.2061	-4.654

Tabel 4.6 Matrik bobot lapis tersembunyi ke lapis *output* pada jaringan kontroller setelah pelatihan

Kolom 1 sampai 5

24.23	-3.7744	6.4306	60.661	4.8295

Kolom 6 sampai 10

3.702	1.9377	0.317	5.1039	20.161

Kolom 11

-35.289

Tidak ada perubahan pada dimensi matrik bobot hasil dari pelatihan dengan algoritma kontrol optimal sebelum dan sesudah pelatihan. Masing-masing bobot hanya diupdate berdasarkan pada algoritma kontrol optimal.

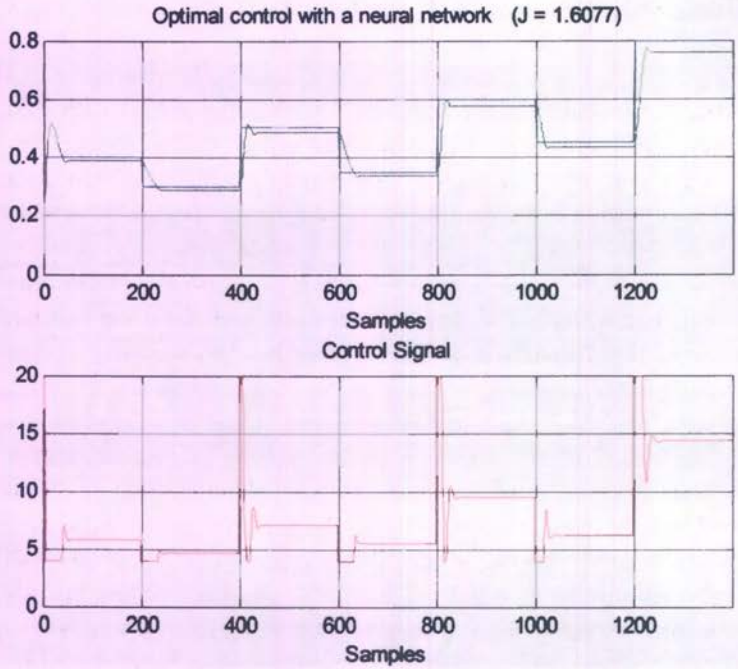
4.4 Simulasi Kontroller Optimal

Setelah kontroller JST dilatih dengan algoritma kontrol optimal, maka didapatkan parameter-parameter dari kontroller. Parameter-parameter tersebut antara lain adalah:

- Susunan *regressor*, yang merupakan jumlah susunan data *input* jaringan. Susunan *regressor* yang diperoleh adalah [2 2 1].
- *Network Deffinition*, yang menunjukkan struktur JST yang digunakan dalam kontroller. Struktur JST meliputi jumlah lapis tersembunyi dan lapis *output* beserta masing-masing fungsi aktivasinya. Struktur JST tersebut adalah 1 lapis tersembunnyi dengan 10 *neuron* dan memiliki fungsi aktivasi *tangent hyperbolic* dan 1 lapis *output* dengan 1 *neuron* yang memiliki fungsi aktivasi linier.
- Bobot masing-masing lapisan. Terdiri atas susunan matrik bobot dari lapis *input* ke lapis tersembunyi dan dari lapis tersembunyi ke lapis *output*. Susunan bobot untuk kontroller dapat dilihat pada tabel 4.5 dan tabel 4.6.

Pada proses simulasi kontroller ini, tidak dilakukan proses iterasi karena bobot jaringan dari masing-masing lapis *(W1c* dan *W2c)* sudah didapatkan melalui proses pelatihan sebelumnya. Sinyal kontrol yang dihasilkan dibatasi antara 4-20 mA. Hal itu dilakukan karena dalam prakteknya sinyal kontrol standar untuk instrument elektronik adalah 4-20 mA. Simulasi dilakukan dengan sampling time 0.02 detik.

Gambar 4.13 menunjukkan performasi sistem dan sinyal kontrol yang dihasilkan.



Gambar 4.13 Performasi sistem dan sinyal kontrol yang dihasilkan dari simulasi proses kontrol optimal

Dari gambar 4.13 dapat kita lihat bahwa respon sistem dapat mengikuti referensi atau *set point*. Walaupun masih terdapat *maximum overshoot*, *settling time* yang realtif lama, *error steady state* atau *offset*, namun secara keseluruhan sistem dapat dibilang stabil. Sinyal kontrol yang dihasilkan lebih realistis, karena terbatas pada 4-20 mA dengan osilasi yang tidak terlalu lama dan frekuensi yang tidak terlalu tinggi. Dari simulasi kontrol optimal didapatkan nilai fungsi kriteria *(J)* sebesar 1.6077.

Untuk mengetahui performansi sistem pada beberapa perubahan set point, maka dilakukan tracking set point naik dan turun setiap 200 sampel pada nilai 0.4 *mol/liter*, 0.3 *mol/liter*, 0.5

mol/liter, 0.35 *mol/liter*, 0.6 *mol/liter*, 0.45 *mol/liter* dan 0.8 *mol/liter*.

Pada 200 sampel pertama, *set point* di set pada 0.4 *mol/liter*. Sistem mengalami *maximum overshoot* pada sampel ke-16 dengan amplitudo sebesar 0.514 *mol/liter*. Sistem *steady* saat amplitudonya mencapai 0.386 *mol/liter* saat sampel ke 60. *Offset* sistem sebesar 4 %. Sinyal kontrol yang dihasilkan pada *set point* 0.4 *mol/liter* mencapai minimum pada sampel ke 5 yang berada pada 4 mA dan mencapai maksimum pada sampael ke 0 dengan nilai 20 mA. Selanjutnya sinyal kontrol akan stabil pada 5.835 mA saat sampel ke 60.

Ketika *set point* dirubah ke 0.3 *mol/liter* pada saat sampel ke 200, sistem tidak mengalami *maximum*. Sistem langsung stabil pada 0.288 *mol/liter* saat sampel ke 244. *Offset* system sebesar 4 %. Sinyal kontrol mencapai minimum sebesar 4 mA saat sampel ke 200, selanjutnya tanpa banyak mengalami osilasi, sinyal kontrol mencapai maksimum pada nilai 4.835 mA saat sampel ke 237. Sinyal kontrol mengalami kondisi stabil pada saat sampel ke 244 sebesar 4.58 mA.

Saat *set point* dinaikkan sampai pada 0.5 *mol/liter* saat sampel ke 400, respon sistem mulai beranjak naik dan akan mencapai *maximum overshoot* pada nilai 0.514 *mol/liter* saat sampel ke 416. Tanpa banyak berosilasi sistem akan menuju kestabilan saat sampel ke 450 pada nilai 0.484 *mol/liter*. *Offset* pada *set point* 0.5 *mol/liter* sebesar 3.2 %. Sementara itu sinyal kontrol maksimum mencapai 20 mA saat sampel ke 400 dan minimum sebesar 4 mA saat sampel 411. Sinyal kontrol akan mencapai stabilitas pada nilai 7.501 mA saat sampel ke 450.

Ketika *set point* diturunkan menjadi 0.35 *mol/liter* saat sampel ke 600, sistem mencapai respon minimum pada 0.335 *mol/liter* saat sampel ke 636. Tanpa banyak berosilasi, stabil saat

sampel ke 650 dengan nilai 0.336 *mol/liter*. *Offset* sistem saat *set point* 0.35 *mol/liter* sebesar 4 %. Sinyal kontrol saat perubahan *set point* terjadi mencapai nilai minimumnya, yaitu sebesar 4 mA saat sampel ke 600. Sinyal kontrol tertinggi terjadi saat sampel ke 635, dengan nilai sebesar 6.213 mA dan stabil pada nilai 5.528 mA saat sampel ke 650. *IAE* sistem sebesar 4.3519.

Sistem masih stabil saat *set point* dinaikkan pada nilai 0.6 *mol/liter* saat sampel ke 800. Respon tertinggi terjadi saat sampel ke 818 dengan nilai sebesar 0.5965 *mol/liter*. Sistem stabil saat sampel 848 dengan konsentrasi konstan sebesar 0.577 *mol/liter*. *Offset* sistem saat *set point* 0.6 *mol/liter* sebesar 3.8 %. Sinyal kontrol mencapai nilai maksimum saat sampel ke 800 pada nilai 20 mA dan minimum saat sampel ke 815 sebesar 4 mA. Sinyal kontrol stabil pada level 9.51 mA saat sampel ke 848.

Penurunan *set point* secara tajam terjadi saat sampel ke 1000. Disini *set point* diturunkan dengan tajam ke 0.45 *mol/liter*. Respon terendah terjadi saat sampel ke 1023, dengan konsentrasi 0.4395 *mol/liter*. System *steady* pada nilai 0.436 saat sampel ke 1040. *Offset* sistem mencapai 3.11%. Sementara itu, sinyal kontrol mencapai minimum dengan nilai 4 mA saat sampel ke 1000 dan maksimum saat 1021 pada 8.39 mA. Sinyal kontrol stabil pada 6.253 mA saat sampel ke 1040.

Set point dinaikkan dengan tajam ke 0.8 *mol/liter* saat sampel ke 1200. Sistem mencapai respon tertinnginya saat sampel ke 1225 dengan konsentrasi 0.774 mol/detik. Sistem stabil pada sampel 1250 saat konsentrasinya 0.76 mol/detik. *Offset* sistem mencapai 5%. Sementara itu sinyal kontrol mencapai maksimum saat sampel ke 1200 pada 20 mA dan minimum saat sampel ke 1224 pada nilai 10.8 mA. Sinyal kontrol stabil pada sampel 1250 pada nilai 14.306 mA.

Seperti terlihat pada tabel 4.9 dan tabel 4.10, sistem stabil pada semua *set point* dengan *offset* yang bervariasi. Tanpa banyak mengalami osilasi, sistem akan langsung menuju kestabilan dengan *settling time* yang relatif cepat. setelah mengalami respon maksimal dan minimal yang tidak terlalu ekstrim. Sifat *non-minimum phase* yang melekat pada reaktor secara otomatis dihilangkan oleh kontroller JST optimal. Begitu pula dengan sinyal kontrol yang dihasilkan oleh kontroller optimal JST. Tanpa banyak mengalami osilasi, sinyal kontrol juga langsung menuju kestabilan. Maksimum dan minimum sinyal kontrol tercapai hanya saat terjadi perubahan *set point*.

Tabel 4.7 Performansi sistem pada setiap *set point* pada proses pelatihan

Set point	Respon Max/Min		*Settling Time*		*Offse Steady*
mol/liter	Amplitudo	Sampel ke	Amplitudo	Sampel ke	*State*
0.4	0.138	5	0.386	155	3.50%
0.3	0.277	208	0.2885	237	3.83%
0.5	0.5314	417	0.482	470	3.60%
0.2	0.16	604	0.205	645	2.50%
0.6	0.6014	821	0.5725	860	4.58%
0.1	0.123	1005	0.14	1038	40%
0.8	0.778	1236	0.76	1260	5%

Tabel 4.8 Sinyal kontrol yang dihasilkan pada setiap *set point* pada proses pelatihan

Set point	Sinyal kontrol maksimal		Sinyal kontrol minimal		Sinyal kontrol stabil	
mol/liter	Amplitudo	Sampel ke	Amplitudo	Sampel ke	Amplitudo	Sampel ke
0.4	32.05	2	-72.8	0	5.83	90
0.3	19.1	200	-36.8	201	4.617	220
0.5	18.08	400	-7.12	417	7.01	470
0.2	16.25	602	-60.5	600	-1.88	637
0.6	23.85	800	4.225	821	9.38	860
0.1	7.89	1005	-87.34	1000	-15.65	1040
0.8	27.02	1204	1.85	1202	14.31	1270

Tabel 4.9 Performansi sistem pada setiap *set point* saat simulasi kontrol

Set point	Respon Max/Min		*Settling Time*		*Offset*
mol/liter	Amplitudo	Sampel ke	Amplitudo	Sampel ke	
0.4	0.514	16	0.386	60	3.50%
0.3	-	-	0.288	244	4.00%
0.5	0.514	416	0.484	450	3.20%
0.35	0.335	636	0.336	650	4.00%
0.6	0.5965	818	0.577	848	3.80%
0.45	0.4395	1023	0.436	1040	3.11%
0.8	0.774	1225	0.76	1250	5%

Tabel 4.10 Sinyal kontrol yang dihasilkan pada setiap *set point* saat simulasi kontrol

Set point	Sinyal kontrol maksimal		Sinyal kontrol minimal		Sinyal kontrol stabil	
mol/liter	Amplitudo	Sampel ke	Amplitudo	Sampel ke	Amplitudo	Sampel ke
0.4	20	0	4	5	5.835	60
0.3	4.835	237	4	200	4.58	244
0.5	20	400	4	411	7.501	450
0.35	6.213	635	4	600	5.528	650
0.6	20	800	4	815	9.51	848
0.45	8.39	1021	4	1000	6.253	1040
0.8	20	1200	10.8	1224	14.306	1250

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil dan analisa dapat ditarik beberapa kesimpulan, antara lain:

1. Telah berhasil dikembangkan sebuah metode kontrol optimal berbasis jaringan syaraf tiruan (JST) pada reaktor *Van Der Vusse*.
2. Secara umum, respon dari sistem dapat mengikuti *set-point*, walaupun masih terdapat *error steady state*.
3. Sistem stabil pada semua set-point, 0.4 *mol/liter*, *0.3 mol/liter*, 0.5 *mol/liter*, *0.35 mol/liter*, 0.6 *mol/liter*, 0.45 *mol/liter* dan 0.8 *mol/liter* dengan masing-masing *offset steady state* sebesar 3.5%, 4%, 3.2%, 4%, 3.8%, 3.11% dan 5%.

5.2 Saran

Untuk pengembangan tugas akhir ini, penulis menyarankan beberapa hal, antara lain:

1. Mengapilkasikan *penalty factor* pada kuadrat sinyal kontrol yang berbeda ($u(t) - u(t-1)$) untuk memeperbaiki respon sistem.
2. Mengaplikasikan metode pembelajaran yang lain, terutama metode pembelajaran untuk jaringan pada kontroller dengan metode *Constant Trace (CT)* atau *Exponential Forgetting Factor and Resetting Algorithm (EFRA)*.

DAFTAR PUSTAKA

Nørgraad, Magnus. Ravn, O. Poulsen, NK. Hansen, LK, " **Neural Network for Modelling and Control of Dynamic Systems"**, Verlag Springer, London: 2000.

Bequette, Wayne B, " **Process Dynamics: Modelling, Analysis and Simulation"**, Prentice Hall, New Jersey, USA.

Lelono, Bambang W, **"Handout Ajar Jaringan Syaraf Tiruan",** Jurusan Teknik Fisika, ITS: 2005.

Antonelly, Rita. Astolfi, Alessandro, **"Continous Stired Tank Reactor: Easy to Stabilise?"**, Automatica, www.sciencedirect.com, 2003.

Conradie, Alex Van Eck, **"Neurocontroller Development for Non-Linier Process Utilising Evolutionary Reinforcement Learning",** Chemical Engineering University of Stellenbosch, South Africa: 2000.

Nørgraad, Magnus, **"Neural Network Based Control System Design TOOLKIT, Version 2"**, Department of Automation, Department of Mathematical Modelling, Denmark Technical University, Denmark:2000.

Kashiwagi, Hiroshi dan Li, Yun, **"Non Parametric Non-linear Model Predictive Control (NMPC)",** Korean Journal, Chemical Engineering, Seoul: 2004.

//LISTING PROGRAM UNTUK PENGAMBILAN DATA
//MATLAB 7.0.1

```
%---------------------- AMBIL DATA ----------------------
% Program untuk membangkitkan data-input output pada reaktor
% Van Der Vusse
% Persamaan dinamik reaktor didefinisikan dalam 'persamaan'
% Persamaan output reaktor didefinisikan dalam 'vdv_out'
%--------------------------------------------------------
% Deklarasi parameter
clear all
close all
clc
disp('PEMBANGKITAN DATA SEDANG BERLANGSUNG.....')
disp('Tekan ctrl + C untuk menghentikan')
global ugl
fighandle=progress;
param.sampling = 0.001;
param.length = 3;
[waktu,u_data]=a_p_r_b_s(param.sampling,param.length,...
                         4,20,20,80);
time_sampling = param.sampling;
panjang_sinyal = param.length;
samples = floor(panjang_sinyal/time_sampling);
if (length(u_data)~=samples),
   error('Dimensi data dan sample tidak sama')
end
mat_model = 'persamaan';
model_out = 'vdv_out';
x0 = [0.5;0.5];
t = -time_sampling;
for i = 1:samples,
   t = t + time_sampling;
   ugl = u_data(i);
   [time,x] = ode45(mat_model,[t-time_sampling t],x0);
   x0 = x(length(time),:);
   eval(['y = ' model_out '(x0);']);
   y_data(i) = y;
progress(fighandle,floor(100*i/samples));
end
y_data = y_data';
subplot(211)
stairs(u_data),grid
title('Data Input Reaktor Van der Vusse')
subplot(212)
plot(y_data),grid
title('Data Output Reaktor Van der Vusse')
clc
disp('Proses pembangkitan data selesai')
```

//LISTING PROGRAM PERSAMAAN INPUT REAKTOR VAN
//DER VUSSE
//MATLAB 7.0.1

```
%----------------- Reaktor Van Der Vusse -------------------
% Program untuk mendefinisikan persamaan dinamik reaktor Van
% Der Vusse
% Dimana: F/V   = ugl
%         Ca    = x1 mol/liter
%         Cb    = x2 mol/liter
%         Caf   = 10 mol/liter
%         k1    = 50
%         k2    = 100
%         k3    = 10
%-------------------------------------------------------------
function dx = persamaan(t,x)
global ugl;
%inisialisasi paramter
x1 = x(1);
x2 = x(2);
%reaksi van der vusse
dx1 = (-50 * x1) - ( 10 * x1 * x1) + ( ugl * (10 - x1));
dx2 = (50 * x1) - (100 * x2) + (ugl * (-x2));
%memasukan nilai
dx = [dx1;dx2];
```

//LISTING PROGRAM PERSAMAAN OUTPUT REAKTOR VAN
//DER VUSSE
//MATLAB 7.0.1

```
% Fungsi yang digunakan untuk mendeklarasikan output dari
% reaktor
% Van Der Vusse, dimana konsentrasi produk B adalah x(2)
function y = vdv_out(x);
 y = x(2);
```

//LISTING PROGRAM PEMODELAN REAKTOR DENGAN JST //MATLAB 7.0.1

```
% ----------------- Pemodelan Reaktor ----------------------
% Program untuk mendapatkan model dari suatu sistem dengan
% Jaringan Syaraf
% Tiruan. Algoritma yang digunakan adalah pembelajaran orde
% dua L. Marquat
% ----------------------------------------------------------
close all
clc
% Membaca input/output data dari experimen
load data_proces_1
u1 = (u_data(1:1500))'; % data input pelatihan dari u_data
u2 = (u_data(1501:3000))';% data input validasi dari u_data
y1 = (y_data(1:1500))'; % data output pelatihan dari y_data
y2 = (y_data(1501:3000))';% data output validasi dari y_data

% plot grafik untuk setiap data
subplot(211), plot(u1)  % plot grafik untuk u1
title('Space Velocity (F/V)')
grid;
subplot(212), plot(y1)  % plot grafik untuk y1
title('Konsentrasi Cyclopentenol (Produk B)')
grid;
%subplot(111)
disp('Tekan sembarang tombol untuk melanjutkan!!!')
pause
close all
clc

disp('Simulasi proses sedang berlangsung........')
% Data pelatihan di-scaling
[u1s,uscales] = dscale(u1);
[y1s,yscales] = dscale(y1);
% Data validasi di-scaling
u2s = dscale(u2,uscales);
y2s = dscale(y2,yscales);


% Pemilihan struktur model non-linier
NetDeff = ['HHHHHHHHHH';'L---------'];
NN = [2 2 1];

% Pelatihan jaringan
trparms = settrain;
[W1,W2,NSSEvec,iteration,lambda]=nnarx(NetDeff,NN,[],[],...
                                       trparms,y1s,u1s);
```

```
% Validasi dari jaringan yang telah dilatih
[W1f,W2f] = wrescale('nnarx',W1,W2,uscales,yscales,NN)
% mengembalikan bobot ke nilai semula
[yhat1,NSSE] = nnvalid('nnarx',NetDeff,NN,W1f,W2f,y1,u1);
[yhat2,NSSE] = nnvalid('nnarx',NetDeff,NN,W1f,W2f,y2,u2);


% Melihat RMSE dan validitas dari data peltihan dan validasi
Vav_train = vaf((y1(1:1498)),yhat1);
error_training = r_m_s_e((y1(1:1498)),yhat1)
Vav_training = mean(Vav_train)
Vav_valid = vaf((y2(1:1498)),yhat2);
error_validation = r_m_s_e((y2(1:1498)),yhat2)
Vav_validation = mean(Vav_valid)
```

//LISTING PROGRAM ALGORITMA NNARX
//MATLAB 7.0.1

```
% >>>>>>>>>>>>>>>  NNARX   <<<<<<<<<<<<<<<<
% Diadopsi dari M. Norgaard dengan modifikasi diperlukan
% ---------------------------------------------------------
function [W1,W2,PI_vector,iteration,lambda]=nnarx…
(NetDef,NN,W1,W2,trparms,Y,U)
% >>>>>>>>>>>>>>>   INISIALISASI   <<<<<<<<<<<<<<<<
N  = length(Y);
na = NN(1);
  [nu,N]       = size(U);
  nb = NN(2:1+nu);                    % model nnarx
  nk = NN(2+nu:1+2*nu);

nmax       = max([na,nb+nk-1]);
nab          = na+sum(nb);

% -- Inisialisasi bobot jika diperlukan --
if isempty(W1)| isempty(W2),
  hidden = length(NetDef(1,:));  % Jumlah dari syaraf
                                 % lap.tersembunyi
  W1 = rand(hidden,nab+1)-0.5;
  W2 = rand(1,hidden+1)-0.5;
end

% -- Inisialisasi 'trparms' jika diperlukan --
if isempty(trparms), trparms=[]; end

% >>>>>>MEMBANGUN REGRESI MATRIK PHI   <<<<<<<<<<<
PHI = zeros(nab,N-nmax);
jj  = nmax+1:N;
for k = 1:na, PHI(k,:)     = Y(jj-k); end
```

```
index = na;
for kk = 1:nu,
  for k = 1:nb(kk), PHI(k+index,:) = U(kk,jj-k-nk(kk)+1);
end
  index = index + nb(kk);
end

% >>>>>     MEMANGGIL FUNGSI PELATIHAN          <<<<<<<<<<<
[W1,W2,PI_vector,iteration,lambda]=marq(NetDef,W1,W2,PHI,...
                                    Y(nmax+1:N),trparms);
```

//LISTING PROGRAM NNVALID
//MATLAB 7.0.1

```
function [Yhat,PI]= nnvalid(method,NetDef,NN,...
                    W1,W2,par1,par2,par3)
%=====================================================
%  NNVALID
%  -------
%  Diadopsi dari Magnus Norgaard, IAU/IMM, dengan perubahan
%  yang diperlukan
%=====================================================

% >>>>>>      MENDAPATKAN PARAMETER-PARAMETER     <<<<<<
  skip = 1;
  Y=par1;
  if nargin>6, U=par2; end

% >>>>>>>>>>>>>>>>>  INISIALISASI <<<<<<<<<<<<<<<<<<<<
Ndat     = length(Y);          % # data
na = NN(1);

% ---------- Model NNARX  ----------
% nnarx atau nnoe model
    [nu,N] = size(U);
    nb = NN(2:1+nu);
    nk = NN(2+nu:1+2*nu);
    nc = 0;
    [nu,Ndat] = size(U);
    nb      = NN(:,2:1+nu);           % nnarx model
    nk      = NN(:,2+nu:1+2*nu);
    if nu>1,
      nab  = na + sum(nb')';
    else
      nab     = na+nb;
    end
```

```
% --------- Inisialisasi umum --------
  nmax = max([na,nb+nk-1,nc])% Sinyal 'paling tua' yang
                          % digunakan sebagai input pada model
  nab  = na+sum(nb);                % na+nb
  nabc = nab+nc;                     % na+nb+nc
  outputs = 1; % Hanya model MISO models dipertimbangkan

N = Ndat - nmax;   % Ukuran dari data pelatihan
L_hidden = find(NetDef(1,:)=='L')'; % Letak dari syaraf
                                    % linier pada lap.tersembunyi
H_hidden = find(NetDef(1,:)=='H')';% Letak dari syaraf tanh
                                    % pada lap.tersemunyi
L_output = find(NetDef(2,:)=='L')';% Letak dari syaraf
                                    % linier pada output
H_output = find(NetDef(2,:)=='H')';% Letak dari syaraf tanh
                                    % pada output
[hidden,inputs] = size(W1);
inputs          = inputs-1;
E        = zeros(outputs,N);
y1       = zeros(hidden,N);
Yhat     = zeros(outputs,N);

% >>>>>>>>  MENYUSUN REGRESSI MATRIK PHI   <<<<<<<<<<<
PHI = zeros(sum(nab),N);
jj  = nmax+1:Ndat;
index = 0;
for o=1:outputs,
  for k = 1:na(o), PHI(k+index,:)     = Y(o,jj-k); end
  index = index+na(o);
  for kk = 1:nu,
    for k = 1:nb(o,kk), PHI(k+index,:) = U(kk,jj-k-
nk(o,kk)+1); end
    index = index + nb(o,kk);
  end
end

 % >>>>>>>>>   MENGHITUNG OUTPUT JARINGAN   <<<<<<<<<<
% ----------Model NNARX  ----------
  Y  = Y(:,nmax+1:Ndat);
  h1 = W1*[PHI;ones(1,N)];
  y1(H_hidden,:) = pmntanh(h1(H_hidden,:));
  y1(L_hidden,:) = h1(L_hidden,:);

  h2 = W2*[y1;ones(1,N)];
  Yhat(H_output,:) = pmntanh(h2(H_output,:));
  Yhat(L_output,:) = h2(L_output,:);

  E = Y - Yhat; % Error antara Y dan bagian deterministik
  SSE      = sum(sum(E.*E));% Sum of squared errors (SSE)
  PI       = SSE/(2*N);      % Indek performansi
```

```
 % >>>>>>>>>>>>>>      PLOT HASIL      <<<<<<<<<<<<<<<<<<<
si=figure-1;

% ---------- Output, Prediksi dan Error prediksi ----------
for ii=1:outputs,
 figure(si+ii)
 subplot(211)
 plot(Y(ii,:),'b-'); hold on
 plot(Yhat(ii,:),'r--');hold off
 xlabel('time (samples)')
 if outputs==1,
   title('Output (solid) and one-step ahead prediction
        (dashed)')
 else
   title(['Output (solid) and one-step ahead prediction
         (dashed) (output # ' num2str(ii) ')']);
 end
 grid

 subplot(212)
 plot(E(ii,:),'g');
 title('Prediction error (y-yhat)')
 xlabel('time (samples)')
 grid
 subplot(111)
 drawnow
end

% --------- Fungsi Korelasi ----------
for ii=1:outputs,
  figure(si+outputs+ii)
  subplot(nu+1,1,1);
  M=min(25,N-1);
  Eauto=crossco(E(ii,:),E(ii,:),M);
  Eauto=Eauto(M+1:2*M+1);
  conf=1.96/sqrt(N);
  plot([0:M],Eauto(1:M+1),'b-'); hold on
  plot([0 M],[conf -conf;conf -conf],'r--');hold off
  set(gca,'xlim',[0 M]);
  xlabel('lag')
  if outputs==1
    title('Auto-correlation function of prediction error')
  else
    title(['Autocorrelation coefficients for prediction
         Error (output # ' num2str(ii) ')']);
  end
  grid

  for i=1:nu,
```

```
    subplot(nu+1,1,i+1);
    UEcross=crossco(E(ii,:),U(i,1:N),M);

    plot([-M:M], UEcross,'b-'); hold on
    plot([-M M],[conf -conf;conf -conf],'r--');hold off
    xlabel('lag')
    title(['Cross-correlation coef. of u' num2str(i) ' and
           prediction error'])
    ymax=min(5*conf,max([abs(UEcross)]));
    axis([-M M -ymax ymax]);
    grid
  end
  subplot(111)
  drawnow
end

% ---------- Meng-ekstrasi model linier dari jaringan ------
----
dy2dx=zeros(outputs*(inputs+1),N);

% Matrik dengan turunan parsial setiap output terhadap
% setiap output dari syaraf lap.tersembunyi
for t=1:N,
  dy2dy1 = W2(:,1:hidden);
  for j = H_output',
    dy2dy1(j,:) = W2(j,1:hidden)*(1-Yhat(j,t).*Yhat(j,t));
  end

  % Matrik dengan turunan parsial output dari setiap syaraf
  % lap.tersembunyi
  % terhadap setiap input:
  dy1dx = W1;
  for j = H_hidden',
    dy1dx(j,:) = W1(j,:)*(1-y1(j,t).*y1(j,t));
  end

  % Matrik dengan turunan parsial setiap output terhadap
  % setiap input
  dl       = (dy2dy1 * dy1dx)';
  dl(inputs+1,:)=dl(inputs+1,:)+W2(:,hidden+1)';
  dy2dx(:,t) = dl(:);
end

figure(si+2*outputs+1)
subplot(212)
plot(dy2dx(1:outputs*inputs,:)')
title('Linearized network parameters')
xlabel('time (samples)')
grid
for ii=1:outputs,
```

```
 subplot(2,outputs,ii);
 hist(E(ii,:),20)
end
subplot(2,outputs,1);
title('Histogram of prediction errors')
subplot(111)
figure(si+1)
```

//LISTING PROGRAM INISIALISASI OPTIMAL TRAINING
//MATLAB 7.0.1

```
% ---------------> OPTRINIT.M <-----------------
% File untuk inisialisasi parameter yang digunakan pada
% "opttrain"

% ----------      Pemilihan       -----------
simul = 'nnet'; % Spesifikasi dari sistem yang digunakan
                % adalah JST
method = 'ff';  % Algoritma pelatihan dengan metode
                % 'forgetting factor'
refty = 'myref';% Sinyal referensi

% ------    Inisialisasi Umum  -------
Ts = 0.2;           % Waktu cuplik (dalam detik)
samples = 1400 ;    % Banyaknya sample pada setiap iterasi
u_0 = 0;            % Input kontrol awal
y_0 = 0;            % Output awal
ulim_min =-inf;     % Input kontrol minimum
ulim_max =inf;      % Input kontrol maksimum


% ----- Spesifikasi JST ------
% Definisi model-NNARX:
% NN, NetDeff, W1f, W2f
% Dan "controller network file" didalamnya harus ada:
% NetDefc, W1c, W2c
nnforw = 'van_vusse_mod'; % Nama file yang didalamnya
                          % terdapat parameter JST model
nnctrl = 'control_rand';  % Nama file yang didalamnya
              % terdapat parameter awal kontroller JST

% ------------ Filter untuk referensi ---------------
Am=1;Bm=1;

% ------------ Parameter-parameter untuk pelatihan ---------
maxiter =5;         % Jumlah itersi maksimum
rho     =1e-6;      % Faktor penalty pada 'squared controls'

% --- Algoritma 'Forgetting factor (ff)' ---
% trparms = [lambda p0]
%    lambda = forgetting factor (nilai sugesti: 0.999)
%    p0     = Diagonal matrik 'covariance' (1-10)
trparms = [0.999 10];

% ------------ Sinyal Referensi ------------
myref =[0.4*ones(100,1);0.4*ones(100,1)];
myref =[myref;0.3*ones(100,1);0.3*ones(100,1)];
```

```
myref =[myref;0.5*ones(100,1);0.5*ones(100,1)];
myref =[myref;0.2*ones(100,1);0.2*ones(100,1)];
myref =[myref;0.6*ones(100,1);0.6*ones(100,1)];
myref =[myref;0.1*ones(100,1);0.1*ones(100,1)];
myref =[myref;0.8*ones(100,1);0.8*ones(100,1)];

% ---- Menspesifikkan vektor-vektor data untuk di-plot -----
% plot_a and plot_b harus dalam bentuk 'cell structures '
% yang didalamnya  terdapat nama vektor tersebut dalam
% 'strings'
plot_a = {'ref_data','y_data'};
plot_b = {'u_data'};
```

//LISTING PROGRAM INISIALISASI PARAMETER
//KONTROLLER OPTIMAL
//MATLAB 7.0.1

```
% ---------------> OPTINIT.M <---------------------
% File untuk inisialisasi parameter yang digunakan pada
% "optcon"

% ----------      Pemilihan       -----------
regty      ='opt';      % Tipe kontroller optimal
refty      ='myref';    % Sinyal referensi
simul      ='nnet';     % Spesifikasi objek kontrol JST


% ------     Inisialisasi Umum  -------
Ts = 0.2;               % Waktu cuplik (dalam detik)
samples  = 1400;        % Jumlah sampel yang akan disimulasi
u_0      = 0;           % Input kontrol awal
y_0      = 0;           % Output awal
ulim_min = 4;           % Input kontrol minimum
ulim_max = 20;          % Input kontrol maksimum


% -- Sistem yang akan dikontrol (JST) --
% Didalam file harus terdapat : NN, NetDeff, W1f, W2f
nnforw = 'van_vusse_mod';          % Nama file

% ----- Spesifikasi kontroller jaringan ------
% Didalam file harus ada variabel-variabel:
% NN, NetDefc, W1c, W2c
nnctrl= 'control_rand2';           % Nama file


% ------------ Filter referensi ---------------
Am = [1];           % Denominator filter
```

```
Bm = [1];           % Numerator filter  (dimulai dalam z^0)


% ------------ Sinyal referensi ------------
myref =[0.4*ones(100,1);0.4*ones(100,1)];
myref =[myref;0.3*ones(100,1);0.3*ones(100,1)];
myref =[myref;0.5*ones(100,1);0.5*ones(100,1)];
myref =[myref;0.35*ones(100,1);0.35*ones(100,1)];
myref =[myref;0.6*ones(100,1);0.6*ones(100,1)];
myref =[myref;0.45*ones(100,1);0.45*ones(100,1)];
myref =[myref;0.8*ones(100,1);0.8*ones(100,1)];

% --- Menspesifikkan vektor-vektor data untuk di-plot ----
% terdapat nama vektor tersebut dalam 'strings'
plot_a = {'ref_data','y_data'};
plot_b = {'u_data'};
```

//LISTING PROGRAM UNTUK SIMULASI KONTROL OPTIMAL
//MATLAB 7.0.1

```
% ---------------------    OPTCON     ---------------------
%  Program unutk mensimulasi optimal kontrol untuk proses
%  yang nonlinear.
%
%  Semua parameter-parametr harus didefinisakan pada file
%  'optinit.m'
%
%  Diadopsi dari Magnus Norgaard IAU, dengan perubahan yang
%  diperlukan.
%-----------------------------------------------------------

clear all
close all
clc
%-----------------------------------------------------------
%----------------  >>>  INISIALISASI  <<<     --------------
%-----------------------------------------------------------
%>>>>>      MEMBACA VARIABEL-VARIABEL DARI FILE       <<<<<<
clear plot_a plot_b
global ugl
optinit
eval(['load ' nnctrl]);        % Membaca jaringan kontroller

% >>>>>>   MENDAPATKAN STRUKTUR REGRESSOR   <<<<<<<<<
na = NN(1);      % # y lampau yang digunakan dalam TDL
nb = NN(2);      % # u lampau yang digunakan dalam TDL
nk = NN(3);      % Waktu tunda dalam sistem
```

```
nab       = na+sum(nb);  % Jumlah dari sinyal input setiap
                         % jaringan
outputs   = 1;           % # output
inputs    = nab-1;       % # input
phi = zeros(inputs,1);   % Inisialisasi vektor 'regressor'

% >>   MENDAPATKAN STRUKTUR DARI 'FORWARD MODEL'   <<<
if strcmp(simul,'nnet'),
  eval(['load ' nnforw]);  % Membaca model jaringan
                           % 'forward' dari sistem
  hiddenf   = length(NetDeff(1,:)); % Jumlah dari syaraf
                                    % tersembunyi
  L_hiddenf = find(NetDeff(1,:)=='L')'; % Letak dari syaraf
                                        % linier tersembunyi
  H_hiddenf = find(NetDeff(1,:)=='H')'; % Letak dari syaraf
                          % 'tangent hiperbolic' tersembunyi
  L_outputf = find(NetDeff(2,:)=='L')'; % Letak dari syaraf
                                        % linier output
  H_outputf = find(NetDeff(2,:)=='H')'; % Letak dari syaraf
                          % 'tangent hiperbolic' output
  y1f       = [zeros(hiddenf,1);1];     % Lapis output
                                        % tersembunyi
end


% > MENDAPATKAN STRUKTUR DARI JARINGAN KONTROLLER<
hiddenc   = length(NetDefc(1,:));% Jumlah syaraf tersembunyi
L_hiddenc = find(NetDefc(1,:)=='L')';% Letak syaraf linier
                                     % tersembunyi
H_hiddenc = find(NetDefc(1,:)=='H')';% Letak dari syaraf
                          % 'tangent hiperbolic' tersembunyi
L_outputc = find(NetDefc(2,:)=='L')';% Letak dari syaraf
                                     % linier output
H_outputc = find(NetDefc(2,:)=='H')';% Letak dari syaraf
                          % 'tangent hiperbolic' output
y1c       = [zeros(hiddenc,1);1]; % Lapis output tersembunyi

%>>>>>>>>>>     INISIALISASI VARIABEL-VARIABEL    <<<<<<<<<
% Mendapatkan panjang dari polinomial-polinomial
nam = length(Am);
nbm = length(Bm);

% Inisialisasi dari sinyal-sinyal
maxlength = 10;  % Panjang maksimum (bisa diubah jika perlu)

ref_old = repmat(y_0,maxlength,1);% Untuk sistem dengan orde
                                  % tinggi
y_old     = repmat(y_0,maxlength,1);
u_old     = repmat(u_0,maxlength,1);
```

```
% Macam-macam inisialisasi
t    = -Ts;
u    = u_0;
y    = y_0;
yhat = y_0;

%>>    MENGHITUNG SINYAL REFERENSI DAN MEM-FILTERNYA
  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
ref=filter(Bm,Am,ref);

% Inisialisasi vektor-vektor data
ref_data    = ref(1:samples);
u_data      = zeros(samples,1);
y_data      = zeros(samples,1);
t_data      = zeros(samples,1);
fighandle=progress;

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

%>>>>>   MEMBACA OUTPUT DARI SISTEM FISIK     <<<<<<<
    phi = [y_old(1:na);u_old(1:nb);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);

%>>>>>>>      MENDAPATKAN SINYAL KONTROL       <<<<<<<<<
  e = ref(i) - y;

  % Kontrol menggunakan jaringan
    phi = [ref(i+1);y;y_old(1:na-1);u_old(1:nb-1);1];
    h1  = W1c*phi;
    y1c(H_hiddenc) = pmntanh(h1(H_hiddenc));
    y1c(L_hiddenc) = h1(L_hiddenc);
    h2 = W2c*y1c;
    u(H_outputc)  = pmntanh(h2(H_outputc));
    u(L_outputc)  = h2(L_outputc);
```

```
 % Memastikan input kontrol masih dalam batasan
  if u>ulim_max,
     u=ulim_max;
  elseif u<ulim_min
     u=ulim_min;
  end

 %>>>>       MENYALIN DATA KE DALAM VEKTOR DATA       <<<<
  u_data(i)        = u;
  y_data(i)        = y;
  t_data(i)        = t;

%>>>>>>>>>          UPDATE WAKTU          <<<<<<<<<<<<<<
  y_old     = shift(y_old,y);
  u_old     = shift(u_old,u);
  ref_old   = shift(ref_old,ref(i));

%>>   CETAK %-UMUR DARI SIMULASI YANG DISELESAIKAN   <<<
  progress(fighandle,floor(100*i/samples));
end
%-------------------------------------------------------------
%---------     >>>   AKHIR DARI LOOP UTAMA   <<<       -------
%-------------------------------------------------------------

%>>>>>>           MENGGAMBAR PLOT          <<<<<<<<<<<<
figure(gcf);clf
set(gcf,'DefaultTextInterpreter','none');

% Plot A
  if(exist('plot_a')==1),
   a_plots=length(plot_a);     % Banyaknya plot dalam plot A
   plmat = zeros(samples,a_plots);       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]);          % Menentukan sumbu-x
    title('Optimal control with a neural network');
    grid on
   legend(plot_a{:});
  end

 % Plot B
  if(exist('plot_b')==1),
   b_plots=length(plot_b);      % Banyaknya plot dalam plot B
   plmat = zeros(samples,b_plots);          for nn =
1:b_plots,
     plmat(:,nn) = eval(plot_b{nn});
```

```
    end
    subplot(2,1,2);
    plot([0:samples-1],plmat,'r');          % Plot plmat
    title('Control Signal')
    xlabel('Samples');
    set(gca,'Xlim',[0 samples-1]);       % Menentukan sumbu-x
    grid on
    legend(plot_b{:});
   end
set(gcf,'DefaultTextInterpreter','tex');
subplot(111)
```