IMPLEMENTASI MODEL ARTIFICIAL NEURAL NETWORK (ANN) *SYSTEM IDENTIFICATION* DAN INVERSE DENGAN ALGORITMA BACKPROPAGATION PADA SISTEM KENDALI BERBASIS INVERSE (*DIRECT INVERSE CONTROL*) MENGGUNAKAN MATLAB

TUGAS - 03

SISTEM BERBASIS PENGETAHUAN

DOSEN PEMBIMBING

Prof. Dr.Eng. Drs. Benyamin Kusumoputro, M.Eng.

Disusun Oleh :

Raihan Nagib (2006574654)

FAKULTAS TEKNIK

PROGRAM STUDI TEKNIK ELEKTRO

DEPOK

2023

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa karena atas segala rahmat dan hidayah-Nya, penulis berhasil menyusun laporan tugas kedua pada mata kuliah Sistem Berbasis Pengetahuan yang berjudul "IMPLEMENTASI MODEL ARTIFICIAL NEURAL NETWORK (ANN) *SYSTEM IDENTIFICATION* DAN INVERSE DENGAN ALGORITMA BACKPROPAGATION PADA SISTEM KENDALI BERBASIS INVERSE (*DIRECT INVERSE CONTROL*) MENGGUNAKAN MATLAB".

Penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada Prof. Dr.Eng. Drs. Benyamin Kusumoputro, M.Eng. selaku dosen pengampu mata kuliah Sistem Berbasis Pengetahuan yang telah memberikan pengajaran, bimbingan, dan saran kepada mahasiswa sehingga penulis berhasil menyelesaikan tugas pertama pada mata kuliah Sistem Berbasis Pengetahuan.

Walaupun penulis sudah membuat algoritma dan menyusun laporan dengan maksimal, penulis menyadari bahwa hasil simulasi dan laporan yang telah dibuat masih banyak kekurangan sehingga penulis sangat mengharapkan adanya kritik dan saran yang dapat membangun dari kepada Prof. Dr.Eng. Drs. Benyamin Kusumoputro, M.Eng. selaku dosen pengampu mata kuliah Sistem Berbasis Pengetahuan supaya laporan ini dapat menjadi lebih baik lagi. Penulis juga berharap laporan ini dapat memberikan manfaat bagi pembaca dan pengembangan dunia pendidikan pada masa kini.

ABSTRAK

Back-propagation Neural Network (BPNN) merupakan suatu algoritma *neural network* untuk melakukan *tuning* pada bobot neural network berdasarkan *error rate* yang telah didapatkan dari *epoch* sebelumnya. Dengan melakukan proses *training* pada bobot neural network, *error rate* dapat berkurang sehingga model yang dibuat akan memiliki performa yang lebih baik. Pada laporan penelitian ini, penulis akan memanfaatkan algoritma BPNN membangun suatu model ANN *system identification* dan inverse. Model ANN *system identification* dan inverse ini akan digunakan untuk membangun sistem kendali berbasis inverse (*direct inverse control*). Dengan menggunakan sistem kendali berbasis inverse, diharapkan akan diperoleh suatu pengendali yang memiliki performa bagus sehingga dapat diterapkan sebagai pengendali pada berbagai macam sistem.

Kata kunci: *Artificial Neural Network, backpropagation, kendali, inverse*

DAFTAR ISI

BAB 1
PENDAHULUAN

2.1 Latar Belakang

Artificial Neural Network (ANN) atau dalam bahasa Indonesia disebut dengan jaringan saraf tiruan merupakan suatu konsep pengolahan informasi dengan menggunakan model komputasi yang terinspirasi dari struktur dan fungsi jaringan saraf biologis manusia. ANN merupakan bagian dari pembelajaran mesin dan merupakan inti dari algoritma *deep learning*.

Nama dan struktur dari ANN terinsiprasi dari otak manusia. ANN akan bekerja dengan cara meniru proses neuron pada otak manusia ketika mengirimkan sinyal antara satu dengan yang lainnya. Model ANN terdiri dari jaringan neuron buatan (disebut juga dengan unit pemrosesan) yang saling terhubung satu sama lain untuk membentuk jaringan saraf buatan. Neuron buatan dalam model ANN ini akan bekerja sama untuk memproses suatu informasi masukan, mempelajari pola-pola yang ada dalam data, dan melakukan tugas-tugas tertentu seperti klasifikasi, *pattern recognition*, hingga kendali.

Pada laporan ini, akan dibahas mengenai implementasi model ANN dengan metode pembelajaran *backpropagation* untuk suatu sistem kendali. Jadi, model ANN yang dibangun akan digunakan sebagai pengendali dari *plant* yang ada. Pada prosesnya, akan digunakan dua buah model ANN yang dihubungkan secara seri. Model ANN yang digunakan yaitu model ANN IDN (*system identification*) yang berguna untuk memprediksi output dari suatu *plant* berdasarkan *input* yang ada, serta model ANN inverse yang berguna untuk memprediksi masukan sebenarnya dari suatu *plant* ketika diketahui nilai *output* tertentu.

Model ANN IDN dan inverse yang dihubungkan secara seri ini dinamakan dengan model ANN *direct inverse control* (DIC) atau dalam bahasa Indonesia disebut dengan model ANN berbasis inverse. Sistem kendali berbasis inverse digunakan untuk menghasilkan sistem kendali dengan performa yang bagus, lebih adaptif, serta dapat menghasilkan keluaran yang akurat. Dengan penerapan sistem kendali berbasis inverse ini diharapkan akan dapat dihasilkan suatu sistem kendali berbasis ANN yang berkualitas dan dapat diterapkan pada sistem di dunia nyata, seperti *drone*, *pressure process rig*, dan lain sebagainya.

Pada laporan ini akan dijelaskan bagaimana proses yang perlu dilakukan untuk memodelkan sistem kendali berbasis inverse untuk diimplementasikan sebagai pengendali dari sistem kontrol. Bilangan random pada rentang tertentu, sinyal step, serta sinyal sinusoidal akan dimanfaatkan sebagai input utama dari sistem kendali berbasis inverse ini. Nilai referensi dan estimasi keluaran dari model ANN pada nantinya akan dikomparasi dengan menggunakan formula *mean squared error* untuk mengukur seberapa baik performa dari sistem kendali berbasis inverse dalam penerapannya sebagai pengendali suatu sistem.

2.2 Rumusan Masalah

- Bagaimana proses yang perlu dilakukan untuk mendapatkan model ANN berbasis inverse yang memiliki performa bagus sebagai pengendali pada suatu sistem kendali?
- Bagaimana pengaruh jenis nilai masukan terhadap performa dari sistem kendali berbasis inverse menggunakan model ANN sebagai pengendali pada suatu sisem kendali?
- Bagaimana algoritma proses dari sistem kendali berbasis inverse dengan memanfaatkan model ANN yang dibangun menggunakan metode pembelajaran *backpropagation*?

- Bagaimana pengaruh dari parameter-parameter model ANN, seperti laju pembelajaran, laju pembelajaran momentum, hingga jumlah *hidden neuron* terhadap performa model ANN sebagai pengendali dari suatu sistem kendali?
- Bagaimana performa dari sistem kendali berbasis inverse, yang dibangun dengan menggunakan model ANN IDN dan inverse, dalam penerapannya sebagai pengendali?

2.3 Tujuan Penelitian

1. Mengembangkan model ANN berbasis inverse yang memiliki performa bagus sebagai pengendali pada suatu sistem kendali melalui proses *training*, validasi, dan *testing* menggunakan algoritma *backpropagation*.
2. Mengetahui pengaruh dari jenis nilai masukan terhadap performa dari sistem kendali berbasis inverse menggunakan model ANN.
3. Mengetahui algoritma proses yang diperlukan untuk memodelkan sistem kendali berbasis inverse dengan memanfaatkan model ANN yang dibangun menggunakan metode pembelajaran *backpropagation*.
4. Menganalisis pengaruh dari parameter-parameter model ANN, seperti laju pembelajara, laju pembelajaran momentum, hingga jumlah *hidden neuron* terhadap performa model ANN sebagai pengendali suatu sistem kendali.
5. Mengetahui performa dari sistem kendali berbasis inverse, yang dibangun dengan menggunakan model ANN IDN dan inverse, dalam penerapannya sebagai pengendali.

2.4 Manfaat Penelitian

Penulisan laporan tentang IMPLEMENTASI MODEL ARTIFICIAL NEURAL NETWORK (ANN) *SYSTEM IDENTIFICATION* DAN INVERSE DENGAN ALGORITMA BACKPROPAGATION PADA SISTEM KENDALI BERBASIS INVERSE (*DIRECT INVERSE CONTROL*) MENGGUNAKAN MATLAB" dilakukan untuk memahami bagaimana tingkat keefektifan penggunaan sistem kendali berbasis inverse yang dibangun dengan memanfaatkan model ANN. Diharapkan laporan ini dapat memberikan manfaat bagi penulis, maupun masyarakat seluas-luasnya mengenai performa dari model ANN sebagai pengendali di suatu sistem kendali.

2.5 Metode Penelitian

Pada penelitian ini, akan dilakukan simulasi penerapan sistem kendali berbasis inverse sebagai pengendali di suatu sistem kendali dengan membuat simulasi program MATLAB pada proses generasi data *plant*, training, validasi, hingga *testing*. Berikut adalah rincian tahapan yang dilakukan selama proses pelatihan model ANN *system identification* dan inverse.

- Membuat suatu *dataset* dengan menggunakan data masukan berupa bilangan acak dari -1 hingga 1, sinyal step, ataupun sinyal sinusoidal. Dataset yang diperoleh tersebut akan digunakan untuk melatih ataupun menguji model ANN *system identification*, inverse, ataupun sistem kendali berbasis inverse.
- Menentukan data *feature*, sebagai input model ANN, dan *target* sebagai data sebenarnya (keluaran dari *plant*) dari *dataset* yang ada. *Feature* dan *target* ini akan digunakan pada proses *training*, validasi, dan *testing* dari model ANN.

- Melakukan proses *feedforward* pada setiap pasangan data pelatihan menggunakan bobot awal acak yang diperoleh dengan menggunakan metode inisialisasi Nguyen-Widrow.
- Melakukan proses *backpropagation* dengan menghitung gradien *descent* dari *loss function*/fungsi kesalahan terhadap nilai bobot dalam jaringan. Pada proses ini, nilai bobot dari model ANN akan dikoreksi selama proses *training* berlangsung.
- Menghitung nilai total error kuadratik pada keluaran data pelatihan (*train*) terhadap data keluaran dari *plant* di setiap epoch.
- Melakukan proses validasi di setiap 50 epoch dengan melakukan proses *feedforward* pada setiap pasangan data validasi dengan menggunakan bobot terbaru.
- Mengambil nilai bobot terbaik dari model berdasarkan proses *training* dan validasi untuk menghindari *overfitting* dengan mengamati nilai total error kuadratik yang diperoleh melalui proses *training* dan validasi. Jika nilai total error kuadratik yang diperoleh melalui proses validasi meningkat pada epoch tertentu, maka bobot pada epoch tersebut akan digunakan pada proses *testing*.
- Melakukan proses *feedforward* pada setiap pasangan data pengujain dengan menggunakan bobot terbaik yang sudah dikoreksi.
- Menghitung nilai total error kuadratik pada keluaran data pengujian terhadap data keluaran dari *plant* (data sebenarnya).

Pada proses pelatihan dan pengujian sistem kendali berbasis inverse, secara umum prosesnya sama. Perbedaannya yaitu, pada proses ini, digunakan bobot awal yang diperoleh dari proses pelatihan model ANN IDN dan inverse. Adapun pada proses ini, proses *training* (*feedforward* dan *backpropagation*) dan *testing* (*feedforward*) akan dilakukan dua kali karena digunakan dua buah model ANN yang berbeda (model ANN IDN dan inverse). Selain itu, keluaran dari model ANN IDN dan inverse juga akan diumpanbalikkan sehingga hasil estimasi model ANN tersebut akan digunakan sebagai masukan pada iterasi selanjutnya. Untuk penjelasan lebih lengkapnya, akan dijelaskan pada BAB III.

2.6 Sistematika Penulisan

- Bab I merupakan bagian pendahuluan yang meliputi latar belakang, rumusam masalah, tujuan penelitian, manfaat penelitian, metodologi, dan sistematika penulisan.
- Bab I merupakan bagian dasar teori dari ANN, sistem kendali, dan model ANN untuk sistem kendali.
- Bab III merupakan bagian deskripsi *plant* dan dataset, program ANN pada sistem kendali berbasis inverse, pengujian model dan pembahasan, perbandingan nilai referensi dengan nilai keluaran sistem kendali berbasis inverse, pengujian parameter-parameter model terhadap total error kuadratik sebagai indikator performa dari sistem kendali berbasis inverse.
- Bab IV merupakan bagian kesimpulan akhir dari keseluruhan percobaan yang telah dilakukan.

BAB II
DASAR TEORI

2.1 Artificial Neural Network (ANN)

Artificial neural network (ANN) atau disebut juga dengan istilah jaringan saraf tiruan merupakan bentuk pemodelan yang memiliki cara kerja yang mirip dengan otak manusia. ANN akan mempelajari sebuah data untuk memberikan nilai-nilai yang sesuai sehingga melalui data masukan yang diberikan, akan dihasilkan data keluaran yang sesuai dengan data sebenarnya. Dalam hal ini, data keluaran yang dihasilkan dapat berupa klasifikasi ataupun estimasi.

Artificial Neural Network (ANN) terdiri dari lapisan input, satu atau lebih lapisan *hidden*, dan lapisan output. Setiap *node* atau neuron buatan pada model ANN akan terhubung satu sama lain dan memiliki bobot. Bobot inilah yang akan menentukan seberapa baik performa dari model ANN yang telah dibangun.

Model ANN akan mengandalkan data pelatihan untuk mempelajari dan meningkatkan akurasinya seiring dengan berjalannya waktu dengan menggunakan metode seperti *backpropagation*. Selama pelatihan, bobot-bobot dari model akan diatur sedemikian rupa sehingga dapat dihasilkan suatu model ANN dengan performa yang bagus. Model ANN dengan performa yang bagus akan menjadi sebuah alat yang ampuh dalam ilmu komputer dan kecerdasan buatan yang memungkinkan digunakan untuk melakukan klasifikasi, mengelompokkan data, ataupun memprediksi suatu nilai tertentu berdasarkan data masukan yang tidak pernah dilihat sebelumnya.

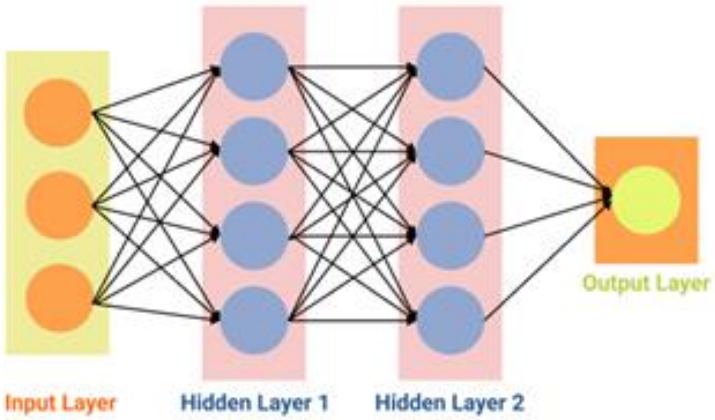
Suatu model ANN akan memiliki tiga buah komponen utama, yakni neuron, koneksi, dan struktur jaringan. Neuron merupakan unit pemrosesan pada model ANN. Neuron akan memiliki beberapa input, mengalirkan sinyal melalui fungsi matematika tertentu, dan menghasilka output. Input dari neuron merupakan perkalian antara nilai bobot dengan sinyal keluaran dari neuron sebelumnya. Adapun output dari neuron akan berupa nilai aktivasi dari akumulasi semua masukan yang ada. Output dari neuron ini akan diteruskan ke neuron berikutnya hingga sampai ke neuron pada lapisan paling akhir.

Neuron pada model ANN akan saling terhubung melalui koneksi yang memiliki bobot. Bobot ini merupakan nilai parameter yang harus disesuaikan selama proses pelatihan jaringan. Bobot yang optimal akan memberikan model ANN yang memiliki performa yang tinggi. Terakhir adalah struktur jaringan dari model ANN yang memiliki berbagai jenis, termasuk jaringan *feedforward*, rekursif, dan berbagai arsitektur lainnya. Struktur jaringan dari ANN akan mempengaruhi performa model ANN untuk menangani tugas-tugas tertentu.

2.2 Lapisan pada Model Artificial Neural Network (ANN)

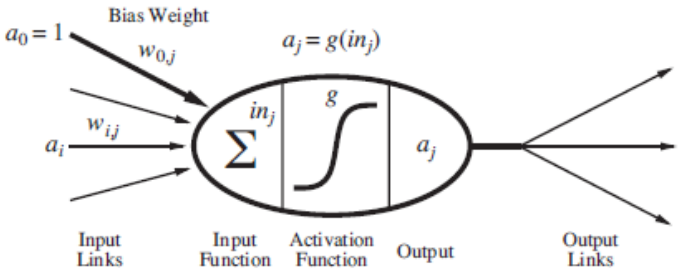
Di dalam tubuh manusia, neuron terbagi menjadi 3 bagian, yaitu *dendrites*, *cell body*, dan *axon*. *Dendrites* diartikan sebagai sinyal masukan dan dipengaruhi oleh *weight* (bobot). *Cell body* diartikan sebagai tempat untuk proses komputasi sinyal. *Axon* diartikan sebagai bagian yang membawa sinyal output ke neuron lainnya. Dengan menganalogikan seperti sistem saraf biologis, ANN dapat direpresentasikan menjadi 3 bagian, yaitu *input layer*, *hidden layer*, dan *output layer*.

- *Input layer* terdiri dari beberapa neuron yang berfungsi untuk menerima data masukan. Setiap neuron yang ada pada *input layer* akan mewakili satu *feature* atau variabel pada data yang dimasukkan ke dalam jaringan. Setiap *neuron* pada *input layer* juga akan terhubung dengan setiap neuron yang ada di dalam *hidden layer*.
- *Hidden layer* adalah lapisan yang berada di antara *input layer* dan *output layer* yang berfungsi untuk meproses dan mempelajari data masukan sehingga dihasilkan nilai bobot yang optimal. Setiap neuron pada *hidden layer* akan mengambil masukan dari laposan sebelumnya, melakukan komputasi dengan bobot dan fungsi aktivasi, dan mengirimkan sinyal keluarannya ke lapisan berikutnya.
- *Output layer* merupakan lapisan terakhir dalam jaringan yang menerima masukan dari lapisan sebelumnya dan berfungsi untuk memberikan informasi mengenai *output* yang sesuai dari sistem berdasarkan data masukan (dan bobot) yang ada. *Output layer* mewakili hasil akhir dari pemrosesan jaringan, seperti klasifikasi, regresi, *pattern recognition*, dan lain sebagainya.



Gambar 2.1 Arsitektur Artificial Neural Network (ANN)

2.3 Prinsip Kerja Artificial Neural Network (ANN)



Gambar 2.2 Model Matematis Neuron

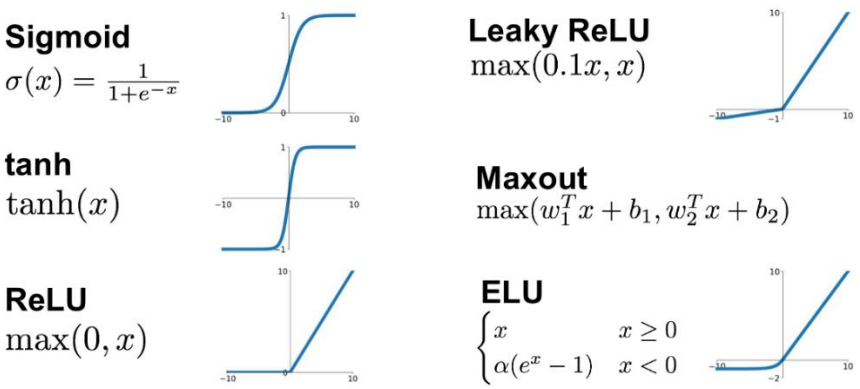
Artificial Neural Network (ANN) berfungsi untuk menyelesaikan suatu masalah dengan cara mempelajari data *testing* dan *training* yang telah diberikan dengan sendirinya dengan menggunakan satu perangkat yang sudah ada pola *training*-nya. Dengan mempelajari pola dari data *testing* dan *training* yang telah diberikan, ANN akan memproses nilai bobotnya hingga output yang dihasilkan sesuai dengan target yang telah ditentukan.

2.3.1 Faktor Bobot

Bobot adalah parameter penting dalam jaringan saraf yang mewakili kekuatan koneksi antara node yang satu dengan node yang lain. Semakin besar nilai bobot maka semakin kuat koneksi antar node. Bobot dapat disesuaikan dan perlu diatur supaya jaringan saraf berfungsi dengan baik. Jaringan saraf akan akan terus bertambah bobotnya karena jaringan saraf akan terus meningkatkan kemampuan belajarnya. ANN dapat belajar dari masalah baru ketika ada masalah baru muncul yang mana ANN akan langsung menyusun kembali nilai bobot supaya sesuai dengan nilai karakter yang ada.

2.3.2 Faktor Aktivasi

Tingkat aktivasi adalah keadaan internal yang dimiliki oleh setiap neuron yang mana neuron akan mengirimkan aktivitasnya sebagai sinyal ke neuron lainnya. Aktivasi mempunyai fungsi penjumlahan input dan bobot yang sampai ke neuron. Hasil penjumlahan tersebut akan diproses di masing-masing neuron oleh fungsi aktivasi yang mana hasilnya akan dibandingkan dengan nilai ambang. Jika hasil yang diperoleh di atas ambang batas maka aktivasi neuron akan dibatalkan. Sebaliknya, jika hasil yang diperoleh di bawah ambang batas maka neuron akan aktif. Setelah neuron diaktifkan, neuron akan mengirimkan nilai *output* ke semua neuron yang ada kaitannya dengan bobot *output* dan *input* selanjutnya.



Gambar 2.3 Jenis-Jenis Fungsi Aktivasi

2.4 Parameter Artificial Neural Network (ANN)

Untuk mendapatkan model yang akurat dengan menggunakan perhitungan algoritma BPNN, dibutuhkan nilai parameter dan *hyperparameter* yang terbaik. Hyperparameter pada machine learning adalah parameter untuk mengendalikan proses pembelajaran.
Berikut adalah *hyperparameter* untuk algoritma BPNN :

1. **Learning Rate (α)**

   Learning rate adalah *hyperparameter* untuk mengatur besarnya perubahan bobot pada setiap epoch. Apabila nilai learning rate semakin besar maka perubahan bobot untuk setiap epoch akan semakin besar. Apabila nilai learning rate makin besar maka proses training akan mencapai fungsi minimum yang lebih cepat tetapi ada kelemahan, yaitu risiko ketidakakuratan pada tahap tersebut akan membesar.

2. **Momentum Coefficient (μ)**

   Momentum adalah suatu metode untuk mempertahankan arah dari *gradient descent*. Arah dari *gradient descent* didapatkan dari hasil kombinasi arah yang

dikomputasi pada iterasi saat ini dengan iterasi sebelumnya. *Momentum coefficient* dapat digunakan untuk mengatur besar pengaruh arah iterasi sebelumnya.

3. **Layer Size (J)**

   Layer size adalah *hyperparameter* untuk menentukan keakuratan dan kualitas model. Apabila lapisan yang tersedia semakin banyak maka model yang terbentuk dari algoritma BPNN akan semakin akurat. Namun, jumlah lapisan tidak hanya berpengaruh pada keakuratan model tetapi juga bergantung pada kualitas model dan kualitas & kuantitas data *training*.

4. **Loss Function**

   Loss function adalah fungsi untuk menghitung perbedaan antara *output* algoritma dan *output* target. Hasil perhitungan *loss function* dapat digunakan untuk meng-*update* bobot.

5. **Epoch Count**

   Epoch count adalah *hyperparameter* untuk menentukan jumlah algoritma yang akan mengolah seluruh dataset *training*. Ketika jumlah epoch semakin banyak maka bobot yang mengalami perubahan pada algoritma akan semakin banyak. Hal ini menyebabkan adanya peningkatan akurasi dari model tersebut.

Selain itu, ada 2 metode untuk inisialisasi awal dari tiap bobot antar neuron :

1. **Random**

   Metode inisialisasi secara *random* adalah metode untuk menentukan bobot awal secara acak yang mana penentuan bobot awal tersebut dapat dilakukan dengan menggunakan *random function* pada MATLAB. Tujuan dari inisialisasi awal secara random adalah untuk menghasilkan *symmetry breaking.* Konsep dari *symmetry breaking* adalah ketika besar bobot pada model sama maka fenomena yang terjadi akan tidak ideal. Dari konsep tersebut, bisa diterapkan untuk menentukan inisialisasi secara random dengan cara menentukan ambang batas atas dan bawah dengan tujuan agar bobot neuron yang dihasilkan tidak terlalu besar untuk proses pembelajaran.

2. **Nguyen-Widrow**

   Metode Nguyen-Widrow adalah suatu algoritma yang dapat memodifikasi bobot dengan tujuan untuk meningkatkan kecepatan proses pembelajaran.
   Berikut adalah cara -cara yang dilakukan dengan metode Nguyen-Widrow

   - Menentukan nilai dari faktor skala yang dilambangkan dengan β
   - Menentukan Inisialisasi bobot secara random pada rentang -0.5 sampai 0.5
   - Menghitung norm dari vektor bobot
   - Melakukan *update* bobot
   - Mengatur *set bias* mengunakan bilangan acak antara β sampai -β

2.5 Algoritma dari Pembelajaran Backpropagation

Backpropagation adalah metode pelatihan yang menggunakan metode ANN untuk menghitung turunan di dalam *deep feedforward neural networks*. *Backpropagation* membentuk bagian penting dari sejumlah algoritma supervised learning untuk training *feedforward neural networks*. Tujuan dari backpropogation adalah untuk meminimalkan error

pada output yang dihasilkan oleh jaringan. Metode backpropagation biasanya menggunakan jaringan multilayer.

Ada 2 tahap perhitungan dari *backpropagation*, yaitu yang pertama melakukan perhitungan maju dan yang kedua menghitung error antara *output* dan target. Perhitungan maju digunakan untuk menghasilkan *output* dengan cara memberikan pola *input* ke dalam *artificial neural network (ANN)*. Perhitungan tersebut dapat memperbaiki bobot di semua neuron karena perhitungan tersebut melakukan *backpropagation* pada error. Pola input yang masuk ke dalam ANN akan diolah oleh setiap neuron yang menghasilkan tingkat aktivasi dengan tujuan untuk menghasilkan *output* ANN yang kemudian membandingkan output ANN tersebut dengan target yang telah ditentukan. Apabila terjadi perbedaan antara *output* ANN dan target maka akan menghasilkan error yang mana error tersebut akan dilakukan *backpropagation* sampai ke lapisan paling depan dengan tujuan untuk melakukan pengubahan bobot pada hubungan neuron.

Berikut adalah tahap dari *training* dan *testing* pada ANN :

1. **Training**

   Tahap *training* pada ANN adalah tahap untuk mendapatkan nilai bobot, bias, alpha dan μ. Berikut adalah tahap-tahap *training* pada ANN :

   a. Inisialisasi data

      Hal ini dilakukan pertama kali dengan menggunakan random value dan nguyen widrow. Pada inisialisasi data ini akan dibuat pembobotan yang dilakukan apakah data yang kita miliki dapat diproses selanjutnya atau tidak. Parameter yang menentukan kedua hal ini adalah α dan μ.

      1. Pembobotan awal (random value)

         Digunakan pembobotan awal yang ada dari rentang -0.5 hingga 0.5. Nilai tersebut secara acak ditempatkan pada pembobotan pada v dan w.

      2. Nguyen Widrow

         Metode ini digunakan untuk penentuan bobot pada *input layer* dan *hidden layer*. Bobot awal yang ditentukan sebelumnya akan mencari faktor skala

         $\beta = 0.7h^{\frac{1}{x}}$

         Selanjutnya, norma dari vektor bobot dicari dengan persamaan:

         $$||v_j|| = \sqrt{\sum_{i=1}^{P} {v_{ij}}^2}$$

   b. Preprocessing

      Preprocessing yang dilakukan adalah rangkaian normalisasi dengan mengubah rentang dari variabel yang digunakan. Hal ini dilakukan agar rentang yang kita miliki sama sehingga pengaruhnya akan sama terhadap data. Preprocessing data yang dilakukan menggunakan metode Z-score.

      (masukin rumus z-score)

   c. Training data

      Pada training data yang dilakukan, feedforward, backpropagation, pembaruan bobot dan bias dilakukan terus menerus setelah adanya error value

yang didapatkan pada hasil data hasil error. Nantinya proses ini akan berakhir apabila nilai dari error yang didapatkan sudah sesuai dengan nilai yang diinginkan.

- *Feedforward*

Pada proses ini, ANN aka mengklasifikasi sampel data ke dalam kelas-kelas yang ditentukan sebelumnya. Feedforward memiliki algoritma sebagai berikut :

1. Komputasi lapisan masukan. Untuk setiap masukan $x_i, i = 1,2,\dots,n$:
   a. Menerima input x_n
   b. Mengirimkan input ke *neuron-neuron* pada *hidden layer*.
2. Komputasi lapisan tersembunyi. Untuk setiap unit tersembunyi $z_j, j = 1,2,\dots,p$:
   a. Menghitung sinyal masukan dari lapisan masukan dengan bobotnya dengan $zin_j = b_j + \sum x_i v_{ij}$.
   b. Menghitung nilai aktivasi setiap *neuron* pada lapisan tersembunyi yang dinyatakan sebagai $z_j = \sigma(zin_j)$.
   c. Mengirimkan nilai aktivasi sebagai masukan pada *neuron-neuron* lapisan tersembunyi berikutnya atau lapisan keluaran.
3. Komputasi lapisan keluaran. Untuk setiap keluaran $y_k, k = 1,2,\dots,m$:
   a. Menghitung sinyal masukan dengan bobotnya dengan$yin_k = b_k + \sum z_j w_{jk}$.
   b. Menghitung nilai aktivasi setiap *neuron* keluaran sebagai keluarannya dengan $y_k = \sigma(yin_k)$.
   c. Kuantisasi nilai keluaran dengan ketentuan :

$$y_k = \begin{cases} 0 & ;0 \le y_k \le 0.3 \\ y_k & ;0.3 < y_k < 0.7 \\ 1 & ;0.7 \le y_k \le 1 \end{cases}$$

Setelah data selesai pada *feedforward*, data akan masuk kedaam backpropagation untuk keperluan optimasi dengan penentuan nilai kesalahan yang ada.

- *Backpropagation*

1. Komputasi error di lapisan keluaran. Untuk setiap keluaran$y_k, k = 1,2,\dots,m$:
   a. Menerima target yang bersesuaian dengan input
   b. Menghitung gradient descend dari fungsi nilai kesalahan dengan persamaan $\delta_k = (t_k - y_k).\sigma'(y_{in_k})$
   c. Menghitung besar koreksi bobot masukan dari *neuron* lapisan keluaran dengan $\Delta w_{jk} = \alpha \delta_k z$, di mana z adalah nilai aktivasi dari *neuron* lapisan tersembunyi sebelumnya

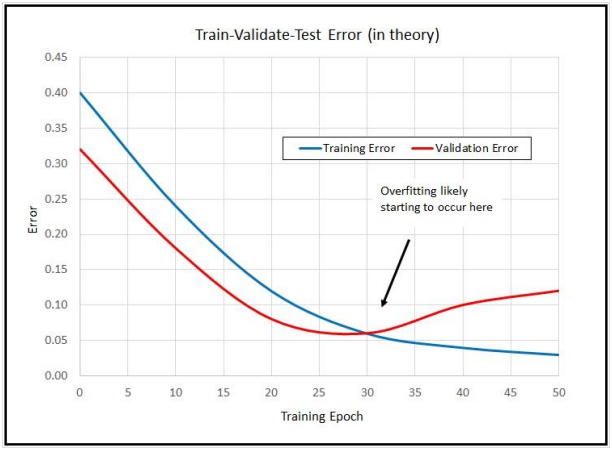
d. Menghitung besar koreksi *bias neuron* keluaran dengan persamaan $\Delta b_k = \alpha \delta_k$
e. Mengirimkan δ_k ke *neuron-neuron* pada lapisan sebelumnya

2. Komputasi kesalahan di lapisan tersembunyi. Untuk setiap unit tersembunyi $z_j, j = 1,2,\dots,p$:
   a. Menghitung semua koreksi error dengan $\delta_{inj} = \sum \delta_k w_{jk}$
   b. Menghitung nilai aktivasi koreksi kesalahan dengan $\delta_j = \delta_{in_j} \sigma'(z_{in_j})$
   c. Menghitung besar koreksi bobot pada lapisan tersembunyi dengan $\Delta v_{ij} = \alpha \delta_j x_i$
   d. Menghitung besar koreksi bias lapisan tersembunyi dengan $\Delta b_j = \alpha \delta_j$
3. Pembaruan bobot dan bias
   Proses ini akan terus dilakukan, apabila data yang sebelumnya sudah ada, data tersebut akan diperbarui dengan fungsi berikut:
   Bobot output layer (w):
   $w_{jk}\,(t+1) = w_{jk}\,(t) + \Delta w_{jk}$
   $w_{0k}\,(t+1) = w_{0k}\,(t) + \Delta w_{0k}$

   Bobot hidden layer (v):
   $v_{jk}\,(t+1) = v_{jk}\,(t) + \Delta v_{jk}$
   $v_{0k}\,(t+1) = v_{0k}\,(t) + \Delta v_{0k}$
4. Menghitung error
   Proses penghitungan error yang dilakukan akan dilakukan terus menerus. Error yang digunakan adalah error kuadratik.
   $error\ (E) = |t_{pk}m - y_{pk}|$

2. Validasi

Tahap validasi merupakan tahap untuk mengukur sejauh mana model ANN yang telah dilatih mampu menggeneralisasi dari data pelatihan ke data yang tidak pernah dilihat sebelumnya. Melalui tahap validasi, dimungkinkan untuk mengevaluasi sejauh mana model ANN mampu beradaptasi terhadap berbagai macam jenis data masukan. Sederhananya, tahapan validasi pada model ANN merupakan proses *testing* terhadap bobot yang sudah dikoreksi yang dilakukan selama proses *training* masih berlangsung. Jadi, proses validasi ini akan dilakukan di tengah-tengah proses *training* sedang berlangsung. Proses validasi ini akan dilakukan setiap proses *training* sudah dilakukan sebanyak 50 epoch. Dengan validasi, akan diamati bagaimana perkembangan nilai bobot dari model untuk menghindari terjadinya peristiwa *overfitting* selama proses *testing*.



Gambar 2.4 Grafik Error Proses *Training* dan Validasi

Dari grafik tersebut dapat diketahui jika nilai error dari validasi akan meningkat di titik tertentu. Adapun nilai error dari proses *training* dalam hal ini tetap menurun nilainya. Peningkatan nilai error validasi di tengah-tengah proses *training* ini mengindikasikan bahwa model kehilangan kemampuan untuk menggeneralisasi dan akan tampil buruk saaat diterapkan pada data yang tidak pernah dilihat sebelumnya (*overfitting*). Oleh karena itu, ketika gejala *overfitting* ini mulai tampak terlihat, proses *training* harus segera dihentikan atau nilai bobot terakhir sebelum nilai error validasi meningkat harus segera dicatat untuk digunakan pada proses *testing*.

3. Testing

Tahap *testing* adalah tahap untuk menguji data hasil *training* dengan data yang lain yang berbeda dengan menggunakan proses *feedforward* tanpa adanya perubahan bobot. *Output* yang dihasilkan dari tahap *testing* akan dibandingkan dengan data target pengujian. Hasil perbandingan tersebut akan didapatkan nilai *recognition rate* (RR) yang merupakan persentase dari keakuratan bobot dan bias terhadap data *input training*.

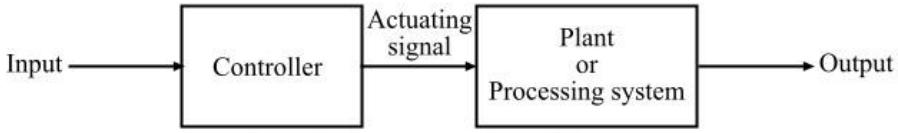
2.6 Sistem Kendali

Sistem kendali merupakan suatu sistem yang dapat digunakan untuk mengatur atau mengendalikan perilaku atau operasi dari suatu sistem ataupun proses. Tujuan utama dari sistem kendali yaitu untuk mencapai hasil yang diinginkan atau menjaga suatu variabel dalam batas yang ditentukan. Sistem kendali digunakan hampir di semua bidang, seperti di antaranya adalah otomotif, indutrsi, elektronika, dan lain sebagainya.

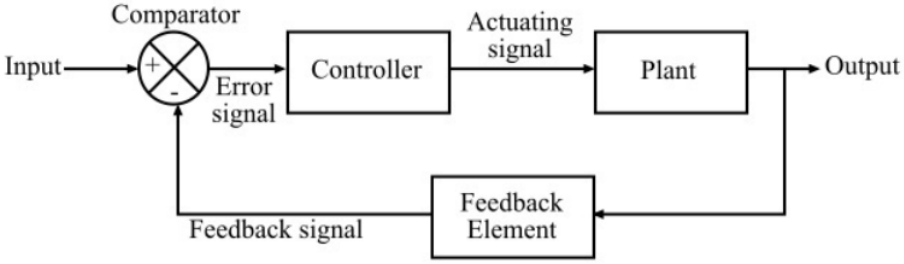
Sistem kendali terdiri dari beberapa komponen utama, di antaranya adalah input, pengendali, *plant*, dan output. Input merupakan masukan dari sistem pengendali. Input dapat meliputi nilai referensi agar nilai output bisa sesuai dengan nilai referensi yang diberikan. Selanjutnya, pengendali merupakan otak dari sistem kendali. Pengendali berguna untuk memproses nilai informasi masukan dengan algoritma tertentu sehingga akan dihasilkan sinyal kendali untuk mengendalikan *plant* agar dihasilkan respons keluaran sesuai dengan apa yang diinginkan. *Plant* merupakan perangkat yang akan dikendalikan oleh sistem kendali. Terakhir, output merupakan respons keluaran dari *plant* ketika diberikan sinyal kendali keluaran dari pengendali.

Pada praktiknya, keluaran dari *plant* akan bernilai konvergern pada keadaan tunak, memiliki nilai *steady state error* yang kecil, dan memiliki karakteristik respons transien yang

bagus jika *plant* dikendalikan dengan pengendali yang bagus. Sistem kendali dibagi menjadi dua jenis utama, yakni sistem kendali terbuka dan sistem kendali tertutup. Sistem kendali terbuka tidak menggunakan umpan balik (*feedback*) dari sistem untuk membuat keputusan. Pengendali akan mengirimkan sinyal kendali tanpa memperhatikan respons aktual dari sistem. Adapun sistem kendali tertutup akan menggunakan umpan balik (*feedback*) untuk memantau kondisi aktual dari sistem sehingga akan diperoleh nilai error. Pada prosesnya, nilai error ini akan diperkecil dengan menggunakan pengendali sehingga respons keluaran dari sistem bisa sesuai dengan *set-point* yang diberikan. Dengan proses inilah, sistem kendali tertutup biasanya akan menghasilkan respons keluara yang lebih baik dan akurat jika dibandingkan dengan sistem kendali terbuka.



Gambar 2.5 Sistem Open Loop



Gambar 2.6 Sistem Closed Loop

2.7 Perbedaan ANN untuk *Pattern Recognition* dengan ANN untuk Sistem Kendali

Dalam prosesnya untuk melakukan percobaan ini, perlu diketahui terlebih dahulu perbedaan penggunaan ANN dalam *pattern recognition* (klasifikasi, klusterisasi, dan lain sebagainya) dengan penggunaan ANN dalam sistem kendali. Pada dasarnya, ukuran sukses pada *pattern recognition* dapat diukur dengan menggunakan *recognition rate*, yakni berapa persen nilai benar/salah dari semua data yang di-*test* dengan menggunakan ANN. Adapun pada ANN dalam sistem kendali, ukuran sukses dapat dilihat dengan menghitung nilai MSE (*mean squared error*), yaitu perbedaan antara hasil keluaran ANN dengan data seharusnya (error kuadratik).

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

dimana :

$\boldsymbol{n}$ = jumlah sampel data

$\boldsymbol{Y_i}$ = nilai aktual

$\boldsymbol{\hat{Y}_i}$ = nilai prediksi

Pada ANN untuk *patterrecognition*, input yang diberikan dapat memiliki nilai dimensi yang sangat besar, hingga mencapai ribuan. Berbeda dengan ANN untuk sistem kendali, dimana dimensi inputnya umumnya tidak terlalu banyak karena hanya berganting pada jumlah input dan jumlah output dari *plant* yang digunakan.

Pada ANN untuk *pattern recognition*, proses normalisasi dilakukan dengan menggunakan *range* angka mulai dari 0 sampai 1. Hal yang berbeda terjadi pada ANN untuk sistem kendali dimana normalisasi harus dilakukan dengan menggunakan range dari -1 sampai dengan 1. Hal ini disebabkan karena daerah kerja dari sistem kendali yang bisa dimulai dari angka negatif.

Pada ANN untuk *pattern recognition*, dimensi data masukan akan mempunyai rentang nilai yang sama antar dimensi inputnya. Sedangkan data masukan pada ANN sistem kendali akan mempunyai rentang nilai yang dapat berbeda antar dimensi inputnya.

BAB III
ALGORITMA

3.1 Deskripsi Sistem

Pada pemodelan ANN untuk sistem kendali, model dari data masukan dan data keluaran dapat dirumuskan dengan menggunakan persamaan sebagai berikut.

$$\boldsymbol{y(k) = f[y(k-1), y(k-2), \dots, y(k-n+1), u(k), u(k-1), \dots, u(k-m+1)]}$$

dimana,

$\boldsymbol{y(k)}$ = keluaran sistem pada waktu sekarang (k)

$\boldsymbol{u(k)}$ = masukan sistem pada waktu sekarang (k)

$\boldsymbol{f(k)}$ = fungsi nonlinear yang tidak diketahui

Nilai n dam m merupakan variabel yang menujukkan orde dari sistem. Kedua variabel ini akan mempengaruhi seberapa jauh data lampau akan digunakan untuk menghitung data keluaran dari sistem kendali ANN.

Sistem yang digunakan pada percobaan ini merupakan sistem orde dua dengan nilai n dan m sama dengan 2. Artinya, *plant* akan memiliki fungsi keluaran yakni sebagai berikut.

$$\boldsymbol{y(k) = f[y(k-1), u(k), u(k-1)]}$$

dimana,

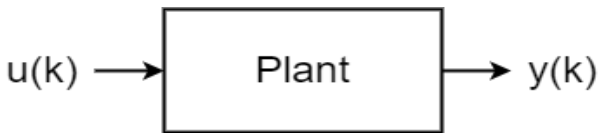
$\boldsymbol{y(k-1)}$ = keluaran sistem satu langkah sebelumnya $(\boldsymbol{k-1})$

$\boldsymbol{u(k)}$ = masukan sistem pada waktu sekarang $(\boldsymbol{k})$

$\boldsymbol{u(k-1)}$ = masukan sistem satu langkah sebelumnya $(\boldsymbol{k-1})$

Berikut ini merupakan persamaan dari *plant* sistem yang akan digunakan pada percobaan.

$$\boldsymbol{y(k) = \frac{1}{1 + y(k-1)^2} + 0.25u(k) - 0.3u(k-1)}$$

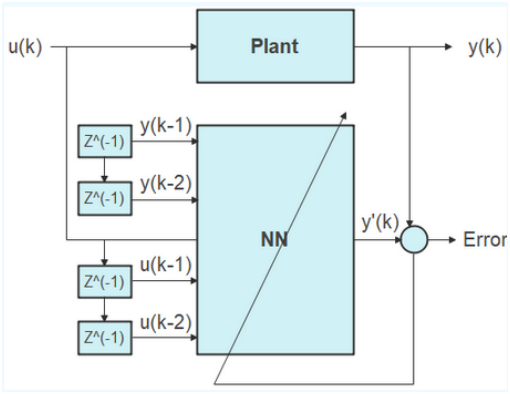


Gambar 3.1 Blok Diagram dari Sistem

Pada percobaan yang dilakukan, nilai input yang dimasukkan ke dalam *plant* ini merupakan data random dengan nilai yang berkisar di antara -1 hingga +1.

3.1.1 Sistem Kendali ANN tanpa Feedback

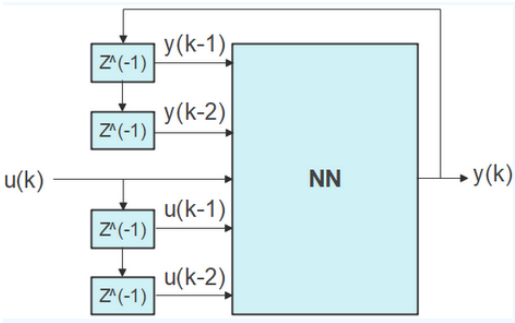
Pada percobaan ini, nilai u(k) dan y(k) yang dikeluarkan oleh *plant* akan diberikan kepada model ANN sebagai input. Adapun nilai u(k) dan y(k) ini akan digunakan untuk men-*training* model ANN dengan menggunakan metode *backpropagation* sehingga akan diperoleh nilai bobot terbaik dari model. Adapun di tengah-tengah proses *training* ini akan dilakukan proses validasi untuk menghindari terjadinya *overfitting*. Model ANN dengan nilai bobot terbaik ini pada nantinya akan digunakan untuk menghasilkan nilai y'(k). Nilai y'(k) akan dibandingkan dengan nilai y(k) keluaran *plant* untuk mengetahui tingkat keakuratan dari model ANN. Adapun cara membandingkan kedua nilai tersebut, yakni dengan menggunakan perhitungan nilai *Mean Squared Error* (MSE).



Gambar 3.2 Sistem Kendali ANN tanpa *Feedback* (tahap 1)

3.1.2 Sistem Kendali ANN dengan Feedback

Pada percobaan ini, nilai input yang digunakan akan sama seperti percobaan sebelumnya. Akan tetapi, untuk input y(k) yang digunakan bukanlah dari *plant* seperti yang dilakukan pada percobaan sebelumnya. Nilai input y(k) pada percobaan ini akan didapatkan melalui keluaran/hasil prediksi dari model ANN. Ini merupakan prinsip sistem kendali loop tertutup. Dengan proses ini, pada nantinya akan dibandingkan hasil prediksi dari model ANN, dengan input *feedback* dari output, dengan keluaran dari *plant* itu sendiri. Semakin mirip nilai keduanya, maka semakin bagus performa dari model ANN yang dibangun. Adapun nilai *Mean Squared Error* (MSE) dalam hal ini akan digunakan sebagai acuan indeks performa model.



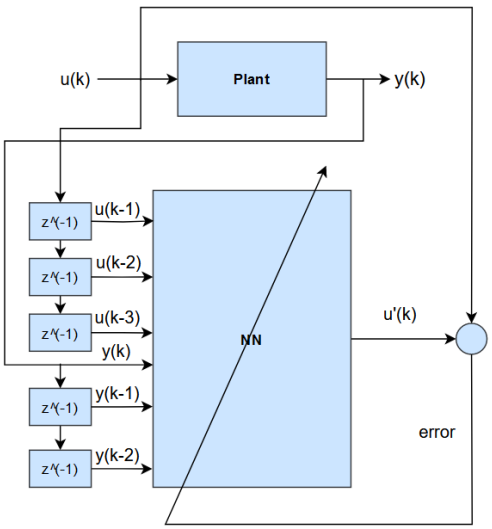
Gambar 3.3 Sistem Kendali ANN dengan *Feedback* (tahap 2)

3.1.3 Sistem Kendali *Direct Inverse Control* (DIC)

Pada model ANN ini, akan digunakan model ANN inverse. Model ANN inverse merupakan pada dasarnya sama seperti model ANN yang disinggung sebelumnya, perbedaannya ada pada input dan outputnya. Model ANN inverse menggunakan input y dan u untuk mengestimasi nilai keluaran berupa u. Tujuan dari model ANN inverse ini yaitu untuk memprediksi nilai input berdasarkan nilai output yang diberikan.

Proses pelatihan model ANN invers melibatkan memberikan pasangan data input-output kepada model sehingga dapat belajar dan menyesuaikan parameter internalnya untuk menghasilkan output yang mendekati input yang diberikan. Ini melibatkan penyesuaian bobot dan bias dalam jaringan saraf agar model dapat memberikan prediksi yang semakin akurat.

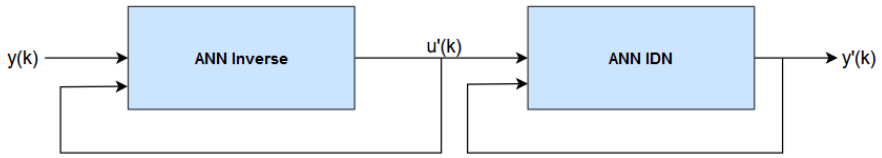
Model ANN inverse banyak diterapkan pada sistem pengendalian proses, pemodelan sistem yang kompleks, ataupun proses pemulihan informasi dari data yang dihasilkan oleh suatu sistem.



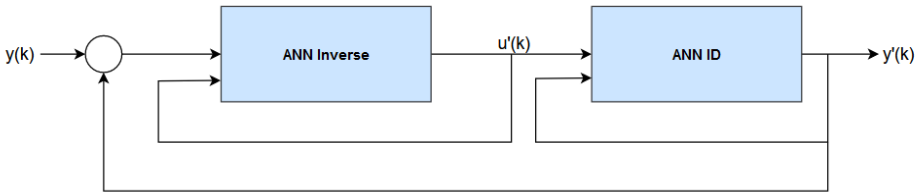
Gambar 3.3 Sistem Kendali ANN Inverse tanpa Feedback

Sistem kendali berbasis inverse (*direct inverse control*) merupakan suatu metode pengendalian keluaran dari suatu sistem dengan memanfaatkan *plant* inverse sebagai pengendalinya. Penambahan pengendali inversi pada DIC ini bertujuan untuk memperbaiki nilai keluaran dari sistem agar bisa sesuai dengan nilai referensinya.

Sama seperti sistem kendali pada umumnya, sistem kendali berbasis inverse juga dibagi menjadi dua jenis, yakni, DIC dengan konfigurasi lup terbuka dan lup tertutup. Pada konfigurasi lup tertutup, estimasi nilai keluaran pada waktu sekarang akan diumpanbalikkan pada waktu berikutnya sebagai input pada model ANN inverse. Nilai umpan balik ini akan digunakan untuk menghitung error antara nilai keluaran dengan nilai referensi. Nilai error tersebut akan digunakan untuk mengoreksi nilai bobot pada model ANN *system identification* ataupun inverse.



Gambar 3.4 Blok Diagram Sistem Kendali Berbasis Inverse dengan Konfigurasi Lup Terbuka



Gambar 3.5 Blok Diagram Sistem Kendali Berbasis Inverse dengan Konfigurasi Lup Tertutup

3.2 Generate Dataset

Generate dataset pada percobaan ini dapat dilakukan dengan memberikan nilai input acak dari rentang -1 sampai dengan 1 pada *plant* yang ada, nilai input tertentu sehingga dihasilkan sinyal berbentuk step, serta input berupa sinyal sinusoidal. Pada nantinya, akan diperoleh dua jenis dataset (berupa *feature* dan *target*), yakni dataset untuk model ANN IDN (*system identification*) dan model ANN inverse. Masing-masing dataset tersebut terdiri dari 50 ribu sampel data (untuk input acak) atau 12 ribu sampel data (untuk input sinyal step dan sinusodidal) yang akan digunakan untuk melakukan proses *training*, validasi, dan *testing* terhadap model ANN.

```
% Contoh kode program untuk input random
%% filename : generateDataset.m

%% PLANT
%                1
% y(k) = ----------------- + (0.25 * u(k)) - (0.3 * u(k-1))
%         (1 + y(k-1)^2))

%% Generate Dataset

% Data Random
y = zeros(50000, 1);
u = (1 + 1) * rand(50000, 1) - 1;

[u_row, u_column] = size(u);
[y_row, y_column] = size(y);

for k =1:y_row
    if k == 1
        y(k) = (1 / (1 + 0)) - (0.25 * u(k)) - (0.3 * 0);
    else
        y(k) = (1 / (1 + y(k-1)^2)) - (0.25 * u(k)) - (0.3 * u(k-1));
    end
end

% INPUT
uk = zeros(u_row, 1);
uk_min1 = zeros(u_row, 1);
uk_min2 = zeros(u_row, 1);
uk_min3 = zeros(u_row, 1);
yk = zeros(y_row, 1);
yk_min1 = zeros(y_row, 1);
yk_min2 = zeros(y_row, 1);


for i = 1:u_row
    uk(i) = u(i);
    yk(i) = y(i);
end

for i = 1:(u_row - 1)
    uk_min1(i + 1) = u(i);
```

```
    yk_min1(i + 1) = y(i);
end

for i = 1:(u_row - 2)
    uk_min2(i + 2) = u(i);
    yk_min2(i + 2) = y(i);
end

for i = 1:(u_row - 3)
    uk_min3(i + 3) = u(i);
end

dataset = cat(2, uk, uk_min1, uk_min2, yk_min1, yk_min2, yk);
dataset_inverse = cat(2, uk_min1, uk_min2, uk_min3, yk, yk_min1, yk_min2, uk);

dataTable = array2table(dataset);
dataTable = renamevars(dataTable, ["dataset1", "dataset2", "dataset3",
"dataset4", "dataset5", "dataset6"], ["u(k)", "u(k-1)", "u(k-2)", "y(k-1)",
"y(k-2)", "y(k)/output"]);


dataTable_inverse = array2table(dataset_inverse);
dataTable_inverse = renamevars(dataTable_inverse, ["dataset_inverse1",
"dataset_inverse2", "dataset_inverse3", "dataset_inverse4",
"dataset_inverse5", "dataset_inverse6", "dataset_inverse7"], ["u(k-1)", "u(k-
2)", "u(k-3)", "y(k)", "y(k-1)", "y(k-2)", "u(k)/output"]);

writetable(dataTable_inverse, 'dataset_inverse.xlsx','Sheet',1);
writetable(dataTable, 'dataset.xlsx','Sheet',1);
```

Berikut adalah cuplikan dataset yang berhasil di-*generate*.

uk	uk1	uk2	yk1	yk2	ykoutput
Number	Number	Number	Number	Number	Number
u(k)	u(k-1)	u(k-2)	y(k-1)	y(k-2)	y(k)/output
-0.8049	0	0	0	0	1.2012
-0.4430	-0.8049	0	1.2012	0	0.7616
0.0938	-0.4430	-0.8049	0.7616	1.2012	0.7424
0.9150	0.0938	-0.4430	0.7424	0.7616	0.3878
0.9298	0.9150	0.0938	0.3878	0.7424	0.3623
-0.6848	0.9298	0.9150	0.3623	0.3878	0.7762
0.9412	-0.6848	0.9298	0.7762	0.3623	0.5942

uk1	uk2	uk3	yk	yk1	yk2	ukoutput
Number	Number	Number	Number	Number	Number	Number
u(k-1)	u(k-2)	u(k-3)	y(k)	y(k-1)	y(k-2)	u(k)/output
0	0	0	1.2012	0	0	-0.8049
-0.8049	0	0	0.7616	1.2012	0	-0.4430
-0.4430	-0.8049	0	0.7424	0.7616	1.2012	0.0938
0.0938	-0.4430	-0.8049	0.3878	0.7424	0.7616	0.9150
0.9150	0.0938	-0.4430	0.3623	0.3878	0.7424	0.9298
0.9298	0.9150	0.0938	0.7762	0.3623	0.3878	-0.6848
-0.6848	0.9298	0.9150	0.5942	0.7762	0.3623	0.9412

Gambar 3.4 Cuplikan Dataset

3.3 Pelatihan dan Pengujian Model ANN IDN dan Inverse

Pada tahap ini, akan dilakukan pelatihan terhadap model ANN IDN dan inverse. Pelatihan ini dilakukan untuk mendapatkan bobot terbaik yang akan digunakan sebagai bobot inisial pada model ANN DIC sehingga hasil yang diperoleh pada proses pengujian model ANN DIC bisa lebih maksimal. Kode program untuk pelatihan dan pengujian model ANN IDN dan inverse dengan input bilangan acak, sinyal step, ataupun sinusoidal secara umum sama, yang membedakan adalah dataset yang digunakan.

Pada proses pelatihan dan pengujian model ANN IDN dan inverse, digunakan arsitektur model ANN tanpa *feedback* seperti yang ditunjukkan pada gambar *3.2*. Objektif utama dari proses ini yaitu untuk mendapatkan bobot terbaik yang akan digunakan sebagai bobot inisial pada model ANN DIC.

3.3.1 Inisialisasi Data

Pertama-tama, perlu dilakukan pegimporan dataset yang telah di-*generate* pada tahap sebelumnya. Melalui sampel data yang ada di dalam dataset yang telah di-*generate* pada tahap sebelumnya, akan dilakukan proses *training*, validasi, dan *testing* terhadap model ANN. Model ANN yang telah dilatih diharapkan akan memiliki kemampuan untuk mengestimasi nilai keluaran dari *plant* berdasarkan informasi data masukan yang diberikan. Berikut adalah program untuk inisialisasi dataset yang akan digunakan dalam melatih model ANN yang dibangun dengan menggunakan algoritma *backpropagation*.

```
%% filename : tahap1.m

% Impor data
dataTable = readtable('dataset.xlsx', 'sheet', 'Sheet1');
```

3.3.2 Pemisahan Data Feature dan Target

Proses ini dilakukan dengan menentukan *feature* dan *target* dari dataset yang telah diimpor. *Feature* dari dataset meliputi kolom u(k), u(k-1), u(k-2), y(k-1), dan y(k-2). Adapun *target* meliputi data y(k).

```
% Penentuan feature dan target dari dataTable
feature = dataTable(:, 1:end-1);
target = dataTable(:, end);
```

3.3.3 Normalisasi

Proses selanjutnya, data *feature* dan *target* akan dinormalisasi dengan mengubah rentang nilai dalam dataset menjadi dari -1 hingga 1. Normalisasi ini sangat penting untuk dilakukan untuk meningkatkan kinerja dari jaringan dalam hal percepatan konvergensi ataupun peningkatan kemampuan dari model untuk menangkap pola dalam data.

```
% Normalisasi data feature
feature = normalize(feature, 'range', [-1, 1]);
target = normalize(target, 'range', [-1 1]);

% Mencari tahu jumlah baris dan kolom dari feature
[feature_num_row, feature_num_column] = size(feature);

% Mencari tahu jumlah baris dan kolom dari target
```

```
[target_num_row, target_num_column] = size(target);
```

3.3.4 Pemisahan Data Training, Validasi, dan Testing

Proses ini dilakukan dengan memisahkan data menjadi tiga bagian, yaitu data *training*, data validasi, dan data *testing*. Rasio pemisahan data akan bergantung dari jenis sinyal input yang digunakan.

```
% Pemisahan Data Training, Data Validasi, dan Data Testing
X_train = table2array(feature(1:5000, :));
y_train = table2array(target(1:5000, :));

X_val = table2array(feature(5000 + 1:7000, :));
y_val = table2array(target(5000 + 1:7000, :));

X_test = table2array(feature(7000 + 1:end, :));
y_test = table2array(target(7000 + 1:end, :));
```

3.3.5 Inisialisasi Model BPNN

Ini merupakan tahap inisialisasi variabel yang menggambarkan jumlah *input* neuron, *hidden* neuron, dan *output* neuron. Pada model yang dibuat, dilakukan inisialisasi variabel untuk 10 buah *hidden neuron* dengan lima buah input *neuron* dan satu buah neuron *output*.

```
% Model BPNN
% Inisialisasi jumlah input layer, hidden layer, dan output
layer
n = feature_num_column;             % input neuron
p = 10;                             % hidden neuron
m = target_num_column;              % output neuron
```

3.3.6 Penentuan Nilai Bobot dan Bias Menggunakan Metode Nguyen Widrow

Proses selanjutnya yaitu menentukan nilai bobot yang ada di antara *input layer* dan *hidden layer* dan bobot yang ada di antara *hidden layer* dan *output layer*. Adapun penentuan nilai bobot dan bias ini akan dilakukan dengan menggunakan metode Nguyen Widrow yang bertujuan untuk meredam perbedaan yang terlalu jauh dalam besaran bobot dan bias antar lapisan jaringan yang dimana ini dapat mencegah masalah gradien yang meledak atau menciut selama pelatihan.

```
% Penentuan nilai bobot secara random dalam skala -0.5 sampai
0.5
a = -0.5;
b = 0.5;

V = rand(n, p) + a;     % Bobot V
W = rand(p, m) - b;     % Bobot W

% Optimasi nilai bobot menggunakan metode Nguyen-Widrow
beta_V = 0.7 * (p) .^ (1/n);    % Nilai beta V
beta_W = 0.7 * (m) .^ (1/p);    % Nilai beta W

V_0j = -beta_V + (beta_V - (-beta_V)) .* rand(1,p);  %
Inisialisasi nilai bobot V0j
W_0k = -beta_W + (beta_W - (-beta_W) .* rand(1,m));  %
Inisialisasi nilai bobot W0k
```

```
V_j = sqrt(sum(V.^2));                                    %
Inisialisasi nilai bobot V_j
W_k = sqrt(sum(W.^2));                                    %
Inisialisasi nilai bobot W_k

Vij_new = (beta_V .* V) ./ V_j;                           %
Inisialisasi nilai bobot Vij_new
Wjk_new = (beta_W .* W) ./ W_k;                           %
Inisialisasi nilai bobot Wjk_new

V = Vij_new;                                        % Bobot Nguyen-
Widrow
W = Wjk_new;                                        % Bobot Nguyen-
Widrow

% Inisialisasi nilai lama
W_jk_lama = 0;
W_0k_lama = 0;
V_ij_lama = 0;
V_0j_lama = 0;
```

3.3.7 Penentuan Parameter Iterasi

Proses ini dikenal dengan inisialisasi *hyperparameter*, dimana parameter-parameter iterasi akan diinisialisasikan, mulai dari jumlah *epoch*, *error* minimum, iterasi *epoch*, *alfa* (laju pembelajaran/*learning rate*), hingga *miu* (laju pembelajaran momentum).

```
% Algoritma BPNN
% Penentuan nilai parameter iterasi
iterasi = 5000;                  % JUMLAH EPOCH
iter = 0;
iter_val = 0;
Ep_stop = 1;
alpha = 0.5;                     % LAJU PEMBELAJARAN
miu = 0.5;                       % LAJU PEMBELAJARAN MOMENTUM
```

3.3.8 Training

Algoritma ini dilakukan untuk men-*training* model dengan melakukan proses iterasi pada tiap *epoch* dan data *training*.

Proses Feedforward

Pada proses *feedforward*, akan dilakukan kalkulasi untuk menentukan keluaran dari *hidden layer* dengan menggunakan data *testing* dan bobot yang telah diinisialisasikan sebelumnya. Setelah diperoleh keluaran dari *hidden layer*, selanjutnya data tersebut akan dijadikan masukan pada fungsi aktivasi. Adapun jenis fungsi aktivasi yang digunakan pada model ini adalah fungsi *bipolar sigmoid*.

$$f(x) = \frac{2}{1 + e^{-x}} - 1$$

Perhitungan Error

Setelah diperoleh diperoleh keluaran dari output layer, akan dilakukan proses perhitungan error pada epoch. Adapun metode perhitungan error yang digunakan pada model ini yaitu dengan menggunakan pendekatan *Mean Squared Error*.

Algoritma Backpropagation

Pada algoritma *backpropagation*, akan dilakukan proses kalkulasi untuk memperoleh hasil turunan dari fungsi aktivasi yang ada pada output layer. Setelah hasil turunan dari fungsi aktivasi diperoleh, maka proses kalkulasi untuk mendapatkan delta pada output layer akan dijalankan.

Setelah delta pada output layer diperoleh, selanjutnya akan dilakkan proses *update* terhadap nilai bobot dan bias yang digunakan pada model.

```
% Training data
while Ep_stop == 1 && iter < iterasi
    iter = iter + 1;
    for a = 1:length(X_train)
        % ============== PROSES FEEDFORWARD ==============
        % Menghitung semua sinyal input dengan bobotnya
        z_inj = V_0j + X_train(a,:) * V;

        % Proses aktivasi menggunakan fungsi bipolar sigmoid
        for j = 1:p
            zj(1,j) = -1 + 2 / (1 + exp(-z_inj(1, j)));
        end

        % Menghitung semua sinyal input dengan bobotnya
        y_ink = W_0k + zj * W;
        % Proses aktivasi menggunakan fungsi bipolar sigmoid
        for r = 1:m
            y_k(a,r) = -1 + 2 / (1 + exp(-y_ink(1, r)));
        end

        % Menghitung nilai error
        E(1, a) = abs(y_train(a,:) - y_k(a));

        % Menghitung nilai total error kuadratik (MSE) data
validasi
        E_mse(1, a) = (y_train(a,:) - y_k(a)).^2;

        % ============== PROSES BACKPROPAGATION ==============
        % Menghitung informasi error
        do_k = (y_train(a,:) - y_k(a)) .* ((1 + y_k(a)) * (1 -
y_k(a)) / 2);
        % Menghitung besarnya koreksi bobot unit output
        w_jk = alpha * zj' * do_k + miu * W_jk_lama;
        % Menghitung besarnya koreksi bias output
        w_0k = alpha * do_k + miu * W_0k_lama;

        W_jk_lama = w_jk;
        W_0k_lama = w_0k;

        % Menghitung semua koreksi error
        do_inj  = do_k * W';
        % Menghitung nilai aktivasi koreksi error
        do_j    = do_inj .* ((1 + zj) .* (1 - zj) / 2);
        % Menghtiung koreksi bobot unit hidden
```

```
        v_ij    = alpha * X_train(a,:)' * do_j + miu *
V_ij_lama;
        % Menghitung koreksi error bias unit hidden
        v_0j    = alpha * do_j + miu * V_0j_lama;

        V_ij_lama = v_ij;
        V_0j_lama = v_0j;

        % Menng-update bobot dan bias untuk setiap unit output
        W = W + w_jk;
        W_0k = W_0k + w_0k;

        % Mengupdate bobot dan bias untuk setiap unit hidden
        V = V + v_ij;
        V_0j = V_0j + v_0j;
    end

MSE_train_total = sum(MSE_train) / length(MSE_train);
MSE_val_total = sum(MSE_val) / length(MSE_val);
```

3.3.9 Validasi

Proses validasi dilakukan selama proses *training* berlangsung, lebih tepatnya setiap proses *training* sudah dilakukan sebanyak 50 *epoch*. Algoritmanya hanya meliputi proses *feedforward* untuk menghitung nilai *mean squared error*.

```
% Melakukan validasi data dengan bobot yang diperoleh dari
training
    % setiap 10 epoch
    if mod(iter, 50) == 0
        while Ep_stop == 1 && iter_val ≤ iter
            iter_val = iter_val + 1;
            for a = 1:length(X_val)
                % ============ PROSES FEEDFORWARD ============
                % Menghitung semua sinyal input dengan bobotnya
                z_inj_val = V_0j + X_val(a,:) * V;

                % Proses aktivasi menggunakan fungsi bipolar
sigmoid
                for j = 1:p
                    zj_val(1,j) = -1 + 2 / (1 + exp(-
z_inj_val(1, j)));
                end

                % Menghitung semua sinyal input dengan bobotnya
                y_ink_val = W_0k + zj_val * W;
                % Proses aktivasi menggunakan fungsi bipolar
sigmoid
                for r = 1:m
                    y_k_val(a,r) = -1 + 2 / (1 + exp(-
y_ink_val(1, r)));
                end

                % Menghitung nilai error tiap epoch
                E_val(1, a) = abs(y_val(a,:) - y_k_val(a));
```

```
                % Menghitung nilai total error kuadratik (MSE)
                % data validasi
                E_mse_val(1, a) = (y_val(a, :) -
y_k_val(a)).^2;
            end
            % Menghitung nilai error validasi pada tiap epoch
            Ep_val(1, iter) = sum(E_val) / length(X_val);

            % Menghitung nilai total error kuadratik (MSE) pada
tiap epoch
            MSE_val(iter_val, 1) = sum(E_mse_val) /
length(X_val);
        end
    end

    % Ini masuk ke dalam while loop proses training karena
proses      training dan validasi dilakukan secara bebarengan
    % Menghitung nilai error training pada tiap epoch
    Ep(1, iter) = sum€ / length(X_train);

    % Menghitung nilai total error kuadratik (MSE) pada tiap
epoch
    MSE_train(iter, 1) = sum(E_mse) / length(X_train);

    acc_p(iter, 1) = 1 - MSE_train(iter, 1);
end

MSE_train_total = sum(MSE_train) / length(MSE_train);
MSE_val_total = sum(MSE_val) / length(MSE_val);
```

3.3.10 Generate File Excel dengan Bobot Terbaik dari Tahap 1

Proses ini dilakukan untuk meng-*generate* file *excel* yang berisi bobot terbaik dari proses pelatihan model ANN IDN dan inverse. Pada nantinya, bobot terbaik dari tahap ini akan digunakan sebagai bobot awal pada proses pelatihan dan pengujian model ANN DIC.

```
writematrix(V, 'bobot_V.xlsx','Sheet', 1);
writematrix(W, 'bobot_W.xlsx','Sheet', 1);
writematrix(V_0j, 'bias_V.xlsx','Sheet', 1);
writematrix(W_0k, 'bias_W.xlsx','Sheet', 1);
```

3.3.11 Testing

Algoritma ini dilakukan untuk melakukan proses *testing* pada model yang telah dibangun dengan melakukan proses *feedforward* pada setiap sampel data *testing*. Dengan dilakukannya proses *testing* ini, akan diketahui nilai *mean squared error* antara data sebenarnya (keluaran *plant*) dengan data prediksi keluaran model ANN.

```
% Melakukan testing
E_test = zeros(length(X_test), 1);
for a = 1:length(X_test)
    % ==================== PROSES FEEDFORWARD
====================
    z_inj_test = X_test(a,:) * V + V_0j;
```

```
    % Proses aktivasi menggunakan sigmoid
    for j=1:p
        zj_test(1,j) = -1 + 2 / (1 + exp(-z_inj_test(1,j)));
%Aktivasi sigmoid
    end

    y_ink_test = zj_test * W + W_0k;

    for k=1:m
        y_k_test(a,k) = -1 + 2 / (1 + exp(-y_ink_test(1,k)));
%Aktivasi sigmoid
    end

    for j = 1:m
        predict(a,j) = y_k_test(j);
    end

    %Menghitung nilai error
    E_test(a, 1) = abs(y_test(a,:) - y_k_test(a));

    % MSE
    E_mse_test(a, 1) = (y_test(a, :) - y_k_test(a)).^2;

end

Ep_test = sum(E_test) / length(X_test);

MSE_test = sum(E_mse_test) / length(X_test);
```

3.3.12 Evaluasi

Pada tahap ini, akan ditampilkan grafik *mean squared error* pada proses *data training* dan validasi, grafik error yang didapat selama proses data *training*, visualisasi *scatter* antara data keluaran *plant* dengan data prediksi keluaran model ANN selama proses *training*, validasi, dan *testing*.

```
figure;
plot(MSE_train, 'r-', 'Linewidth', 1);
hold on
plot(MSE_val, 'b-', 'LineWidth', 1);
hold off
ylabel('Total error kuadratik'); xlabel('Epoch');
legend("MSE data training", "MSE data validasi");

figure;
plot(Ep, 'r-', 'Linewidth', 1);
ylabel('Total Error'); xlabel('Epoch');
legend("Error data training");

x = 1:40000;
y = 1:5000;
z = 1:5000;

figure;
scatter(x, y_train, 'Linewidth', 1.5);
```

```
hold on
scatter(x, y_k, 'Linewidth', 1.5);
legend("Output Plant Training", "Output ANN Training");
hold off

figure;
scatter(y, y_val, 'Linewidth', 1.5);
hold on
scatter(y, y_k_val, 'Linewidth', 1.5);
legend("Output Plant Validasi", "Output ANN Validasi");
hold off

figure;
scatter(z, y_test, 'Linewidth', 1.5);
hold on
scatter(z, y_k_test, 'Linewidth', 1.5);
legend("Output Plant Testing", "Output ANN Testing");
hold off


disp("Error final Data Training = " + (Ep(end)));
disp("Error final Data Validasi = " + (Ep_val(end)));
disp("Error final Data Testing = " + (Ep_test));

disp("MSE final Data Training = " + (MSE_train(end)));
disp("MSE final Data Validasi = " + (MSE_val(end)));

disp("MSE total Data Training = " + (MSE_train_total));
disp("MSE total Data Validasi = " + (MSE_val_total));
disp("MSE total Data Testing = " + (MSE_test));
```

3.4 Pelatihan dan Pengujian Model ANN DIC

Setelah dilakukan pelatihan dan pengujian terhadap model ANN IDN dan inverse, sistem dapat digabungkan menjadi model ANN DIC seperti yang ditunjukkan pada gambar *3.4* dan *3.5*. Pada laporan ini, hanya akan dilakukan proses pelatihan dan pengujian terhadap model ANN DIC dengan konfigurasi lup terbuka saja.

Sama seperti sebelumnya, kode program yang digunakan pada proses pelatihan dan pengujian model ANN DIC dengan input bilangan acak, sinyal step, ataupun sinyal sinusoidal akan sama, yang membedakan adalah dataset yang digunakan. Objektif utama dari proses pelatihan dan pengujian model ANN DIC ini yaitu untuk mengetahui bagaimana performa dari model ANN DIC dalam penerapannya sebagai pengendali dari suatu sistem

3.4.1 Inisialisasi Data

Pertama-tama, perlu dilakukan pegimporan dataset yang telah di-*generate* pada tahap sebelumnya. Melalui sampel data yang ada di dalam dataset yang telah di-*generate* pada tahap sebelumnya, akan dilakukan proses *training*, validasi, dan *testing* terhadap model ANN. Model ANN yang telah dilatih diharapkan akan memiliki kemampuan untuk mengestimasi nilai keluaran dari *plant* berdasarkan informasi data masukan yang diberikan. Berikut adalah program untuk inisialisasi

dataset yang akan digunakan dalam melatih model ANN yang dibangun dengan menggunakan algoritma *backpropagation*.

```
% PERCOBAAN OPEN LOOP
%                    --------------            --------------
%                   |              |          |              |
%                   |   Inverse    |   u(k)   |              |
%     y(k) ------ > |    Plant     | ------ > |    Plant     | --> y'(k)
%             |---> |              |    | |-> |              |---------|
%             |     |              |    | |   |              |         |
%             |      --------------     | |    --------------          |
%             |                         | |                            |
%             |-------------------------| |----------------------------|
|
%


clc;
clear;

% Impor data
dataTable_inverse = readtable('dataset_inverse.xlsx', 'sheet',
'Sheet1');
```

3.4.2 Pemisahan Data Feature dan Target

Proses ini dilakukan dengan menentukan *feature* dan *target* dari dataset yang telah diimpor.

```
% DATA PREPROCESSING
% Penentuan feature dan target dari dataTable
feature_inverse = dataTable_inverse(:, 1:end-1);
target_inverse = dataTable_inverse(:, end);
```

3.4.3 Normalisasi

Proses selanjutnya, data *feature* dan *target* akan dinormalisasi dengan mengubah rentang nilai dalam dataset menjadi dari -1 hingga 1. Normalisasi ini sangat penting untuk dilakukan untuk meningkatkan kinerja dari jaringan dalam hal percepatan konvergensi ataupun peningkatan kemampuan dari model untuk menangkap pola dalam data.

```
% Normalisasi data feature
feature_inverse = normalize(feature_inverse, 'range', [-0.5, 0.5]);
target_inverse = normalize(target_inverse, 'range', [-0.5, 0.5]);

% Inisialisasi Variabel untuk Menampung nilai Y_referensi & Y_model

% Mencari tahu jumlah baris dan kolom dari feature
[feature_num_row_inverse, feature_num_column_inverse] =
size(feature_inverse);

% Mencari tahu jumlah baris dan kolom dari target
[target_num_row_inverse, target_num_column_inverse] =
size(target_inverse);
```

3.4.4 Pemisahan Data Training, Validasi, dan Testing

Proses ini dilakukan dengan memisahkan data menjadi tiga bagian, yaitu data *training*, data validasi, dan data *testing*. Rasio pemisahan data akan bergantung dari jenis input yang digunakan.

```
% Pemisahan Data Training, Data Validasi, dan Data Testing
X_train_inverse = table2array(feature_inverse(1:45000, :));
X_train = table2array(feature_inverse(1:45000, :));
y_train_inverse = table2array(target_inverse(1:45000, :));
y_train = table2array(feature_inverse(1:45000, 4));


X_test_inverse = table2array(feature_inverse(45000 + 1:end, :));
X_test = table2array(feature_inverse(45000 + 1:end, :));
y_test_inverse = table2array(target_inverse(45000 + 1:end, :));
y_test = table2array(feature_inverse(45000 + 1:end, 4));


% Inisisalisasi variabel penampung data aktual dan estimasi dari
model
vektor_u_ANN_train = zeros(length(X_train_inverse), 1);
vektor_u_ref_train = table2array(target_inverse(1:45000, :));
vektor_y_ANN_train = zeros(length(X_train_inverse), 1);
vektor_y_ref_train = table2array(feature_inverse(1:45000, 4));

vektor_u_ANN_test = zeros(length(X_test_inverse), 1);
vektor_u_ref_test = table2array(target_inverse(45000 + 1:end, :));
vektor_y_ANN_test = zeros(length(X_test_inverse), 1);
vektor_y_ref_test = table2array(feature_inverse(45000 + 1:end, 4));
```

3.4.5 Inisialisasi Model BPNN

Ini merupakan tahap inisialisasi variabel yang menggambarkan jumlah *input* neuron, *hidden* neuron, dan *output* neuron. Pada model yang dibuat, dilakukan inisialisasi variabel untuk 10 buah *hidden neuron* dengan lima buah input *neuron* dan satu buah neuron *output*.

```
% Model BPNN
% Inisialisasi jumlah input layer, hidden layer, dan output layer
model ANN
% Inverse
n_inverse = feature_num_column_inverse;             % input layer
p_inverse = 10;                                     % hidden layer
m_inverse = target_num_column_inverse;              % output layer

% Inisialisasi jumlah input layer, hidden layer, dan output layer
model ANN
% Plant
n = feature_num_column_inverse - 1;                                 %
input layer
p = 10;                                             % hidden layer
m = target_num_column_inverse;
```

3.4.6 Impor Nilai Bobot dari Tahap Sebelumnya

Tahap ini dilakukan dengan mengimpor bobot dan bias terbaik yang diperoleh melalui proses pelatihan model ANN IDN dan inverse yang dilakukan pada tahap sebelumnya.

```
% Impor bobot dan bias yang sudah dilatih dari training model ANN
Plant dan
% model ANN Inverse
V = readmatrix("bobot_V.xlsx");
W = readmatrix("bobot_W.xlsx");
V_0j = readmatrix("bias_V.xlsx");
W_0k = readmatrix("bias_W.xlsx");

V_inverse = readmatrix("bobot_V_inverse.xlsx");
W_inverse = readmatrix("bobot_W_inverse.xlsx");
V_0j_inverse = readmatrix("bias_V_inverse.xlsx");
W_0k_inverse = readmatrix("bias_W_inverse.xlsx");

% Inisialisasi nilai lama
W_jk_lama = 0;
W_0k_lama = 0;
V_ij_lama = 0;
V_0j_lama = 0;
```

3.4.7 Penentuan Parameter Iterasi

Proses ini dikenal dengan inisialisasi *hyperparameter*, dimana parameter-parameter iterasi akan diinisialisasikan, mulai dari jumlah *epoch*, *error* minimum, iterasi *epoch*, *alfa* (laju pembelajaran/*learning rate*), hingga *miu* (laju pembelajaran momentum).

```
% Algoritma BPNN
% Penentuan nilai parameter iterasi
iterasi = 5000;                 % JUMLAH EPOCH
iter = 0;
iter_val = 0;
Ep_stop = 1;
alpha = 0.5;                    % LAJU PEMBELAJARAN
miu = 0.5;                      % LAJU PEMBELAJARAN MOMENTUM
```

3.4.8 Training

Algoritma ini dilakukan untuk men-*training* model dengan melakukan proses iterasi pada tiap *epoch* dan data *training*. Adapun proses *training* ini tetap dilakukan oleh penulis karena dengan adanya proses *training* ini, akan diperoleh bobot yang lebih optimal dalam melakukan estimasi nilai pada model ANN yang ada pada DIC. Dalam hal ini, nilai bobot dari model akan menjadi lebih baik jika dibandingkan dengan nilai bobot yang diimpor dari proses pelatihan model ANN IDN dan inverse yang dilakukan secara independen. Terbukti, dengan melakukan proses *training* pada model ANN DIC ini, didapatkan nilai MSE yang lebih kecil jika dibandingkan dengan pengujian model ANN DIC tanpa proses *training* dimana nilai bobot diperoleh dari proses pelatihan model ANN IDN dan inverse secara independen.

Proses Feedforward

Pada proses *feedforward*, akan dilakukan kalkulasi untuk menentukan keluaran dari *hidden layer* dengan menggunakan data *testing* dan bobot yang telah diinisialisasikan sebelumnya. Setelah diperoleh keluaran dari *hidden layer*, selanjutnya data tersebut akan dijadikan masukan pada fungsi aktivasi. Adapun jenis fungsi aktivasi yang digunakan pada model ini adalah fungsi *bipolar sigmoid*.

$$f(x) = \frac{2}{1 + e^{-x}} - 1$$

Perhitungan Error

Setelah diperoleh diperoleh keluaran dari output layer, akan dilakukan proses perhitungan error pada epoch. Adapun metode perhitungan error yang digunakan pada model ini yaitu dengan menggunakan pendekatan *Mean Squared Error*.

Algoritma Backpropagation

Pada algoritma *backpropagation*, akan dilakukan proses kalkulasi untuk memperoleh hasil turunan dari fungsi aktivasi yang ada pada output layer. Setelah hasil turunan dari fungsi aktivasi diperoleh, maka proses kalkulasi untuk mendapatkan delta pada output layer akan dijalankan.

Setelah delta pada output layer diperoleh, selanjutnya akan dilakkan proses *update* terhadap nilai bobot dan bias yang digunakan pada model.

```
% Training data
while Ep_stop == 1 && iter < iterasi
    iter = iter + 1;
    for a = 3:length(X_train_inverse)
        % ================== PROSES FEEDFORWARD ==================
        % ======================= INVERSE ========================
        % Menghitung semua sinyal input dengan bobotnya
        z_inj_inverse_train = V_0j_inverse + X_train_inverse(a,:) *
V_inverse;

        % Proses aktivasi menggunakan fungsi bipolar sigmoid
        for j = 1:p_inverse
            zj_inverse_train(1,j) = -1 + 2 / (1 + exp(-
z_inj_inverse_train(1, j)));
        end

        % Menghitung semua sinyal input dengan bobotnya
        y_ink_inverse_train = W_0k_inverse + zj_inverse_train *
W_inverse;
        % Proses aktivasi menggunakan fungsi bipolar sigmoid
        for r = 1:m_inverse
            y_k_inverse_train(a,r) = -1 + 2 / (1 + exp(-
y_ink_inverse_train(1, r)));
        end

        vektor_u_ANN_train(a, 1) = y_k_inverse_train(a, 1);

        % Menghitung nilai error
```

```
        E_inverse_train(1, a) = abs(y_train_inverse(a,:) -
y_k_inverse_train(a));

        % Menghitung nilai total error kuadratik (MSE) data validasi
        E_mse_inverse_train(1, a) = (y_train_inverse(a, :) -
y_k_inverse_train(a)).^2;

        % ================== PROSES FEEDFORWARD ==================
        % ======================== PLANT ========================
        if a == 3
            feature_train = [vektor_u_ANN_train(a, 1) X_train(a, 1)
X_train(a, 2) X_train(a, 5) X_train(a, 6)];
        elseif a == 4
            feature_train = [vektor_u_ANN_train(a, 1)
vektor_u_ANN_train(a - 1, 1) X_test(a, 2) X_train(a, 5) X_train(a,
6)];
        else
            feature_train = [vektor_u_ANN_train(a, 1)
vektor_u_ANN_train(a - 1, 1) vektor_u_ANN_train(a - 2, 1) X_train(a,
5) X_train(a, 6)];
        end

        % Menghitung semua sinyal input dengan bobotnya
        z_inj_train = V_0j + feature_train * V;

        % Proses aktivasi menggunakan fungsi bipolar sigmoid
        for j = 1:p
            zj_train(1,j) = -1 + 2 / (1 + exp(-z_inj_train(1, j)));
        end

        % Menghitung semua sinyal input dengan bobotnya
        y_ink_train = W_0k + zj_train * W;
        % Proses aktivasi menggunakan fungsi bipolar sigmoid
        for r = 1:m
            y_k_train(a,r) = -1 + 2 / (1 + exp(-y_ink_train(1, r)));
        end

        vektor_y_ANN_train(a, 1) = y_k_train(a, 1);

        % Menghitung nilai error
        E_train(1, a) = abs(y_train(a,:) - y_k_train(a));

        % Menghitung nilai total error kuadratik (MSE) data validasi
        E_mse_train(1, a) = (y_train(a, :) - y_k_train(a)).^2;


        if a < length(X_train_inverse)
            X_train_inverse(a + 1, 1) = vektor_u_ANN_train(a, 1);
            X_train(a + 1, 5) = vektor_y_ANN_train(a, 1);
            X_train_inverse(a + 1, 2) = vektor_u_ANN_train(a - 1, 1);
            X_train_inverse(a + 1, 3) = vektor_u_ANN_train(a - 2, 1);
            X_train(a + 1, 6) = vektor_y_ANN_train(a - 1, 1);
        end
```

```
        % =============== PROSES BACKPROPAGATION =================
        % ====================== INVERSE ========================
        % Menghitung informasi error
        do_k_inverse = (y_train_inverse(a,:) - y_k_inverse_train(a))
.* ((1 + y_k_inverse_train(a)) * (1 - y_k_inverse_train(a)) / 2);
        % Menghitung besarnya koreksi bobot unit output
        w_jk_inverse = alpha * zj_inverse_train' * do_k_inverse + miu
* W_jk_lama_inverse;
        % Menghitung besarnya koreksi bias output
        w_0k_inverse = alpha * do_k_inverse + miu *
W_0k_lama_inverse;

        W_jk_lama_inverse = w_jk_inverse;
        W_0k_lama_inverse = w_0k_inverse;

        % Menghitung semua koreksi error
        do_inj_inverse  = do_k_inverse * W_inverse';
        % Menghitung nilai aktivasi koreksi error
        do_j_inverse    = do_inj_inverse .* ((1 + zj_inverse_train)
.* (1 - zj_inverse_train) / 2);
        % Menghtiung koreksi bobot unit hidden
        v_ij_inverse    = alpha * X_train_inverse(a,:)' *
do_j_inverse + miu * V_ij_lama_inverse;
        % Menghitung koreksi error bias unit hidden
        v_0j_inverse    = alpha * do_j_inverse + miu *
V_0j_lama_inverse;

        V_ij_lama_inverse = v_ij_inverse;
        V_0j_lama_inverse = v_0j_inverse;

        % Menng-update bobot dan bias untuk setiap unit output
        W_inverse = W_inverse + w_jk_inverse;
        W_0k_inverse = W_0k_inverse + w_0k_inverse;

        % Mengupdate bobot dan bias untuk setiap unit hidden
        V_inverse = V_inverse + v_ij_inverse;
        V_0j_inverse = V_0j_inverse + v_0j_inverse;

        % =============== PROSES BACKPROPAGATION ================
        % ======================= PLANT =========================
        % Menghitung informasi error
        do_k = (y_train(a,:) - y_k_train(a)) .* ((1 + y_k_train(a)) *
(1 - y_k_train(a)) / 2);
        % Menghitung besarnya koreksi bobot unit output
        w_jk = alpha * zj_train' * do_k + miu * W_jk_lama;
        % Menghitung besarnya koreksi bias output
        w_0k = alpha * do_k + miu * W_0k_lama;

        W_jk_lama = w_jk;
        W_0k_lama = w_0k;

        % Menghitung semua koreksi error
        do_inj  = do_k * W';
```

```
        % Menghitung nilai aktivasi koreksi error
        do_j    = do_inj .* ((1 + zj_train) .* (1 - zj_train) / 2);
        % Menghtiung koreksi bobot unit hidden
        v_ij    = alpha * feature_train' * do_j + miu * V_ij_lama;
        % Menghitung koreksi error bias unit hidden
        v_0j    = alpha * do_j + miu * V_0j_lama;

        V_ij_lama = v_ij;
        V_0j_lama = v_0j;

        % Menng-update bobot dan bias untuk setiap unit output
        W = W + w_jk;
        W_0k = W_0k + w_0k;

        % Mengupdate bobot dan bias untuk setiap unit hidden
        V = V + v_ij;
        V_0j = V_0j + v_0j;
    end

    % Menghitung nilai error training pada tiap epoch
    Ep_inverse(1, iter) = sum(E_train) / length(X_train_inverse);
    Ep(1, iter) = sum(E_inverse_train) / length(X_train_inverse);

    % Menghitung nilai total error kuadratik (MSE) pada tiap epoch
    MSE_inverse_train(iter, 1) = sum(E_mse_inverse_train) /
length(X_train_inverse);
    MSE_train(iter, 1) = sum(E_mse_train) / length(X_train_inverse);
end

MSE_inverse_train_total = sum(MSE_inverse_train) /
length(MSE_inverse_train);
MSE_train_total = sum(MSE_train) / length(MSE_train);
```

3.4.9 Testing

Algoritma ini dilakukan untuk melakukan proses *testing* pada model yang telah dibangun dengan melakukan proses *feedforward* pada setiap sampel data *testing*. Dengan dilakukannya proses *testing* ini, akan diketahui nilai *mean squared error* antara data sebenarnya (keluaran *plant*) dengan data prediksi keluaran model ANN.

```
% Testing Data
for a = 3:length(X_test_inverse)
    % ================== PROSES FEEDFORWARD ==================
    % ======================= INVERSE ========================
    % Menghitung semua sinyal input dengan bobotnya
    z_inj_inverse_test = V_0j_inverse + X_test_inverse(a,:) *
V_inverse;

    % Proses aktivasi menggunakan fungsi bipolar sigmoid
    for j = 1:p_inverse
        zj_inverse_test(1,j) = -1 + 2 / (1 + exp(-
z_inj_inverse_test(1, j)));
    end
```

```
    % Menghitung semua sinyal input dengan bobotnya
    y_ink_inverse_test = W_0k_inverse + zj_inverse_test *
W_inverse;
    % Proses aktivasi menggunakan fungsi bipolar sigmoid
    for r = 1:m_inverse
        y_k_inverse_test(a,r) = -1 + 2 / (1 + exp(-
y_ink_inverse_test(1, r)));
    end

    vektor_u_ANN_test(a, 1) = y_k_inverse_test(a, 1);

    % Menghitung nilai error
    E_inverse_test(1, a) = abs(y_test_inverse(a,:) -
y_k_inverse_test(a));

    % Menghitung nilai total error kuadratik (MSE) data
validasi
    E_mse_inverse_test(1, a) = (y_test_inverse(a, :) -
y_k_inverse_test(a)).^2;

    % ================== PROSES FEEDFORWARD ==================
    % ======================== PLANT ========================
    if a == 3
        feature_test = [vektor_u_ANN_test(a, 1) X_test(a, 1)
X_test(a, 2) X_test(a, 5) X_test(a, 6)];
    elseif a == 4
        feature_test = [vektor_u_ANN_test(a, 1)
vektor_u_ANN_test(a - 1, 1) X_test(a, 2) X_test(a, 5) X_test(a,
6)];
    else
        feature_test = [vektor_u_ANN_test(a, 1)
vektor_u_ANN_test(a - 1, 1) vektor_u_ANN_test(a - 2, 1)
X_test(a, 5) X_test(a, 6)];
    end

    % Menghitung semua sinyal input dengan bobotnya
    z_inj_test = V_0j + feature_test * V;

    % Proses aktivasi menggunakan fungsi bipolar sigmoid
    for j = 1:p
        zj_test(1,j) = -1 + 2 / (1 + exp(-z_inj_test(1, j)));
    end

    % Menghitung semua sinyal input dengan bobotnya
    y_ink_test = W_0k + zj_test * W;
    % Proses aktivasi menggunakan fungsi bipolar sigmoid
    for r = 1:m
        y_k_test(a,r) = -1 + 2 / (1 + exp(-y_ink_test(1, r)));
    end

    vektor_y_ANN_test(a, 1) = y_k_test(a, 1);

    % Menghitung nilai error
```

```
    E_test(1, a) = abs(y_test(a,:) - y_k_test(a));

    % Menghitung nilai total error kuadratik (MSE) data
validasi
    E_mse_test(1, a) = (y_test(a, :) - y_k_test(a)).^2;



    if a < length(X_test_inverse)

    end

    if a < length(X_test_inverse)

    end

    if a < length(X_train_inverse)

    end


    if a < length(X_train_inverse)
      X_test_inverse(a + 1, 1) = vektor_u_ANN_test(a, 1);
      X_test_inverse(a + 1, 2) = vektor_u_ANN_test(a - 1, 1);
      X_inverse_inverse(a + 1, 3) = vektor_u_ANN_test(a - 2,
1);
      X_test(a + 1, 5) = vektor_y_ANN_test(a, 1);
      X_test(a + 1, 6) = vektor_y_ANN_test(a - 1, 1);
    end

end

Ep_inverse_test = sum(E_inverse_test) / length(X_test_inverse);
MSE_inverse_test = sum(E_mse_inverse_test) /
length(X_test_inverse);
Ep_test = sum(E_test) / length(X_test_inverse);
MSE_test = sum(E_mse_test) / length(X_test_inverse);
```

3.4.10 Evaluasi

Pada tahap ini, akan ditampilkan grafik *mean squared error* pada proses *data training*, grafik error yang didapat selama proses data *training*, visualisasi *plot* dan *scatter* antara data keluaran *plant* dengan data prediksi keluaran model ANN selama proses *training* dan *testing*.

```
figure;
plot(vektor_y_ref_test, 'r-', 'Linewidth', 1);
hold on
plot(vektor_y_ANN_test, 'b-', 'LineWidth', 1);
hold off
ylabel('Value'); xlabel('Sample Data');
legend("y(k) referensi testing", "y(k) keluaran testing");
```

```
figure;
plot(vektor_u_ref_test, 'r-', 'Linewidth', 1);
hold on
plot(vektor_u_ANN_test, 'b-', 'LineWidth', 1);
hold off
ylabel('Value'); xlabel('Sample Data');
legend("u(k) referensi testing", "u(k) keluaran testing");

figure;
plot(vektor_y_ref_train, 'r-', 'Linewidth', 1);
hold on
plot(vektor_y_ANN_train, 'b-', 'LineWidth', 1);
hold off
ylabel('Value'); xlabel('Sample Data');
legend("y(k) referensi traning", "y(k) keluaran training");

figure;
plot(vektor_u_ref_train, 'r-', 'Linewidth', 1);
hold on
plot(vektor_u_ANN_train, 'b-', 'LineWidth', 1);
hold off
ylabel('Value'); xlabel('Sample Data');
legend("u(k) referensi training", "u(k) keluaran training");


x = 1:5000;
y = 1:45000;

figure;
scatter(x, vektor_y_ref_test, 'Linewidth', 1);
hold on
scatter(x, vektor_y_ANN_test, 'Linewidth', 1);
legend("y(k) referensi testing", "y(k) keluaran testing");
hold off


figure;
scatter(x, vektor_u_ref_test, 'Linewidth', 1);
hold on
scatter(x, vektor_u_ANN_test, 'Linewidth', 1);
legend("u(k) referensi testing", "u(k) keluaran testing");
hold off


figure;
scatter(y, vektor_y_ref_train, 'Linewidth', 1);
hold on
scatter(y, vektor_y_ANN_train, 'Linewidth', 1);
legend("y(k) referensi training", "y(k) keluaran training");
hold off
```

```
figure;
scatter(y, vektor_u_ref_train, 'Linewidth', 1);
hold on
scatter(y, vektor_u_ANN_train, 'Linewidth', 1);
legend("u(k) referensi training", "u(k) keluaran training");
hold off

disp("MSE Inverse train = " + MSE_inverse_train_total);
disp("MSE train = " + MSE_train_total);
disp("MSE Inverse test = " + MSE_inverse_test);
disp("MSE test = " + MSE_test);
```

BAB IV
SIMULASI DAN ANALISIS HASIL

4.1 Hasil Pengujian Model dan Pembahasan

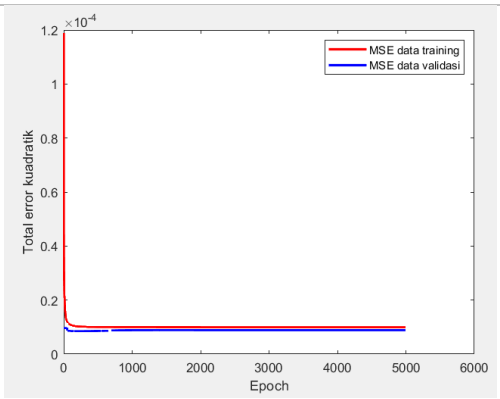
Proses pengujian model DIC akan dilakukan dengan menggunakan tiga buah variasi input, yakni input data *random*, sinyal step, serta sinyal sinusoidal. Sebelum dilakukan proses pengujian model DIC, perlu diperoleh terlebih dahulu bobot terbaik dari pengujian model ANN IDN dan inverse dengan menggunakan tiga buah variasi input yang berbeda tersebut. Bobot terbaik tersebut akan digunakan sebagai bobot inisial pada model DIC. Adapun pada proses pengujian model DIC, akan dilakukan juga proses *training* sehingga akan dihasilkan bobot terbaik pada model ANN IDN ataupun ANN Inverse setelah keduanya digabung menjadi model DIC. Berikut akan dijabarkan hasil dari setiap tahapan.

4.1.1 Random Input

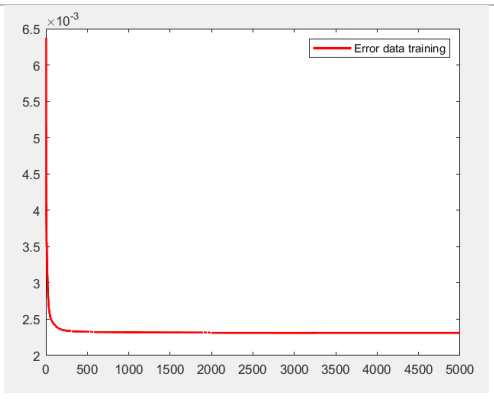
4.1.1.1 Pelatihan dan Pengujian model ANN IDN

Pada proses *training*, *validasi*, dan *testing* model ANN IDN , digunakan beberapa parameter pada model ANN yang di antaranya adalah sebagai berikut.

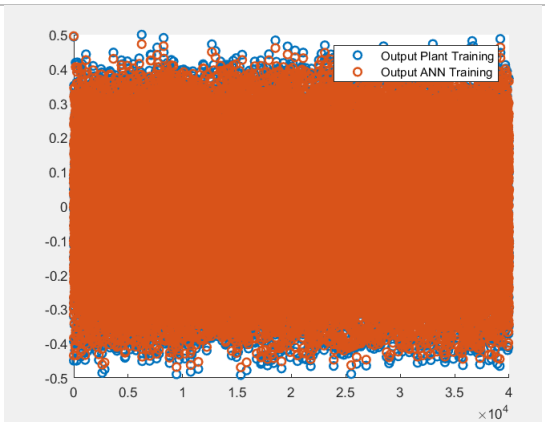
- Jumlah *hidden* neuron = 10
- Jumlah iterasi = 5000
- Laju pembelajaran = 0.5
- Laju pembelajaran momentum = 0.5
- Jumlah data *training* = 40000
- Jumlah data *validasi* = 5000
- Jumlah data *testing* = 5000



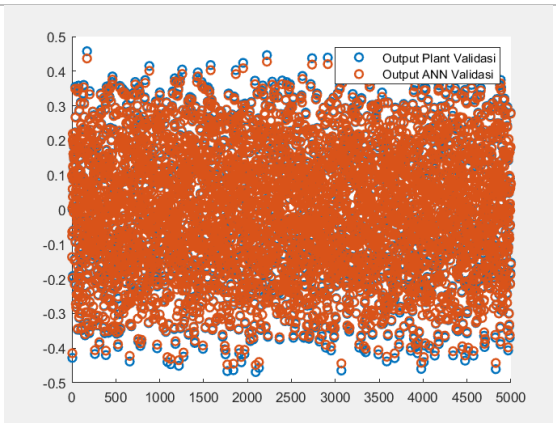
Gambar 4.1 Grafik Total Error Kuadratik Data *Training* dan Validasi



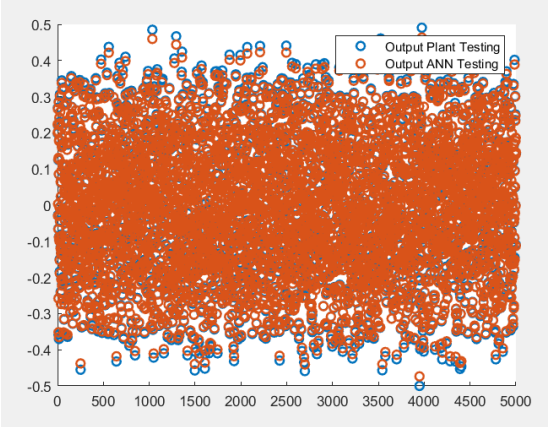
Gambar 4.2 Grafik Error Data Training



Gambar 4.3 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN (Data *Training*)



Gambar 4.4 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN (Data Validasi)



Gambar 4.5 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN (Data *Testing*)

```
Error final Data Training = 0.002314
Error final Data Validasi = 0.0020968
Error final Data Testing = 0.0021771
MSE final Data Training = 9.9165e-06
MSE final Data Validasi = 8.8591e-06
MSE total Data Training = 1.0067e-05
MSE total Data Validasi = 8.8282e-06
MSE total Data Testing = 9.6945e-06
```

VarName1	VarName2	VarName3	VarName4	VarName5	VarName6	VarName7	VarName8	VarName9	VarName10
Number	Number	Number	Number	Number	Number	Number	Number	Number	Number
0.3489	0.3001	1.4466	-0.5026	-0.8407	-0.1810	-0.2688	0.2868	0.4353	-0.2411
0.1330	0.4993	-0.3024	-0.3291	1.5239	0.9383	-1.2781	0.0395	0.8040	-0.2089
-0.1745	0.1303	0.9734	-0.2425	-0.5663	1.7850	-0.6958	-0.4154	0.4567	0.3106
-0.1959	0.2097	1.3719	-0.0022	0.3008	-3.0792	-3.4409	-0.7515	1.1655	0.5092
0.5801	0.3129	1.1859	-0.2009	-0.6894	2.3040	-0.7474	-0.0974	-1.0524	0.1329

Gambar 4.6 Bobot V Terbaik

VarName1
Number
-0.6950
-0.6536
-0.6585
0.6611
-0.6264
0.1847
0.7202
-0.6587
-0.5804
0.6520

Gambar 4.7 Bobot W terbaik

	VarName1	VarName2	VarName3	VarName4	VarName5	VarName6	VarName7	VarName8	VarName9	VarName10
	Number	Number	Number	Number	Number	Number	Number	Number	Number	Number
1	0.1760	-0.1069	-0.1940	0.1569	0.0606	-0.5103	0.6524	0.0984	0.1324	-0.0268

Gambar 4.8 Bias V terbaik

VarName1
Number
-0.0541

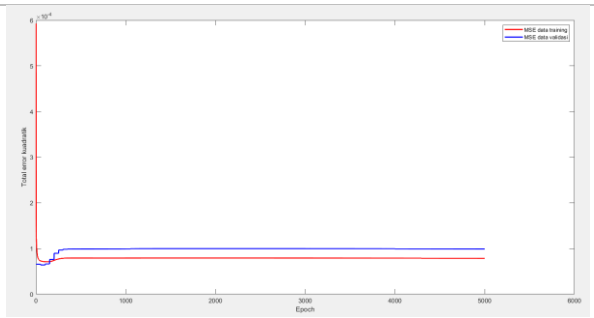
Gambar 4.9 Bias W terbaik

4.1.1.2 Pelatihan dan Pengujian model ANN Inverse

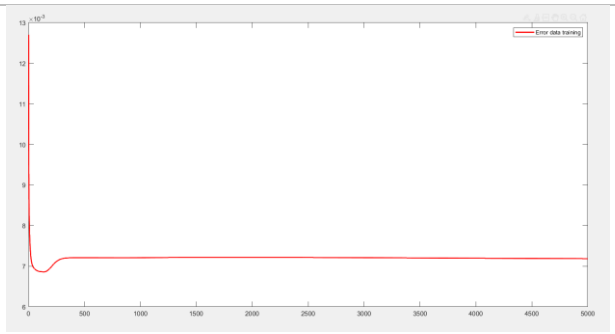
Pada proses *training*, *validasi*, dan *testing* model ANN inverse, digunakan beberapa parameter pada model ANN yang di antaranya adalah sebagai berikut.

- Jumlah *hidden* neuron = 10
- Jumlah iterasi = 5000
- Laju pembelajaran = 0.5
- Laju pembelajaran momentum = 0.5

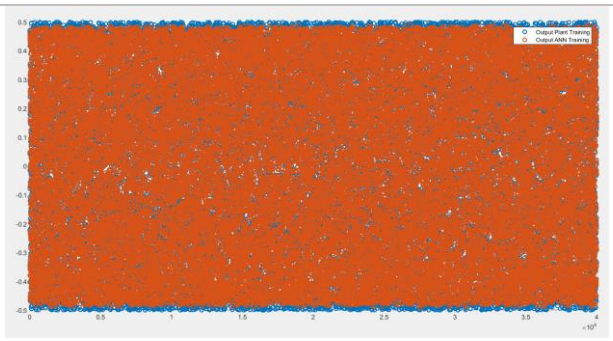
- Jumlah data *training* = 40000
- Jumlah data *validasi* = 5000
- Jumlah datai *testing* = 5000



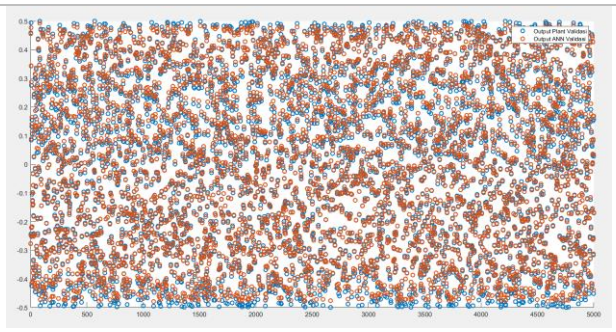
Gambar 5.0 Grafik Total Error Kuadratik Data *Training* dan Validasi



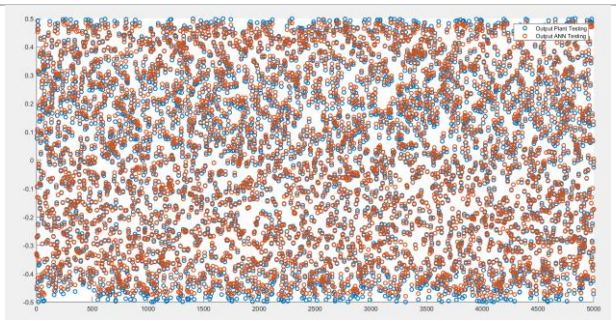
Gambar 5.1 Grafik Error Data Training



Gambar 5.2 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN (Data *Training*)



Gambar 5.3 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN (Data Validasi)



Gambar 5.4 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN (Data *Testing*)

```
Error final Data Training = 0.0071863
Error final Data Validasi = 0.0080278
Error final Data Testing = 0.0081668
MSE final Data Training = 7.8511e-05
MSE final Data Validasi = 9.8971e-05
MSE total Data Training = 7.8855e-05
MSE total Data Validasi = 9.7994e-05
MSE total Data Testing = 9.9937e-05
```

VarName1	VarName2	VarName3	VarName4	VarName5	VarName6	VarName7	VarName8	VarName9	VarName10
Number	Number	Number	Number	Number	Number	Number	Number	Number	Number
-1.8781	0.8377	-0.7755	-1.4662	0.8635	-0.4362	-0.7716	0.7576	-0.7765	-0.0538
0.9539	0.1805	-0.3011	0.1327	0.0127	0.0718	-0.0145	0.0172	0.1901	0.3781
-0.0204	0.0836	0.0294	0.2916	0.1810	2.8696	-0.2513	0.2638	-0.3311	0.3444
-0.2929	1.4830	0.0685	-1.4054	1.4572	-0.5098	-1.4540	1.4535	-1.4608	1.4818
-4.6330	0.4101	5.1875	-1.9354	0.5417	-0.6294	-0.3334	0.3023	0.3725	0.0780
-1.8684	0.1145	-0.5209	1.6488	-0.0297	0.3573	-0.1059	0.1327	-0.4099	0.4258

Gambar 5.5 Bobot V Inverse Terbaik

VarName1
Number
0.3099
-0.8693
-0.3887
0.8454
-0.8314
0.3686
0.8313
-0.8313
0.8837
-0.7895

Gambar 5.6 Bobot W Inverse terbaik

VarName1	VarName2	VarName3	VarName4	VarName5	VarName6	VarName7	VarName8	VarName9	VarName10
Number	Number	Number	Number	Number	Number	Number	Number	Number	Number
0.3828	0.2249	0.0786	0.1464	0.0674	0.0183	-0.0720	0.0726	0.3019	-0.1852

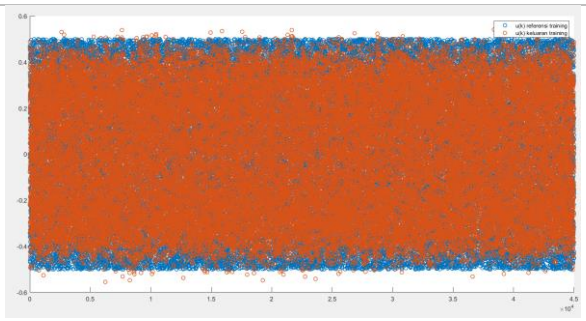
VarName1
Number
0.0437

Gambar 5.7 Bias V Inverse terbaik	*Gambar 5.8* Bias W Inverse terbaik

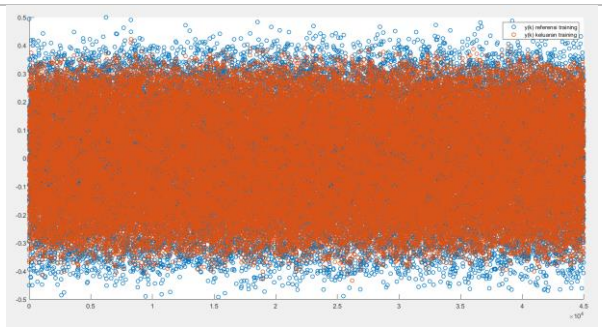
4.1.1.3 Pelatihan dan Pengujian model DIC

Pada proses *training* dan *testing* model DIC, digunakan beberapa parameter pada model ANN yang di antaranya adalah sebagai berikut.

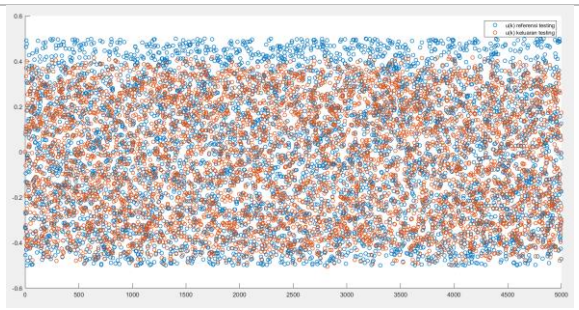
- Jumlah *hidden* neuron = 10
- Jumlah iterasi = 5000
- Laju pembelajaran = 0.5
- Laju pembelajaran momentum = 0.5
- Jumlah data *training* = 45000
- Jumlah datai *testing* = 5000



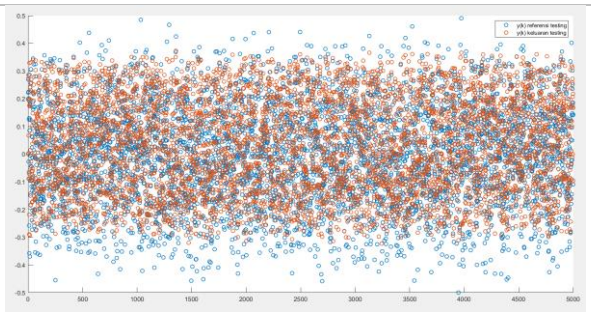
Gambar 5.9 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN selama proese *training* (u(k)/inverse)



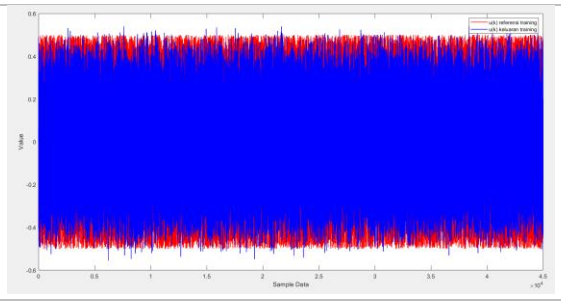
Gambar 6.0 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN selama proses *training* (y(k)/ID)

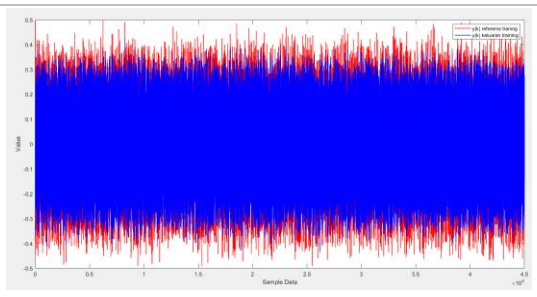


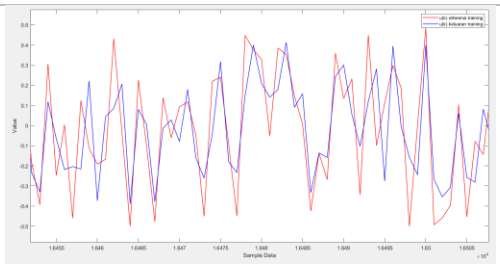
Gambar 6.1 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN selama proese *testing* (u(k)/inverse)



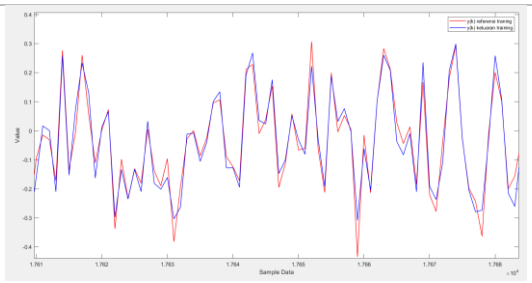
Gambar 6.2 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN selama proese *testing* (y(k)/ID)



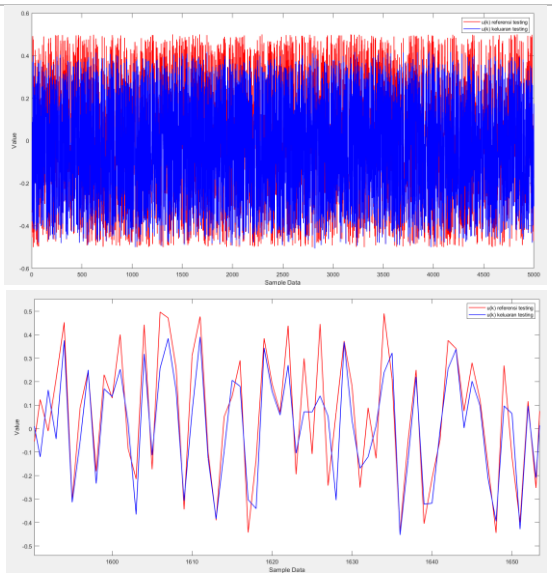




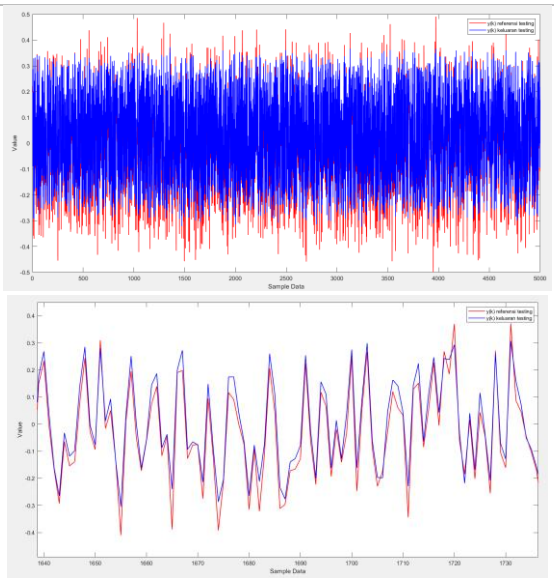
Gambar 6.3 Plot Perbandingan Data Keluaran *Plant* dan Model ANN selama proese *training* (u(k)/inverse)



Gambar 6.4 Plot Perbandingan Data Keluaran *Plant* dan Model ANN selama proses *training* (y(k)/ID)



Gambar 6.5 Plot Perbandingan Data Keluaran *Plant* dan Model ANN selama proese *testing* (u(k)/inverse)



Gambar 6.6 Plot Perbandingan Data Keluaran *Plant* dan Model ANN selama proses *testing* (y(k)/ID)

```
MSE Inverse train = 0.026677
MSE train = 0.0015926
MSE Inverse test = 0.020408
MSE test = 0.00214
```

Dapat dilihat melalui grafik yang diperoleh dari proses *training* model ANN inverse dan IDN dengan input nilai acak, dapat diketahui jika nilai *mean squared error* total yang diperoleh melalui proses *training*, validasi, dan *testing* sangat kecil nilainya. Hal ini mengindikasikan jika model ANN yang dibuat sudah memiliki performa cukup bagus untuk mulai diterapkan pada model ANN DIC.

Dari grafik MSE data *training* dan validasi yang diperoleh, dapat diketahui jika nilai MSE dari proses validasi tidak meningkat selama proses *training* berlangsung. Artinya, model ANN tidak mengalami *overfitting* saat melakukan estimasi terhadap nilai keluaran *plant*. Itulah sebabnya, bobot dan bias yang diperoleh setelah 5000 *epoch* akan digunakan sebagai bobot dan bias awal pada model ANN DIC.

Dalam proses algoritmanya, penurunan nilai MSE data *training* pada setiap *epoch*-nya untuk training model ANN IDN dan inverse ini disebabkan karena adanya proses *backpropagation* yang akan memperbaiki nilai bobot dan bias berdasarkan informasi error

antara estimasi nilai keluaran *plant* oleh model ANN dengan data keluaran *plant* sebenarnya. Perbandingan hasil estimasi nilai keluaran *plant* oleh model ANN dengan data keluaran *plant* sebenarnya pun dapat dilihat dengan menggunakan *scatter plot*. Titik berwarna biru menggambarkan keluaran dari *plant* sedangkan yang berwarna oren menggambarkan hasil estimasi keluaran *plant* oleh model ANN.

Untuk model ANN DIC, dapat diketahui jika dilakukan proses *training* dan *testing*. Pada dasarnya, terdapat dua cara yang bisa dilakukan untuk mencoba menerapkan model ANN DIC, yakni dengan melakukan *training* tahap 1 dari model ANN IDN dan inverse, dilanjutkan dengan *training* tahap 2 (ada di dasar teori untuk penjelasan mengenai *training* tahap 1 dan tahap 2), dilanjutkan dengan proses *testing* pada model ANN DIC dengan menggunakan bobot inisial yang diperoleh dari *training* tahap 2 dari model ANN IDN dan inverse. Adapun, pada laporan ini, penulis tidak menggunakan cara itu, melainkan *training* model ANN IDN dan inverse hanya dilakukan di tahap 1 saja, dilanjutkan dengan *training* model ANN DIC dengan menggunakan bobot inisial dari *training* tahap 1 dari model ANN IDN dan inverse, dan diakhiri dengan proses *testing* model ANN DIC. Hal ini penulis dan tim lakukan karena pada mekanisme tersebutlah penulis mendapatkan performa dari model ANN DIC yang sedikit lebih baik dibandingkan dengan mekanisme pertama.

Pada proses pelatihan dan pengujian model ANN DIC, diperoleh nilai *mean squared error* yang tidak terlalu kecil. Ini mengindikasikan jika model ANN DIC yang dibuat belum memiliki performa yang sangat memuaskan. Hal ini bisa disebabkan karena variasi data bernilai acak itu sangatlah banyak, dimana ketelitian sampel datanya bisa sampai 4 angka di belakang koma. Oleh karena itu, penulis mengajukan dua buah hal untuk dicoba. Yang pertama adalah dengan memperkecil rentang nilai normalisasi menjadi dari -0.5 dan 0.5, dan yang kedua yakni dengan memperbanyak data pelatihan. Dengan cara ini, penulis mendapatkan hasil yang sedikit lebih baik.

Dengan rentang nilai normalisasi yang menjadi lebih kecil, akan jadi lebih kecil selisih antara data aktual dengan data prediksi keluaran dari model ANN, walaupun kecenderungannya kurang lebih masih sama seperti sebelumnya. Hal tersebut menyebabkan nilai mse yang diperoleh akan jadi semakin lebih kecil. Ini telah dibuktikan oleh penulis melalui serangkaian percobaan yang telah dilakukan.

Selanjutnya, dalam usaha untuk meningkatkan performa dari model ANN DIC, penulis telah memperbanyak data *training* agar model ANN (dari 7000 sampel data menjadi 40000 sampel data). Upaya ini dilakukan untuk bisa membuat model ANN DIC bisa menjadi lebih familiar terhadap berbagai macam nilai input yang diberikan kepadanya. Hasilnya, performa dari model ANN DIC menjadi sedikit lebih baik dari sebelumnya.

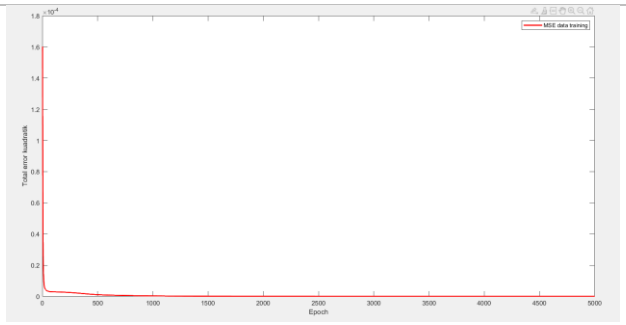
4.1.2 Step Input

4.1.2.1 Pelatihan dan Pengujian model ANN IDN

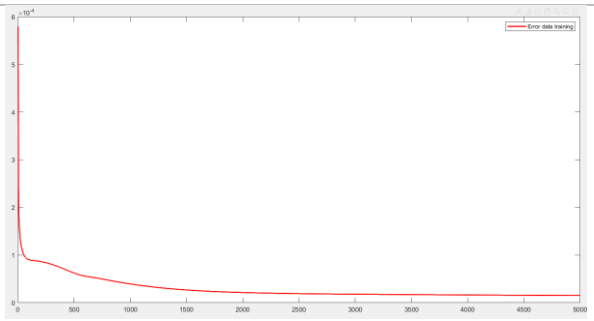
Pada proses *training* dan *testing* model ANN IDN , digunakan beberapa parameter pada model ANN yang di antaranya adalah sebagai berikut.

- Jumlah *hidden* neuron = 10
- Jumlah iterasi = 5000
- Laju pembelajaran = 0.5

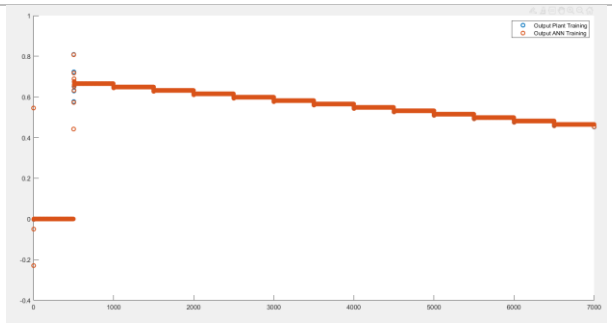
- Laju pembelajaran momentum = 0.5
- Jumlah data *training* = 7000
- Jumlah data *testing* = 5000



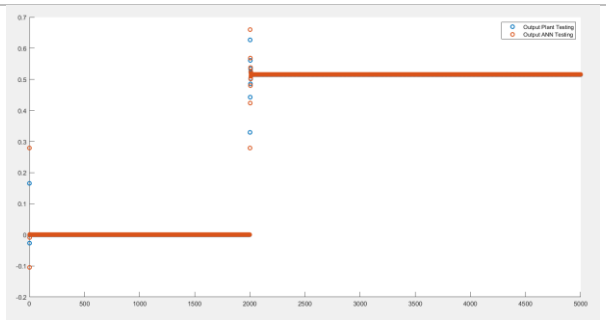
Gambar 6.7 Grafik Total Error Kuadratik Data *Training*



Gambar 6.8 Grafik Error Data *Training*



Gambar 6.9 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN (Data *Training*)



Gambar 7.0 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN (Data *Testing*)

```
Error final Data Training = 1.5129e-05
Error final Data Testing = 0.00071483
MSE final Data Training = 3.0204e-08
MSE total Data Training = 4.8355e-07
MSE total Data Testing = 5.0733e-06
```

VarName1	VarName2	VarName3	VarName4	VarName5	VarName6	VarName7	VarName8	VarName9	VarName10
Number	Number	Number	Number	Number	Number	Number	Number	Number	Number
-0.8082	0.1512	-0.6537	0.0690	-0.3272	-0.1436	0.1900	-0.3091	-0.2740	-0.3057
-0.1750	0.2404	-0.3045	-0.9522	-0.1861	1.8707	-1.0908	0.2641	-1.3425	-0.8841
0.6374	-1.4706	-0.3062	-0.1738	-0.2911	0.2163	-1.0855	-0.1242	0.1341	-0.2119
0.0599	0.3379	-0.6434	0.2832	0.2849	0.1531	2.5845	0.6155	-0.2858	-0.5687
0.5769	2.1115	-0.2611	0.1743	0.3665	-0.8452	-2.1377	0.9712	-0.2095	-0.2160

Gambar 7.1 Bobot V Terbaik

VarName1
Number
0.9871
-2.0779
0.7531
1.0484
0.3424
-2.3066
-1.3430
0.5700
1.0885
1.2853

Gambar 7.2 Bobot W terbaik

VarName1	VarName2	VarName3	VarName4	VarName5	VarName6	VarName7	VarName8	VarName9	VarName10
Number	Number	Number	Number	Number	Number	Number	Number	Number	Number
-0.4492	1.1340	0.2338	0.8019	0.7093	-1.4373	-0.3441	-1.0430	-0.8155	0.2722

Gambar 7.3 Bias V terbaik

VarName1
Number
0.1087

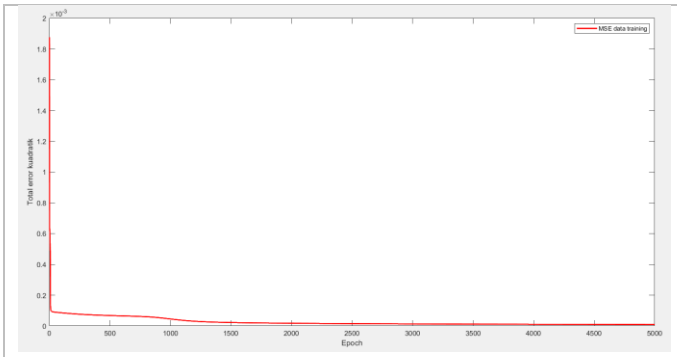
Gambar 7.4 Bias W terbaik

4.1.2.2 Pelatihan dan Pengujian model ANN Inverse

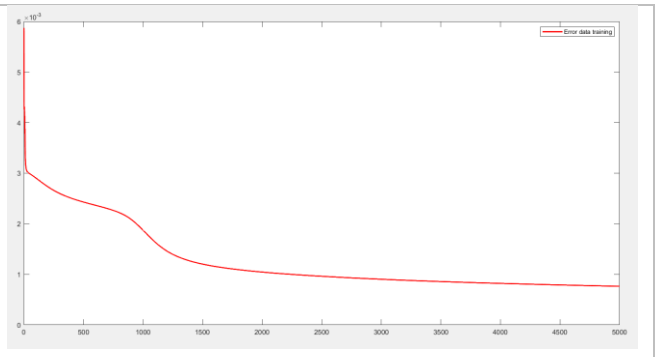
Pada proses *training* dan *testing* model ANN Inverse , digunakan beberapa parameter pada model ANN yang di antaranya adalah sebagai berikut.

- Jumlah *hidden* neuron = 10

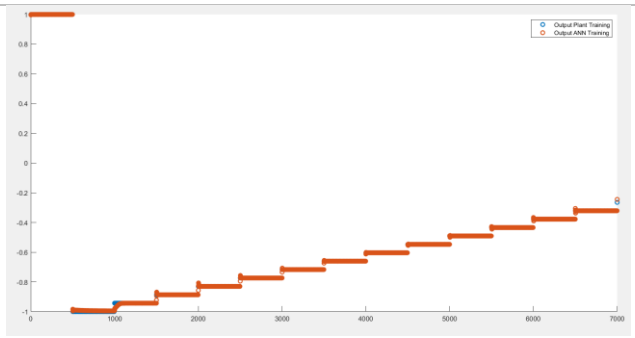
- Jumlah iterasi = 5000
- Laju pembelajaran = 0.5
- Laju pembelajaran momentum = 0.5
- Jumlah data *training* = 7000
- Jumlah data *testing* = 5000



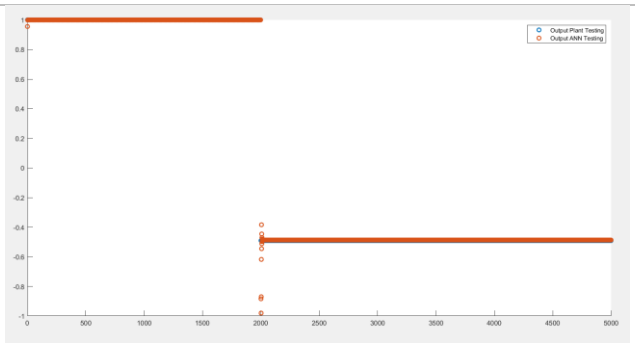
Gambar 7.5 Grafik Total Error Kuadratik Data *Training*



Gambar 7.6 Grafik Error Data Training



Gambar 7.7 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN (Data *Training*)



Gambar 7.8 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN (Data *Testing*)

```
Error final Data Training = 0.00076079
Error final Data Testing = 0.0021793
MSE final Data Training = 9.1105e-06
MSE total Data Training = 2.922e-05
MSE total Data Testing = 0.00012022
```

VarName1	VarName2	VarName3	VarName4	VarName5	VarName6	VarName7	VarName8	VarName9	VarName10
Number	Number	Number	Number	Number	Number	Number	Number	Number	Number
2.7041	-0.2547	-0.5330	1.0537	1.4358	-0.4061	0.9246	0.6980	0.9527	-0.4423
4.4489	0.1872	-0.1456	-0.6794	0.2164	1.2663	-0.6346	-0.7122	0.1553	0.1443
0.8374	-0.4098	-0.1614	0.1322	1.0823	0.0275	0.2963	0.1227	2.0880	-0.0473
-5.1129	1.0486	-1.2688	2.3776	5.0552	-0.9404	3.7695	2.4976	2.9782	-2.0371
-2.0523	-0.3655	-0.9079	0.4110	2.3503	-0.3230	-0.1852	-0.2603	0.0745	0.0858
1.3895	-0.0410	-0.2281	-0.3596	2.2201	0.2951	-0.6210	-0.3868	-0.8054	-0.0319

Gambar 7.9 Bobot V Inverse Terbaik

VarName1
Number
7.2414
-0.2855
0.5301
-1.0426
-7.4434
0.3866
-1.2951
-0.8114
-1.8739
0.9700

Gambar 8.0 Bobot W Inverse terbaik

VarName1	VarName2	VarName3	VarName4	VarName5	VarName6	VarName7	VarName8	VarName9	VarName10
Number	Number	Number	Number	Number	Number	Number	Number	Number	Number
-0.4492	1.1340	0.2338	0.8019	0.7093	-1.4373	-0.3441	-1.0430	-0.8155	0.2722

Gambar 8.1 Bias V Inverse terbaik

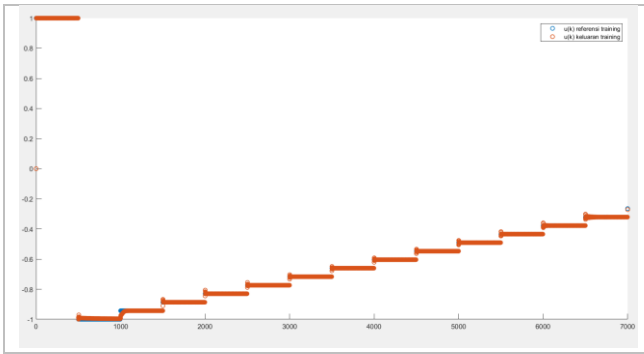
VarName1
Number
-2.1182

Gambar 8.2 Bias W Inverse terbaik

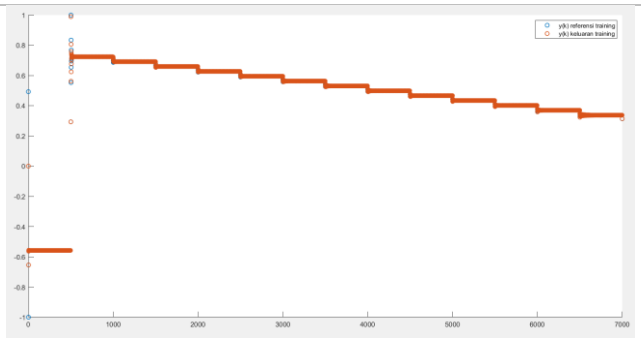
4.1.2.3 Pelatihan dan Pengujian model DIC

Pada proses *training* dan *testing* model DIC , digunakan beberapa parameter pada model ANN yang di antaranya adalah sebagai berikut.

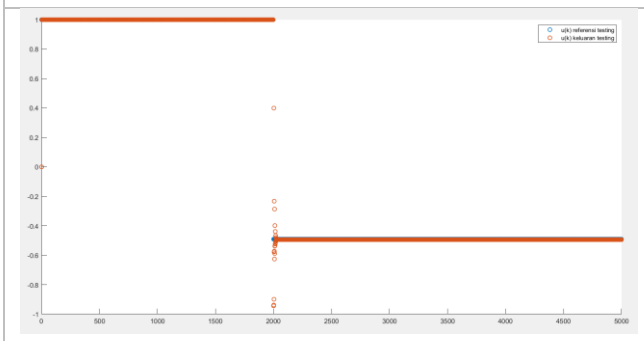
- Jumlah *hidden* neuron = 10
- Jumlah iterasi = 5000
- Laju pembelajaran = 0.5
- Laju pembelajaran momentum = 0.5
- Jumlah data *training* = 7000
- Jumlah data *testing* = 5000



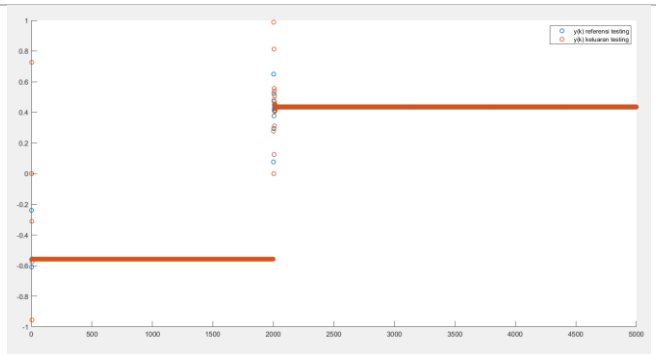
Gambar 8.3 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN selama proese *training* (u(k)/inverse)



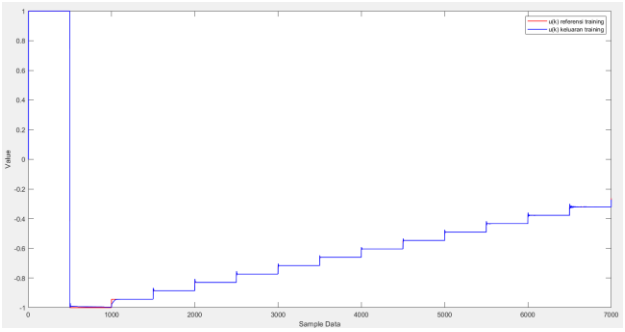
Gambar 8.4 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN selama proses *training* (y(k)/ID)



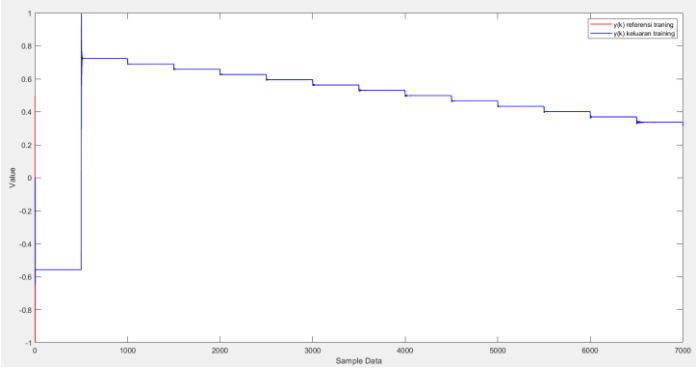
Gambar 8.5 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN selama proese *testing* (u(k)/inverse)



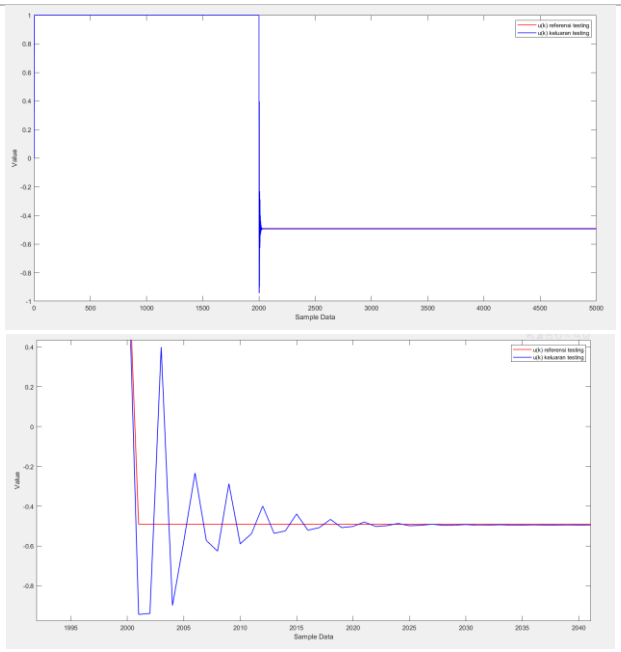
Gambar 8.6 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN selama proese *testing* (y(k)/ID)



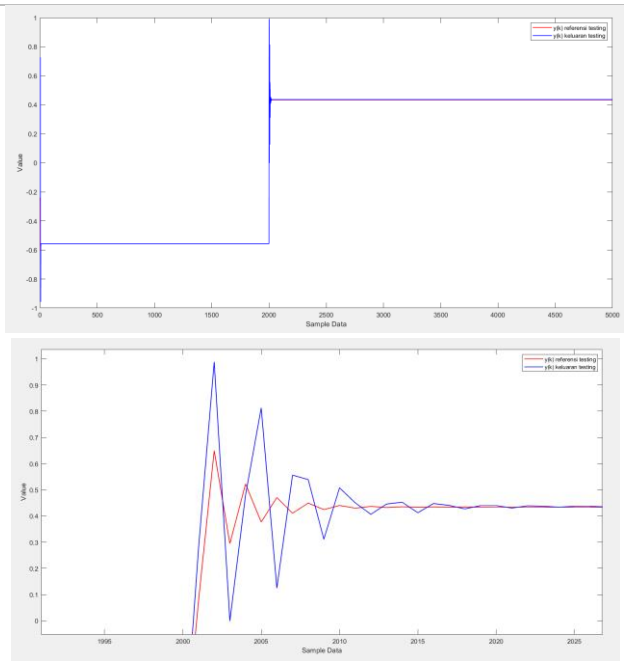
Gambar 8.7 Plot Perbandingan Data Keluaran *Plant* dan Model ANN selama proese *training* (u(k)/inverse)



Gambar 8.8 Plot Perbandingan Data Keluaran *Plant* dan Model ANN selama proses *training* (y(k)/ID)



Gambar 8.9 Plot Perbandingan Data Keluaran *Plant* dan Model ANN selama proese *testing* (u(k)/inverse)



Gambar 9.0 Plot Perbandingan Data Keluaran *Plant* dan Model ANN selama proses *testing* (y(k)/ID)

```
MSE Inverse train = 1.0524e-05
MSE train = 1.4862e-06
MSE Inverse test = 0.00031308
MSE test = 0.00049768
```

Dapat dilihat melalui grafik yang diperoleh dari proses *training* model ANN inverse dan IDN dengan input sinyal step, dapat diketahui jika nilai *mean squared error* total yang diperoleh melalui proses *training*, validasi, dan *testing* sangat kecil nilainya. Hal ini mengindikasikan jika model ANN yang dibuat sudah memiliki performa cukup bagus untuk mulai diterapkan pada model ANN DIC.

Dalam proses algoritmanya, penurunan nilai MSE data *training* pada setiap *epoch*-nya untuk training model ANN IDN dan inverse ini disebabkan karena adanya proses *backpropagation* yang akan memperbaiki nilai bobot dan bias berdasarkan informasi error antara estimasi nilai keluaran *plant* oleh model ANN dengan data keluaran *plant* sebenarnya. Perbandingan hasil estimasi nilai keluaran *plant* oleh model ANN dengan data keluaran *plant* sebenarnya pun dapat dilihat dengan menggunakan *scatter plot*. Titik berwarna biru menggambarkan keluaran dari *plant* sedangkan yang berwarna oren menggambarkan hasil estimasi keluaran *plant* oleh model ANN.

Untuk model ANN DIC, dapat diketahui jika dilakukan proses *training* dan *testing* dengan nilai bobot awal yang diperoleh dari *training* model ANN IDN dan inverse pada tahap sebelumnya. Pada dasarnya, terdapat dua cara yang bisa dilakukan untuk mencoba menerapkan model ANN DIC, yakni dengan melakukan *training* tahap 1 dari model ANN IDN dan inverse, dilanjutkan dengan *training* tahap 2 (ada di dasar teori untuk penjelasan mengenai *training* tahap 1 dan tahap 2), dilanjutkan dengan proses *testing* pada model ANN DIC dengan menggunakan bobot inisial yang diperoleh dari *training* tahap 2 dari model ANN IDN dan inverse. Adapun, pada laporan ini, penulis tidak menggunakan cara itu, melainkan *training*

model ANN IDN dan inverse hanya dilakukan di tahap 1 saja, dilanjutkan dengan *training* model ANN DIC dengan menggunakan bobot inisial dari *training* tahap 1 dari model ANN IDN dan inverse, dan diakhiri dengan proses *testing* model ANN DIC. Hal ini penulis dan tim lakukan karena pada mekanisme tersebutlah penulis mendapatkan performa dari model ANN DIC yang sedikit lebih baik dibandingkan dengan mekanisme pertama.

Pada proses pelatihan dan pengujian model ANN DIC, diperoleh nilai *mean squared error* yang sudah sangat kecil. Ini mengindikasikan jika model ANN DIC yang dibuat sudah memiliki performa yang sangat memuaskan. Hal ini bisa disebabkan karena pada proses pelatihan, penulis memberikan berbagai macam variasi nilai *step* yang mungkin saja bisa diterima sebagai masukan dari model ANN DIC. Jika hal ini tidak dilakukan, maka, model ANN hanya akan belajar untuk "membentuk" sinyal step saja (pola yang dipelajari hanya bentuk sinyal step), tetapi untuk *steady state* errornya akan masih sangat tinggi. Oleh karena itu, cara tersebut haruslah dilakukn untuk mengatasi nilai *steady state* error tersebut.

Pada proses pelatihan model ANN DIC, penulis mencoba untuk melatih model ANN dengan memberikan sampel data sinyal step dengan berbagai macam variasi *final value*. Hal ini perlu dilakukan agar model ANN DIC bisa familiar dengan berbagai macam variasi sinyal step. Dalam hal ini, sampel data akan diturunkan/dinaikkan nilainya sebanyak 0.1 setiap 500 sampel data. Dengan cara ini, dapat diperoleh berbagai macam variasi sinyal step (karena keterbatasan waktu, variasi sinyal step dilakukan secara demikian, tetapi untuk hasil yang lebih bagus, bisa diberikan berbagai macam sinyal step dengan variasi *step time* dan *final value*). Dengan semakin banyaknya variasi sinyal step yang diberikan kepada model ANN yang dilatih, maka akan semakin kecil nilai mse yang dapat diperoleh.

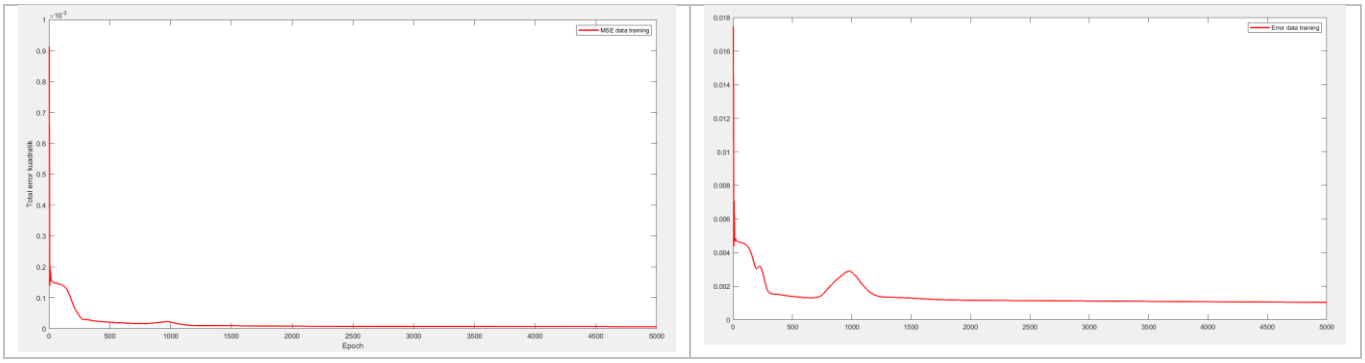
Dengan cara yang dilakukan tersebut, diperoleh sinyal keluaran yang telah mengikuti sinyal referensi.

4.1.3 Sinusoidal Input

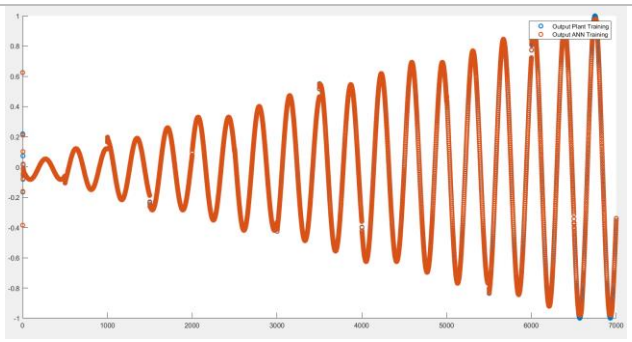
4.1.3.1 Pelatihan dan Pengujian model ANN IDN

Pada proses *training* dan *testing* model ANN IDN , digunakan beberapa parameter pada model ANN yang di antaranya adalah sebagai berikut.

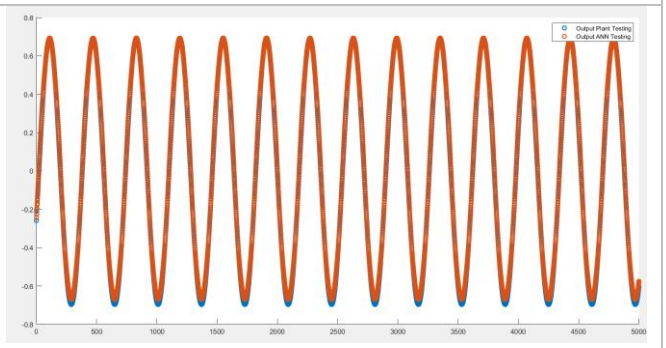
- Jumlah *hidden* neuron = 10
- Jumlah iterasi = 5000
- Laju pembelajaran = 0.5
- Laju pembelajaran momentum = 0.5
- Jumlah data *training* = 7000
- Jumlah data *testing* = 5000



Gambar 9.1 Grafik Total Error Kuadratik Data *Training*



Gambar 9.2 Grafik Error Data Training



Gambar 9.3 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN (Data *Training*)

Gambar 9.4 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN (Data *Testing*)

```
Error final Data Training = 0.0010362
Error final Data Testing = 0.014212
MSE final Data Training = 6.5493e-06
MSE total Data Training = 1.5848e-05
MSE total Data Testing = 0.00033641
```

VarName1	VarName2	VarName3	VarName4	VarName5	VarName6	VarName7	VarName8	VarName9	VarName10
Number	Number	Number	Number	Number	Number	Number	Number	Number	Number
2.0707	0.8456	0.3911	-1.5898	0.8868	-1.7514	-1.0983	-3.4451	-2.7617	-2.7362
0.8388	-0.0343	0.0740	-0.2365	-0.4726	-3.3630	-1.0760	1.3991	-1.3874	-0.7117
-1.8735	-0.5778	0.6023	0.6496	-0.1886	-2.7670	0.3220	3.8398	-0.2817	1.8012
0.6789	0.1492	1.1585	0.7359	0.1452	-1.7543	-0.7302	0.7130	2.3194	-1.6548
0.0161	-0.3445	-0.1513	-1.4865	-0.3425	0.3544	-0.9487	-1.7079	2.3030	-0.6649

Gambar 9.5 Bobot V Terbaik

VarName1
Number
-0.8626
-0.3255
-0.1338
0.7075
-0.2610
6.0092
0.4960
-1.1846
6.8100
1.2098

Gambar 9.6 Bobot W terbaik

VarName1	VarName2	VarName3	VarName4	VarName5	VarName6	VarName7	VarName8	VarName9	VarName10
Number	Number	Number	Number	Number	Number	Number	Number	Number	Number
-0.4160	-0.0153	-0.1687	0.5307	-0.0276	-7.6409	0.5072	0.0099	8.8100	-0.6007

Gambar 9.7 Bias V terbaik

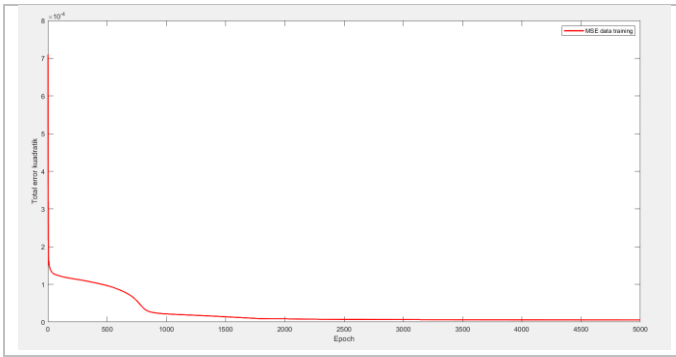
VarName1
Number
-0.6989

Gambar 9.8 Bias W terbaik

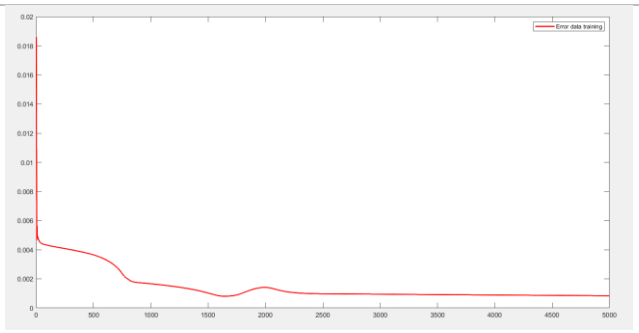
4.1.3.2 Pelatihan dan Pengujian model ANN Inverse

Pada proses *training* dan *testing* model ANN Inverse , digunakan beberapa parameter pada model ANN yang di antaranya adalah sebagai berikut.

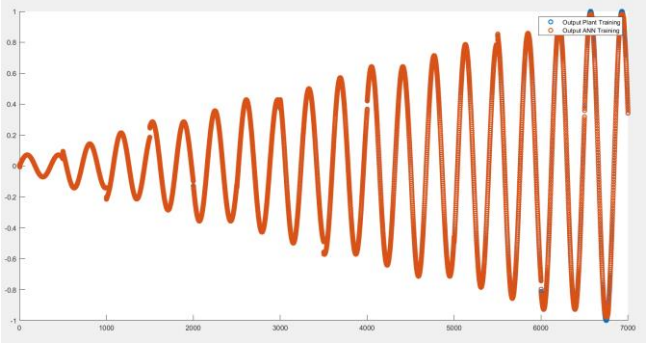
- Jumlah *hidden* neuron = 10
- Jumlah iterasi = 5000
- Laju pembelajaran = 0.5
- Laju pembelajaran momentum = 0.5
- Jumlah data *training* = 7000
- Jumlah data *testing* = 5000



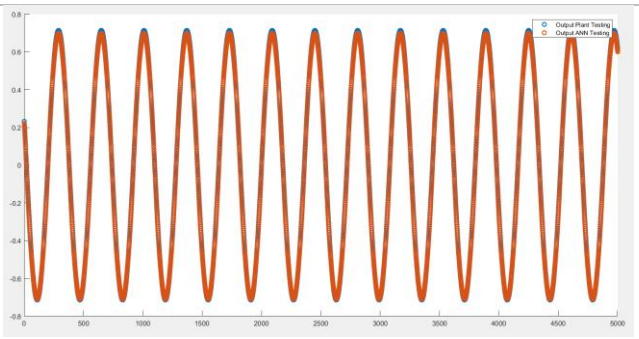
Gambar 9.9 Grafik Total Error Kuadratik Data *Training*



Gambar 10.0 Grafik Error Data Training



Gambar 10.1 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN (Data *Training*)



Gambar 10.2 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN (Data *Testing*)

```
Error final Data Training = 0.0008409
Error final Data Testing = 0.010059
MSE final Data Training = 5.4352e-06
MSE total Data Training = 2.381e-05
MSE total Data Testing = 0.00014564
```

VarName1	VarName2	VarName3	VarName4	VarName5	VarName6	VarName7	VarName8	VarName9	VarName10
Number	Number	Number	Number	Number	Number	Number	Number	Number	Number
0.4606	0.2801	0.3285	-0.8363	2.3752	0.5770	0.9231	-0.3204	-1.3806	-0.9467
1.2628	-0.5377	-0.5617	0.4608	-1.0907	0.7451	0.5644	0.0085	0.2158	-0.6908
0.5146	0.2675	-0.2232	-0.9560	-3.3524	-0.4354	0.6320	-0.4468	0.4573	-1.5095
-3.6852	-3.2193	-0.4425	-2.0743	-1.2345	0.4503	0.8425	-1.4739	-0.9978	-3.1365
-0.9387	-4.0737	3.2282e-04	-1.6927	0.4619	0.7180	1.4863	-0.2298	-0.3382	-1.0466
-0.2035	-1.7173	-0.1983	1.0414	1.5587	0.1820	0.4338	0.7980	0.2747	0.6166

Gambar 10.3 Bobot V Inverse Terbaik

VarName1
Number
5.4943
6.0662
0.1665
1.2108
-1.1056
-0.3028
-0.5522
0.5519
0.3668
1.3662

Gambar 10.4 Bobot W Inverse terbaik

VarName1	VarName2	VarName3	VarName4	VarName5	VarName6	VarName7	VarName8	VarName9	VarName10
Number	Number	Number	Number	Number	Number	Number	Number	Number	Number
8.2493	-8.4529	0.0493	-1.2999	-0.1321	-0.5725	-0.3880	-0.0633	0.1884	0.4521

Gambar 10.5 Bias V Inverse terbaik

VarName1
Number
0.9319

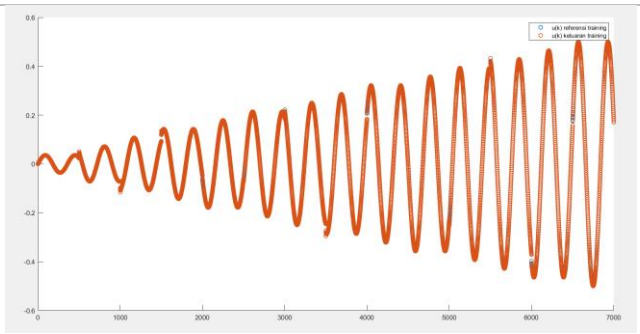
Gambar 10.6 Bias W Inverse terbaik

4.1.3.3 Pelatihan dan Pengujian model DIC

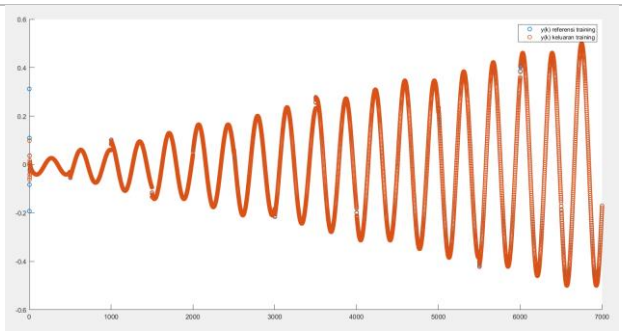
Pada proses *training* dan *testing* model DIC , digunakan beberapa parameter pada model ANN yang di antaranya adalah sebagai berikut.

- Jumlah *hidden* neuron = 10

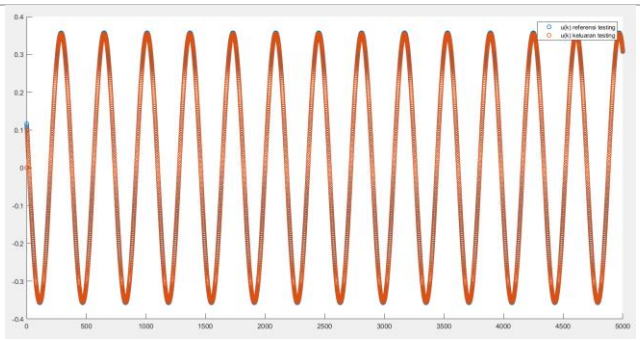
- Jumlah iterasi = 5000
- Laju pembelajaran = 0.5
- Laju pembelajaran momentum = 0.5
- Jumlah data *training* = 7000
- Jumlah data *testing* = 5000



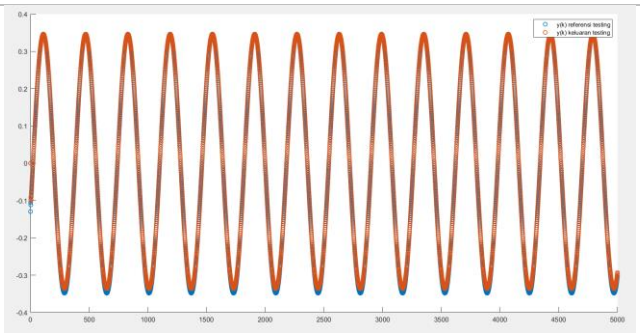
Gambar 10.7 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN selama proese *training* (u(k)/inverse)



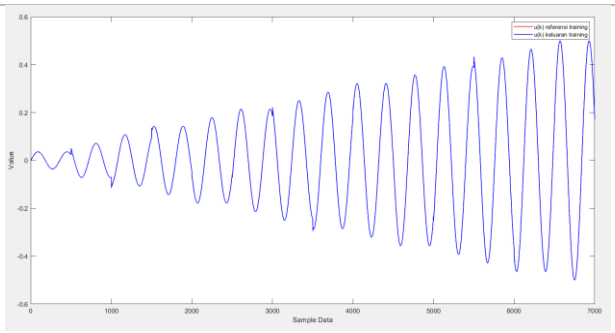
Gambar 10.8 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN selama proses *training* (y(k)/ID)



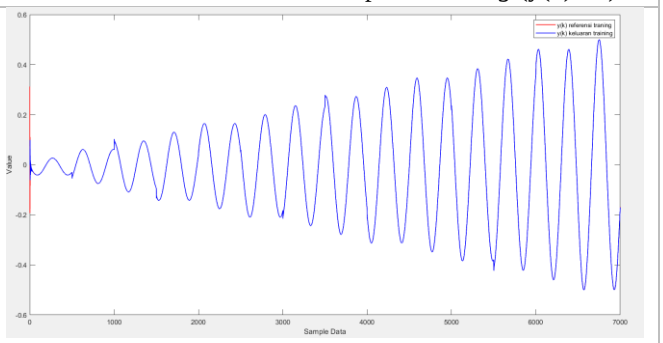
Gambar 10.9 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN selama proese *testing* (u(k)/inverse)



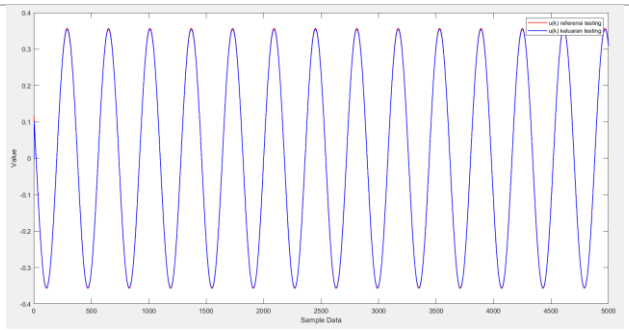
Gambar 10.10 Scatter Plot Perbandingan Data Keluaran *Plant* dan Model ANN selama proese *testing* (y(k)/ID)



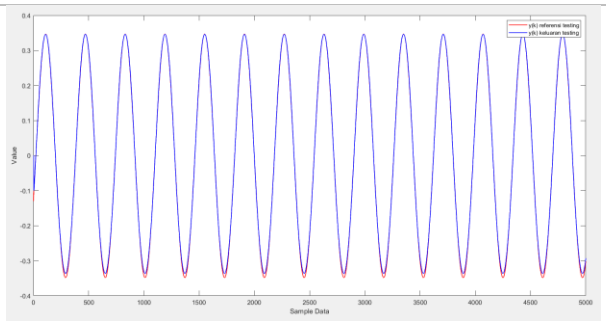
Gambar 10.11 Plot Perbandingan Data Keluaran *Plant* dan Model ANN selama proese *training* (u(k)/inverse)



Gambar 10.12 Plot Perbandingan Data Keluaran *Plant* dan Model ANN selama proses *training* (y(k)/ID)



Gambar 10.13 Plot Perbandingan Data Keluaran *Plant* dan Model ANN selama proese *testing* (u(k)/inverse)



Gambar 10.14 Plot Perbandingan Data Keluaran *Plant* dan Model ANN selama proses *testing* (y(k)/ID)

```
MSE Inverse train = 3.4928e-07
MSE train = 7.6646e-07
MSE Inverse test = 1.8004e-06
MSE test = 2.6444e-05
```

Dapat dilihat melalui grafik yang diperoleh dari proses *training* model ANN inverse dan IDN dengan input sinyal sinusoidal, dapat diketahui jika nilai *mean squared error* total yang diperoleh melalui proses *training*, validasi, dan *testing* sangat kecil nilainya. Hal ini mengindikasikan jika model ANN yang dibuat sudah memiliki performa cukup bagus untuk mulai diterapkan pada model ANN DIC.

Dari grafik MSE data *training* dan validasi yang diperoleh, dapat diketahui jika nilai MSE dari proses validasi tidak meningkat selama proses *training* berlangsung. Artinya, model ANN tidak mengalami *overfitting* saat melakukan estimasi terhadap nilai keluaran *plant*. Itulah sebabnya, bobot dan bias yang diperoleh setelah 5000 *epoch* akan digunakan sebagai bobot dan bias awal pada model ANN DIC.

Dalam proses algoritmanya, penurunan nilai MSE data *training* pada setiap *epoch*-nya untuk training model ANN IDN dan inverse ini disebabkan karena adanya proses *backpropagation* yang akan memperbaiki nilai bobot dan bias berdasarkan informasi error antara estimasi nilai keluaran *plant* oleh model ANN dengan data keluaran *plant* sebenarnya. Perbandingan hasil estimasi nilai keluaran *plant* oleh model ANN dengan data keluaran *plant* sebenarnya pun dapat dilihat dengan menggunakan *scatter plot*. Titik berwarna biru menggambarkan keluaran dari *plant* sedangkan yang berwarna oren menggambarkan hasil estimasi keluaran *plant* oleh model ANN.

Untuk model ANN DIC, dapat diketahui jika dilakukan proses *training* dan *testing* dengan nilai bobot awal yang diperoleh dari *training* model ANN IDN dan inverse pada tahap sebelumnya. Pada dasarnya, terdapat dua cara yang bisa dilakukan untuk mencoba menerapkan model ANN DIC, yakni dengan melakukan *training* tahap 1 dari model ANN IDN dan inverse, dilanjutkan dengan *training* tahap 2 (ada di dasar teori untuk penjelasan mengenai *training* tahap 1 dan tahap 2), dilanjutkan dengan proses *testing* pada model ANN DIC dengan menggunakan bobot inisial yang diperoleh dari *training* tahap 2 dari model ANN IDN dan inverse. Adapun, pada laporan ini, penulis tidak menggunakan cara itu, melainkan *training* model ANN IDN dan inverse hanya dilakukan di tahap 1 saja, dilanjutkan dengan *training* model ANN DIC dengan menggunakan bobot inisial dari *training* tahap 1 dari model ANN IDN dan inverse, dan diakhiri dengan proses *testing* model ANN DIC. Hal ini penulis dan tim lakukan karena pada mekanisme tersebutlah penulis mendapatkan performa dari model ANN DIC yang sedikit lebih baik dibandingkan dengan mekanisme pertama.

Pada proses pelatihan dan pengujian model ANN DIC, diperoleh nilai *mean squared error* yang sudah sangat kecil. Ini mengindikasikan jika model ANN DIC yang dibuat sudah memiliki performa yang sangat memuaskan. Hal ini bisa disebabkan karena pada proses pelatihan, penulis memberikan berbagai macam variasi nilai amplitudo yang mungkin saja bisa diterima sebagai masukan dari model ANN DIC. Jika hal ini tidak dilakukan, maka, model ANN hanya akan belajar untuk "membentuk" sinyal sinusoidal saja (pola yang dipelajari hanya bentuk sinyal sinusoidal), tetapi untuk nilai error amplitudo dari sinyal sinusoidal akan masih

sangat tinggi. Oleh karena itu, cara tersebut haruslah dilakukn untuk mengatasi nilai error tersebut.

Pada proses pelatihan model ANN DIC dengan input sinyal sinusoidal, model dapat dilatih dengan memvariasikan amplitudo dan frekuensi dari sinyal sinusoidal. Dengan cara ini, model ANN akan menghasilkan performa yang lebih baik untuk berbagai macam kemungkinan input yang ada. Dalam hal ini, karena keterbatasan waktu dan perangkat, penulis hanya melatih model ANN dengan memvariasikan amplitudo dari sinyal sinusoidal. Dalam hal ini, sampel data akan amplitudo sinyal sinusoidal akan dinaikkan/diturunkan nilainya sebanyak 0.1 setiap 500 sampel data. Dengan cara ini, dapat diperoleh berbagai macam variasi sinyal sinusoidal. Dengan semakin banyaknya variasi sinyal sinusoidal yang diberikan kepada model ANN yang dilatih, maka akan semakin kecil nilai mse yang dapat diperoleh.

Dengan cara yang dilakukan tersebut, diperoleh sinyal keluaran yang telah mengikuti sinyal referensi.

4.2 Analisis Keseluruhan

Dari hasil percobaan yang telah dilakukan, dapat diketahui jika model *Artificial Neural Network* (ANN) yang telah dibangun sudah memberikan performa yang cukup baik pada masukan nilai acak, dan performa yang sangat baik pada masukan sinyal step dan sinusoidal. Performa yang optimal dari percobaan model ANN DIC akan sangat dipengaruhi oleh jenis data masukan yang ada. Datamasukan dengan “pola” yang lebih sederhana, akan memberiakn keakuratan hasil yang lebih maksimal.

Dalam suatu model *Artificial Neural Network* (ANN), secara umum terdapat beberapa parameter penting yang harus ada dalam dataset untuk keefektifan dari model yang dibangun.

1. Kualitas yang bagus dari dataset. Ibarat suatu suara, tingkat kebisingan yang rendah tentu merupakan salah satu fitur kritis dalam masalah klasifikasi suara.
2. Kuantitas dari dataset. Pada dasarnya, *Artificial Neural Network* (ANN) membutuhkan data yang sangat besar untuk dilatih (*training*). Oleh sebab itu, tingkat akurasi dapat dipengaruhi oleh kuantitas dari dataset.
3. Variabilitas dari data. Variasi dari semua kemungkinan kasus yang ada pada dataset merupakan hal penting yang dapat meningkatkan kualitas dari dataset.

Dari sisi arsitektur model *Artificial Neural Network* (ANN) yang dibangun, terdapat beberapa hal yang dapat dipertimbangkan, di antaranya adalah sebagai berikut.

1. Arsitektur. Apakah model yang digunakan sesuai dengan masalah yang ada pada dataset? Terkadang penggunaan model *Artificial Neural Network* (ANN) untuk melakukan klasifikasi suatu masalah tertentu bukanlah merupakan ide yang bagus. Ini merupakan hal penting yang perlu dipertimbangkan sebelum memulai *machine learning* dengan menggunakan ANN.
2. *Overtraining*. Pelatihan model secara berlebihan. Oleh karena itu, proses penambahan beberapa lapisan dropout terkadang dapat membantu mengatur model menjadi lebih baik.
3. *Tuning hyperparameter*. Proses *tuning hyperparameter* yang digunakan pada model *Artificial Neural Network* (ANN) perlu dilakukan untuk mendapatkan kemungkinan parameter terbaik untuk jumlah *layer*, *learning rate* (alpha), *optimizer*, dan lain-lain.

Terkait dengan dataset yang digunakan, dapat diketahui jika data masukan berupa nilai acak dapat dibilang memiliki pola yang jauh lebih sulit untuk dipelajari jika dibandingkan

dengan data masukan berupa sinyal step atau sinusoidal. Tidak heran jika kesederhanaan pola pada sampel data yang ada di dataset merupakan salah satu faktor penting penyebab performa model yang baik.

Seperti yang sudah disinggung sebelumnya, penulis melakukan mekanisme *training* tahap 1 dari model ANN inverse dan IDN, dilanjutkan dengan *training* model ANN DIC, dan terakhir *testing* model ANN DIC. Hasilnya, mse yang diperoleh sudah cukup baik pada input nilai acak, dan sangat baik pada input sinyal step dan sinusoidal. Dalam penerapan model ANN DIC sebagai pengendali, nilai error yang dihasilkan selama pengujian model merupakan hal yang harus menjadi perhatian utama.

Pada dasarnya, pengendali diciptakan untuk mereduksi nilai error di antara data referensi dan data aktual menjadi nol sehingga nilai *steady state error* menjadi nol pada saat keadaan tunak. Proses mereduksi nilai error itu harus selalu dilakukan selama sistem dijalankan (masa transien utamanya hingga keadaan tunak). Itulah sebabnya, diperlukan suatu pengendali yang dapat mereduksi nilai error secara kontinu selama simulasi dijalankan.

Nilai error yang kecil memang sudah cukup bagus, tetapi dalam sistem besar yang kompleks, dikhawatirkan error yang kecil bisa saja membuat performa dari sistem menjadi tidak optimal, baik itu dari segi efisiensi, hingga ekonomi. Kemungkinan terburuknya, nilai error bisa memberikan dampak yang fatal bagi pengendali, serta sistem yang ada di dalamnya, seperti kebocoran, kebakaran, hingga yang lainnya.

Dari hal itulah, penulis merasa jika pengkoreksian bobot pada model ANN yang diterapkan sebagai pengendali harus selalu dilakukan selama sistem berjalan. Pengkoreksian nilai bobot ini dapat dilakukan dengan menghadirkan algoritma *backpropagation* di setiap saat selama penerapan model ANN sebagai pengendali. Dengan adanya pengkoreksian bobot selama model ANN dijalankan, akan jadi sangat mungkin bagi model ANN untuk mereduksi nilai error menjadi nol sehingga nilai *steady state error* pada kondisi tunak akan bernilai nol.

Yang menjadi masalah sekarang adalah, berapa lama waktu yang dibutuhkan oleh model ANN agar bisa mereduksi nilai error menjadi nol. Pada hakikatnya, masa transien tidak bisa berlangsung terlalu lama (dan terlalu cepat juga). Karakteristik respons transien merupakan suatu hal yang tetap selalu harus diperhatikan. Dalam penerapan model ANN sebagai pengendali, penulis berhipotesis jika, perbaikan nilai karakteristik respons transien dari sistem yang menggunakan model ANN (DIC) sebagai pengendalinya, bisa diwujudkan dengan men-*training* model ANN dengan data sebanyak mungkin. Dengan cara ini, model ANN akan menjadi familiar dengan berbagai macam kemungkinan nilai input *error* yang ada.

Tidak hanya itu, variasi nilai *error* yang masuk ke dalam model ANN (pengendali) juga tidak boleh terlalu lebar. Variasi nilai *error* yang terlalu tinggi dikhawatirkan dapat menyebabkan model ANN akan menjadi lebih sulit untuk belajar terhadap pola yang ada pada nilai input. Oleh karena itu, mungkin dapat diterapkan semacam *filter* ataupun teknologi sejenisnya untuk memperkecil rentang variasi nilai *error* yang masuk ke dalam pengendali (semacam memperkecil rentang nilai normalisasi) sehingga akan menjadi lebih mudah bagi model ANN untuk belajar mengenai “pola” yang ada pada sampel data.

BAB V
KESIMPULAN

Penerapa model ANN sebagai pengendali dari suatu sistem sangat memungkinkan dan sangat bagus untuk diterapkan. Dengan model ANN, akan dapat dihasilkan pengendali dengan performa yang tinggi. Dari beberapa percobaan yang telah dilakukan dalam penerapan model ANN sebagai pengendali, penulis dapat memberikan kesimpulan yang dapat direpresentasikan dalam beberapa poin poin penting, yaitu:

- Model ANN dapat digunakan untuk memodelkan suatu *plant* pada suatu sistem kendali. *Plant* ini merupakan gambaran dari pengendali dari suatu sistem kendali.
- Model ANN memiliki performa yang baik jika diterapkan sebagai pengendali dalam suatu sistem kendali.
- Normalisasi merupakan tahapan yang penting untuk dilakukan sebelum memulai proses pembelajaran karena dengan normalisasi, akan lebih mudah bagi model untuk mempelajari "pola" yang ada pada sekumpulan data.
- Terdapat dua mekanisme yang dapat diterapkan dalam mengaplikasikan model ANN DIC sebagai pengendali. Cara pertama yaitu dengan melakukan *training* tahap 1 dan tahap 2 dari model ANN IDN dan inverse, dilanjutkan dengan melakukan *testing* terhadap model ANN DIC. Adapun cara kedua yaitu dengan melakukan *training* tahap 1 dari model ANN IDN dan inverse, dilanjutkan dengan melakukan *training* dan *testing* terhadap model ANN DIC. Berdasarkan percobaan yang dilakukan penulis, mekanisme kedua memberikan hasil yang sedikit lebih baik jika dibandingkan dengan mekanisme pertama.
- Jenis data masukan akan sangat mempengaruhi performa dari model ANN DIC sebagai pengendali pada nantinya.
- Bobot merupakan suatu hal yang sangat penting dalam model ANN. Pada penerapannya sebagai pengendali, diperlukan performa pengendali yang sangat bagus, dimana nilai error antara data referensi dan aktual bernilai nol. Oleh karena itu, kehadiran algoritma *backpropagation* harus selalu ada pada penerapan model ANN DIC sebagai pengendali agar nilai *error* bisa selalu diperbaiki sehingga menjadi nol pada saat keadaan tunak.
- Untuk mendapatkan bobot inisial yang bagus, diperlukan pelatihan terhadap model ANN dengan jumlah sampel data sebanyak mungkin. Semakin banyak jumlah sampel data, maka besar kemungkinan nilai mse yang diperoleh akan semakin kecil, serta waktu yang dibutuhkan pengendali untuk mereduksi nilai error menjadi nol akan jadi semakin kecil.

REFERENSI

[1] B. Kusumaputro, Artificial Neural Network, 2022