UNIVERSITAS INDONESIA

SISTEM BERBASIS PENGETAHUAN
LAPORAN PROYEK JARINGAN SYARAF TIRUAN UNTUK IDENTIFIKASI SISTEM

Hansel Matthew (1806194914)

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO

DEPOK
MEI 2021

DAFTAR ISI

BAB I
PENDAHULUAN

2.1 Latar Belakang

Sistem kendali adalah cabang ilmu yang fokus mempelajari pengendalian respon dari sebuah sistem. Sistem tersebut dapat datang dari banyak ruang lingkup seperti pada sistem proses kimia, produksi tenaga listrik, bidang manufaktur, produksi minyak dan gas, dan masih banyak lagi. Sistem kendali pertama yaitu sistem kendali klasik PID dikembangkan pada tahun 1940 untuk pengendali dalam proses industri. 90% dari lup pengendalian menggunakan pengendali PID (Proporsional – Integral – Diferensial). Sistem kendali klasik ini Sebagian besar masih bekerja secara manual dengan parameter ditentukan secara trial and error. Penalaan ulang dari parameter ini menghabiskan waktu yang lama serta memakan biaya yang tinggi. Sering sekali industri menggunakan pengendali dengan nilai parameter *default* sehingga memberikan kinejar yang tidak sama untuk perubahan dari titik operasi.

Seiring berkembangnya zaman, sistem kendali menjadi semakin dibutuhkan, hal itu dikarenakan kemampuan manusia tidak dapat lagi mengalahkan kemampuan mesin dalam hal mengendalikan dengan akurat dan presisi. Sistem kendali pun mengalamni beberapa perkembangan menjadi berbagai jenis seperti optimal control, robust control, non linear control, dan lain lain. Salah satu perkembangannya adalah intelligent control, yaitu suatu pengendali adaptif yang berbasis pengetahuan. Terciptalah suatu sistem yang memiliki cara kerja mirip dengan sistem kerja jaringan saraf manusia yang dinamakan *Artificial Neural Network. Artificial Neural Network*. *Artificial Neural Network* (ANN) atau Jaringan Saraf Tiruan adalah suatu sistem adaptif yang dapat mengubah strukturnya untuk memecahkan masalah berdasarkan informasi eksternal maupun internal yang mengalir melalui jaringan tersebut. Sistem ini tidak menggunakan sebuah pemodelan matematis karena menggunakan pemodelan *blackbox*. Pemodelan ini dilakukan dengan cara menganalisa perilaku sistem jika diberikan variasi pasangan input dan output. Dari perilaku inilah maka jaringan saraf tiruan akan berusaha untuk menghasilkan output yang diinginkan.

2.2 Rumusan Masalah

a. Bagaimana konsep dasar jaringan saraf tiruan?
b. Bagaimana cara kerja metode pembelajaran jaringan saraf tiruan dengan menggunakan algoritma *backpropagation*?
c. Bagaimana suatu jaringan saraf tiruan dapat mengklasifikasikan suatu dataset?

2.3Tujuan Penelitian

Berdasarkan permasalahan yang telah dipaparkan, maka tujuan dari laporan ini adalah untuk menganalisis dan mempelajari cara kerja dari jaringan saraf tiruan pada dataset yang diberikan.

BAB II
DASAR TEORI

2.1Artificial Neural Network

Artificial Neural Network (ANN) atau Jaringan Saraf Tiruan lahir dari usaha manusia yang ingin memodelkan otak manusia untuk membuat suatu sistem yang paling sempurna. Model ANN pertama kali dikenalkan oleh Mc. Culloh dan Pitts sebagai komputasi dari aktivitas syarat. Hasil penelitian mereka menjadi dasar bagi penelitian di bidang ANN pada masa berikutnya.

Pada tahun 1958, Rosenblat, Widrow dan Hoff pertama kali menemukan aturan pembelajaran pada perceptrons. Minski dan Papert (1969) meyakini bahwa penggunaan perceptrons sangat terbatas yaitu hanya sebagai metode perhitungan dalam kehidupan nyata. Bernard Widrow menemukan neuron sederhana yang mirip dengan perceptron yang disebut ADALINE (neuron linier adaptif), dan jaringan multilayernya yang disebut MADALINE (multiple adaline). Selanjutnya, Widrow juga mengembangkan program pembelajaran terbimbing yang disebut dengan metode pembelajaran Least Mean Square (LMS) atau Widrow Hoff. Di era berikutnya, jaringan syaraf tiruan berkembang sedemikian rupa sehingga menemukan berbagai metode pembelajaran dan aturan pembelajaran.

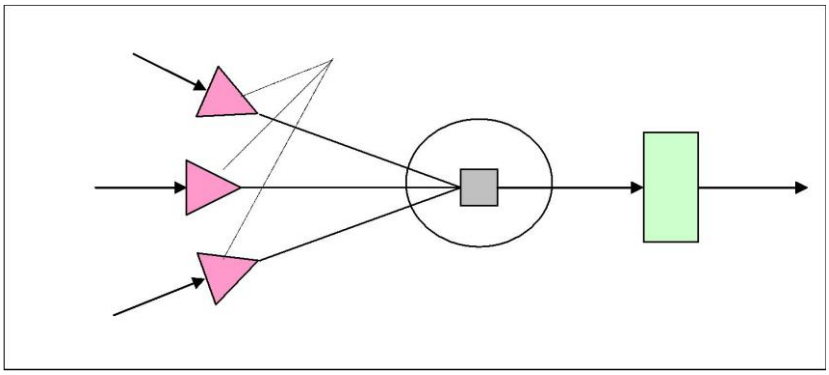
ANN merupakan jaringan yang dibuat dengan meniru jaringan syaraf manusia dengan diilhami oleh struktur dan cara kerja otak dan sel syaraf manusia. Sebuah neuron memiliki sebuah badan sel, pengirim sinyal dan penerima sinyal. Neuron biologis pada manusia dapat dilihat pada gambar 2.1

Gambar 2.1. Ilustrasi Neuron Biologis Manusia

Cara kerja dari sebuah neuron adalah akan bereaksi apabila potensial listrik mencapai suatu batasan tertentu. Neuron akan menjumlahkan sinyal yang masuk melalui dendrite yang dikalikan dengan pembobot sinapsis[1]. Proses pembelajaran terjadi dengan perubahann yang

terjadi pada sinasis. Sinyal yang masuk akan dijumlahkan dan dikonversi dengan suatu fungsi aktivitas yang kemudian akan mengeluarkan suatu sinyal pemicu yang dialirkan ke neuron lain.

Prinsip kerja dari neuron ini kemudian dimodelkan secara matematis. Model matematik orde pertama dari neuron dapat dilihat pada Gambar 2.2



Gambar 2.2. Model Matematik dari Neuron

Pemodelan matematika inilah yang menjadi dasar dari ANN. Elemen dasar dari sebuah ANN adalah sebuah neuron. Neuron ini akan mengubah sinyal msukan menjadi sebuah keluaran. Setiap neuron mempunyai suatu inputan yang memiliki bobotnya masing – masing. Sinyal kemudian akan dikonversi menggunakan sebuah fungsi aktivasi. Fungsi aktivasi yang digunakan dapat beragam seperti *Relu, sigmoid, tanh* dan lain lain

2.2 *Backpropagation*

Backpropagation adalah suatu algoritma ANN yang memiliki kekuatan utama pada klasifikasi suatu pola atau disebut sebagai *pattern recognition*. Jaringan syaraf *backpropagation* dapat digunakan untuk memprediksi luaran dari sebuah sistem. Pada mulanya data akan diberikan pada sistem. Jika sistem menghasilkan luaran yang tidak sesuai dengan yang diharapkan, maka *backpropagation* akan memodifikasi bobot pada hubungan neuron.[2]

Backpropagation menggunakan gradient descent untuk mengoptimasi nilai dari bobot tersebut. Gradient descent bertujuan adalah untuk terus mengambil sampel gradien parameter model ke arah yang berlawanan berdasarkan bobot w serta memperbarui secara konsisten hingga ANN dapat mencapai fungsi minimum global J (w). Gradient descent memiliki suatu parameter yaitu Learning Rate (α) yang mengatur seberapa besar perubahan bobot pada setiap epoch.

Pada gradient descent dikenal juga istilah momentum. Momentum adalah suatu metode agar mempertahankan arah dari gradient descent. Hal ini diperoleh dengan mengkombinasikan

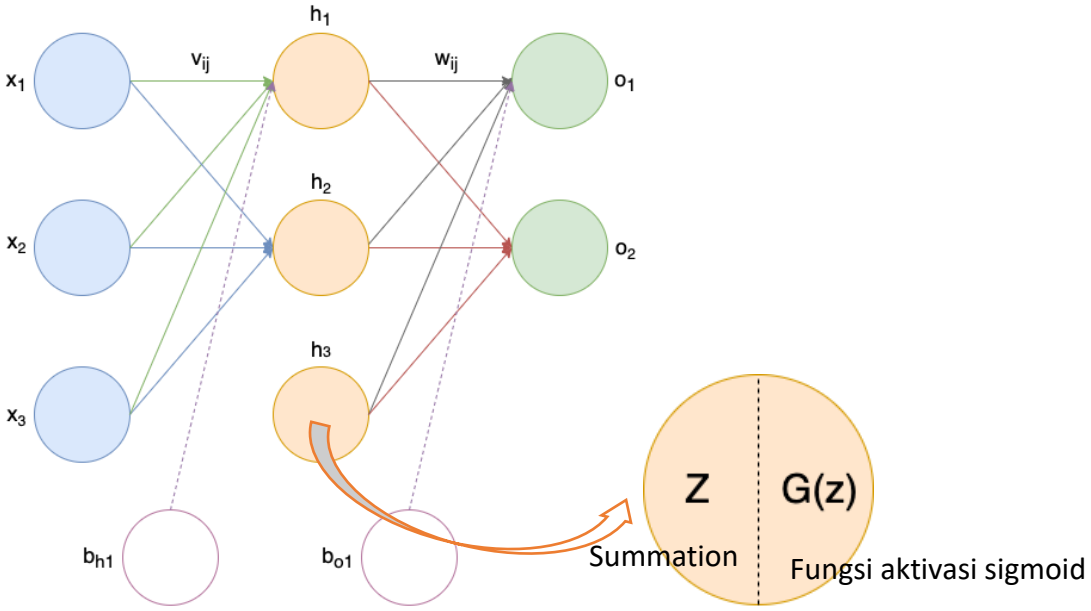
arah yang dikomputasi pada iterasi sekarang dengan iterasi sebelumnya. Besar pengaruh arah iterasi sebelumnya dapat diatur dengan mengatur koefisien momentum (μ**).**

ANN terdiri dari 3 layer, yaitu *input layer* yang memiliki neuron sebanyak I buah, *hidden layer* yang memiliki neuron sebanyak J buah, dan *output layer* yang memiliki neuron sebanyak K buah. Antara layer dihubungkan dengan bobot. Antara *input layer* dengan *hidden layer* dihubungkan dengan bobot **w**. Antara layer *hidden layer* dengan *output layer* dihubungkan dengan bobot **v**. Pada layer input dan hidden ditambah sebuah neuron bobot bias, yang berguna sebagai konstanta.[3]

Algoritma *backpropagation* dapat dibagi menjadi dua buat proses yaitu training dan testing. Dimana kedua proses tersebut dapat dibagi lagi menjadi:

1. Training
   a. Proses pengolahan data input (*feedforward*).
   b. Perhitungan error (*backpropagation*).
   c. Pembaruan bobot.
2. Testing
   a. Pengelompokkan nilai output (*Quantizing*).
   b. Perhitungan *Recognition Rate*.

Diagram sederhana pemodelan jaringan saraf tiruan yang digunakan dapat dilihat pada Gambar 2.3.



Gambar 2.3. Struktur Neural Networks Sederhana

Keterangan:

∂ x → Input layer ∂ h → Hidden layer

∂ o → Output layer

∂ i → Ukuran input layer

∂ j → Ukuran hidden layer

∂ v → Bobot hidden layer

∂ w → Bobot output layer

Berikut merupakan algoritma pemrograman *backpropagation*:

A. Inisialisasi bobot

Inisialisasi awal dari tiap bobot hubungan antar neuron dapat dilakukan menggunakan dua buah metode yaitu random dan nguyen widrow.

1. Random

Metode inisialisasi random adalah metode dimana bobot awal ditentukan secara acak. Hal ini dapat kita lakukan menggunakan fungsi random pada MATLAB. Inisialisasi awal yang acak dibutuhkan untuk menghasilkan symmetry breaking. Suatu strategi yang efektif untuk diterapkan pada inisialisasi random adalah menentukan batas atas dan batas bawah dari nilai acak tersebut. Hal ini dilakukan untuk memastikan bahwa bobot neuron tidak terlalu besar sehingga proses pembelajaran lebih efisien.

2. Nguyen Widrow

Metode nguyen widrow adalah sebuah algoritma modifikasi sederhana dari bobot yang mampu meningkatkan kecepatan proses pembelajaran. Algoritma nguyen-widrow adalah sebagai berikut:

- Menentukan besar faktor skala (β)

  $\beta = 0{,}7 * J^{\frac{1}{I}}$

  Dengan: J = Ukuran *hidden layer*; I = Ukuran *input layer*.
- Inisialisasi bobot secara random pada rentang -0.5 sampai 0.5
- Menghitung norm dari vektor bobot

  $\|v_j\| = \sqrt{\sum_{i=1}^{I} v_{ij}^2}$ dan $\|w_k\| = \sqrt{\sum_{j=1}^{J} w_{jk}^2}$

- Memperbaharui bobot

  $v_{ij} = \frac{\beta v_{ij}}{\|v_j\|}$ $w_{jk} = \frac{\beta w_{jk}}{\|w_k\|}$
- Mengatur bias sebagai bilangan acak antara β sampai - β

B. Data Preprocessing

Preprocessing pertama adalah normalisasi data yaitu proses mengubah data pada suatu feature agar mempunyai rentang yang sama yaitu diantara 0-1. Hal ini dilakukan agar seluruh feature mempunyai rentang yang sama sehingga tidak ada feature yang mendominasi satu dengan yang lainnya

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

C. Algoritma Pembelajaran

Algoritma untuk proses pembelajaran adalah sebagai berikut:

Selama kondisi stopping FALSE, maka untuk setiap pasangan pelatihan:

Proses *forward pass*

1. Komputasi *Input Layer*. Setiap x_i, i = 1,2,...,I
   - Menerima input x_i
   - Mengirimkannya ke semua *Hidden Layer* diatasnya.
2. Komputasi *Hidden Layer*. Setiap h_j, j = 1,2,...,J
   - $z\_in_j = v_{0j} + \Sigma x_{pi} v_{ij}$
   - $z_j = \frac{1}{1+e^{-z\_in_j}}$
3. Komputasi *Output Layer*. Setiap o_k, k = 1,2,...,K
   - $o\_in_k = w_{0k} + \Sigma z_j w_{jk}$
   - $o = \frac{1}{1+e^{-y\_in_k}}$
   - Kuantisasi output
     - 0 <= O_k < 0.3 → O_k = 0
     - 0.7 < O_k <= 1 → O_k = 1
     - Else → O_k = O_k.

Pada layer output diterapkan suatu threshold untuk membantu proses pembelajaran. Hal ini dilakukan karena sebuah output yang bernilai 1 atau 0 pada neuron output nyaris mustahil untuk tercapai tanpa adanya fungsi matematika tambahan. Threshold yang dipilih adalah 0.7 untuk batas atas dan 0.3 untuk batas bawah. Sehingga output yang bernilai lebih besar dari 0.7 akan dikonversi menjadi nilai biner 1 dan output yang bernilai lebih kecil dari 0.3 akan dikonversi menjadi nilai biner 0

Proses *backward pass*

1. Komputasi di *Output Layer*. Setiap o_k, k = 1,2,...,K
   - Menghitung informasi error:
     $\delta_k = (t_{pk} - y_{pk}).f'(y\_in_k)$
   - Menghitung besarnya koreksi bobot *output layer*:
     $\Delta w_{jk} = \alpha \delta_k z_j$
   - Menghitung besarnya koreksi bias output:
     $\Delta w_{0k} = \alpha \delta_k$
4. Komputasi di *Hidden Layer*. Setiap h_j, j = 1,2,...,J
   - Menghitung semua koreksi error:
     $\delta\_in_j = \Sigma \delta_k w_{jk}$
   - Menghitung nilai aktivasi koreksi error :
     $\delta_j = \delta_{in_j} f'(z\_in_j)$
   - Menghitung koreksi bobot unit hidden :
     $\Delta v_{ij} = \alpha \delta_j x_i$
   - Menghitung koreksi error bias unit hidden :
     $\Delta v_{0j} = \alpha \delta_j$

Proses memperbaharui bobot

1. Bobot ke *hidden layer*:
   $v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij}$
   $v_{0j}(baru) = v_{0j}(lama) + \Delta v_{0j}$
2. Bobot ke *output layer*:
   $w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk}$
   $w_{0k}(baru) = w_{0k}(lama) + \Delta w_{0k}$

Proses perhitungan error

Error dihitung dengan menggunakan fungsi kesalahan fungsi error kuadratis:

$$E = \frac{1}{2}\Sigma(t_{pk} - y_{pk})^2$$

D. Algoritma Pengujian

Setiap p = 1 sampai p = Pu :

Proses *feedforward*

3.1. Komputasi *Input Layer*. Setiap x_uji_{pi}, i = 1,2,...,I

- Menerima input x_uji_{pi}
- Mengirimkannya ke semua *Hidden Layer*.

3.2. Komputasi *Hidden Layer*. Setiap Z_j, j = 1,2,...,J

- $z\_in\_uji_j = v_{0j} + \Sigma x\_uji_{pi} v_{ij}$
- $z\_uji_j = \frac{1}{1+e^{-z\_in\_uji_j}}$

3.3. Komputasi *Output Layer*. Setiap y_k, k = 1,2,...,K

- $y\_in\_uji_k = w_{0k} + \Sigma z\_uji_j w_{jk}$
- $y\_uji_{pk} = \frac{1}{1+e^{-y\_in\_uji_k}}$
- Kuantisasi output
  - 0 <= y_uji_{pk} < 0.3 → y_rec_{pk} = 0
  - 0.7 < y_uji_{pk} <= 1 → y_rec_{pk} = 1
  - Else → $y\_rec_{pk} = y\_uji_{pk}$.

3.4. Menghitung *recognition rate*:

$$RR = \frac{banyaknya\ data\ output\ yang\ benar}{banyaknya\ data\ pengujian} \times 100\%$$

Output dikatakan benar jika nilai biner nya sama dengan binernya target.

2.3 Sistem Kendali Berbasis Aritificial Neural Networks

ANN dapat digunakan pada sistem kendali. Pengendali ini akan tidak berbasis pada sebuah model matematis[4]. Hal ini dilakukan karena terdapat beberapa plant yang sulit untuk dimodelkan secara matematis. Fungsi dari sebuah plant biasanya memiliki beberapa sifat yaitu:

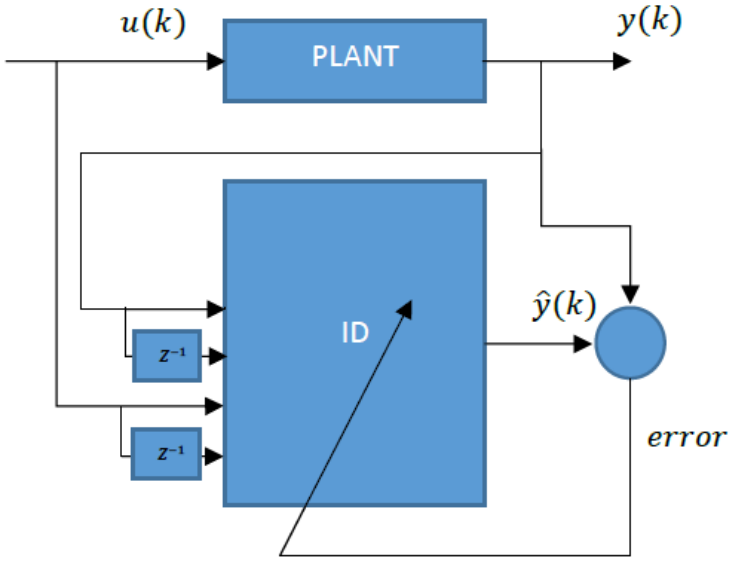
- Non Linear
- Cross coupling
- Underactuated
- Tidak mampu ditulis model matematikanya

ANN akan digunakan untuk membuat sebuah fungsi transfer sebuah sistem rumit dimana sistem rumit adalah sistem yang belum tentu mempunyai fungsi transfer yang jelas. Fungsi transfer dapat berupa fungsi spatial dan fungsi waktu. Fungsi transfer ini disimpan dalam bentuk bobot dalam struktur arsitektur ANN tersebut.

Terdapat beberapa perbedaan yang harus kita buat ketika menggunakan ANN untuk sistem kendali. Komponen perubahan tersebut dapat dilihat pada tabel dibawah ini

	Pattern Recognition	**Sistem Kendali**
Ukuran Sukses	Recognition Rate	MSE/MSSE
Dimensi Input	Relatif Besar	Relatif Kecil
Normalisasi	Value berkisar 0 ~ 1	Value Berkisar -1 ~ 1
Input Range	Memiliki range yang sama	Memiliki range yang berbeda

Sistem kendali menggunakan MSE/MSSE untuk memberikan gambaran antara hasil luaran dari sistem dengan data seharusnya. Dimensi input biasanya tidak banyak karena hanya bergantung pada jumlah input dan output dari plant yang digunakan. Normalisasi pada sistem kendali ANN juga dapat mempunyai nilai -1 karena daerah kerja sistem kendali dapat bernilai negative. Setiap dimensi input juga dapat memiliki range yang berbeda, untuk mengatasi hal ini kita menggunakan proses penyamaan range tersebut yaitu metode Z-Score



Gambar 2.4. Penggunaan ANN pada identifikasi sistem

Pada penggunaan sistem kendali berbasis ANN, input input akan mendeskripsikan sistem yang akan masuk ke dalam proses identifikasi. Ketidaksesuaian antara output dengan nilai sessunguhhnya akan dikirim balik untuk menghasilkan nilai error yang akan mengubah nilai bobot pada parameter neural network sehingga jaringan dapat merepresentasikan sistem dengan baik

BAB III
IDENTIKASI SISTEM DAN PENGENDALI MENGGUNAKAN ARTIFICIAL NEURAL NETWORK

3.1Pembuatan Database

Data dari plant akan di generate dengan menghasilkan sebanyak 3010 angka random. 3010 angka tersebut kemudian kita masukkan kedalam persamaan plant sehingga menghasilkan sebanyak 3010 data output. Pasangan input dan output ini akan menjadi database untuk proses identifikasi sistem.

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

Dataset dihasilkan dengan dua buah jenis acak yaitu acak biasa dan acak sinusoid. Berikut merupakan program MATLAB untuk pembuatan database:

```
%% Random Output
rng(0,'twister'); %Mulai seed dan fungsi randomizer
u = (1-(-1)).*rand(3010,1)-1; %nilai random

y=0;
for i = 2:length(u)
    if i == 1 then
        y(i)=0
    end
y(i)=(1/(1+y(i-1)^2)) + 0.25*u(i) - 0.300*u(i-1);
end

%% Sinusoid Output
u_sinusoid = sind(0:(length(u)-1)); %nilai random sinusoid

y_sinusoid = 0;
for i = 2:length(u_sinusoid)
    if i == 1 then
        y_sinusoid(i)=0
    end
y_sinusoid(i)=(1/(1+y_sinusoid(i-1)^2))+ 0.250*u_sinusoid(i)-
0.300*u_sinusoid(i-1);
end
```

Program akan menghasilkan 3010 buah data untuk acak biasa pada variable y dan 3010 buah data acak sinusoid pada variable y_sinusoid. **uk** merupakan data input **u(k)** yang didapat melalui proses randomisasi yang telah di normalisasi. Sedangkan **yk** merupakan target **y(k)** yang didapat melalui fungsi

Data kemudian dinormalisasi menggunakan metode max-min agar mendapatkan nilai random pada range -1 dan +1.

3.2 Proses Pelatihan

Pada proses learning identifikasi plant, data yang digunakan adalah data yang berasal dari generate data sebelumnya. Masukan dari sistem akan berjumlah 5 yaitu $y(k-1)$,$y(k-2)$ $x(k)$, $x(k-2)$,$x(k-1)$. Data dibagi menjadi 50:50 menjadi 2 bagian yaitu training dan testing.

Berikut merupakan program untuk membagi data menjadi bagian training dan testing:

```
%% Data Splitting
input = [y(2:1505)' y(1:1504)' u(3:1506) u(2:1505) u(1:1504)];
target = y(3:1506)';
[input_row,input_col] = size(input);
[target_row,target_col] = size(target);
n_train = length(input);
```

Inisialisasi bobot menggunakan metode nguyen-widrow dengan program MATLAB sebagai berikut:

```
%% Nguyen Widrow
 beta = 0.7 * z_size^(1/x_size);
 weight_xz = rand(x_size, z_size) - epsilon_init;
 weight_zy = rand(z_size, y_size) - epsilon_init;
for i = 1:z_size
    norma(i) = sqrt(sum(weight_xz(:,i).^2));
    weight_xz(:,i) = beta*((weight_xz(:,i))/norma(i));
end

bias_xz = (rand(1, z_size) - epsilon_init) * beta;
bias_zy = rand(1, y_size) - epsilon_init;
```

Kemudian program akan melakukan proses feedforward dan backpropagation yang akan mengupdate bobot sehingga sistem dapat menghasilkan luaran yang sama dengan persamaan plant yang menghasilkan data pada database.

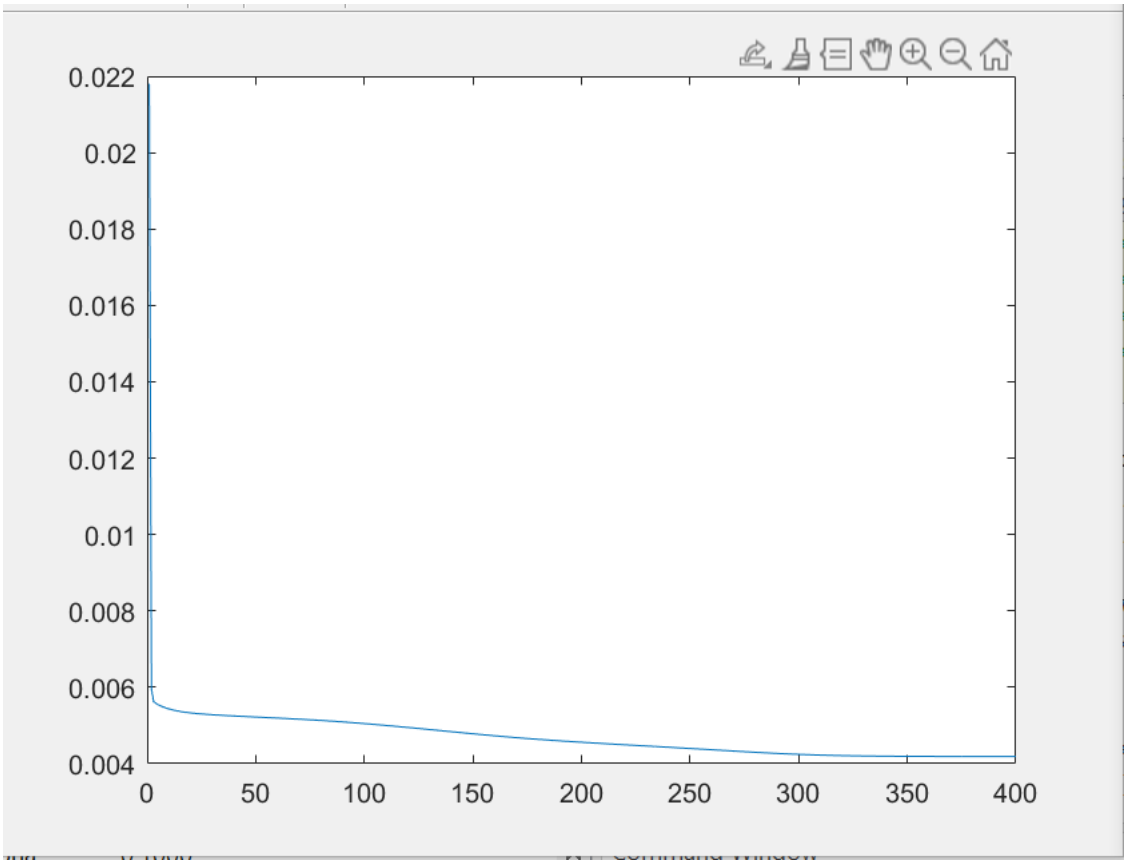
Program MATLAB untuk feedforward dan backpropagation adalah sebagai berikut:

```
%% Backpropagation
while stop == 0 && epoch_count <= epoch
    for n = 1:data_count
        %Forward Pass
        %Input -> Hidden
        xi = input(n,:);
        ti = target(n,:);

        z_in = bias_xz + xi*weight_xz;

        for m=1:z_size
            z(1,m) = 1/(1+exp(-z_in(1,m)));
        end

        %Hidden -> Output
```

```
        y_in = bias_zy + z*weight_zy;

        for l=1:y_size
           yk(1,l) = 1/(1+exp(-y_in(1,l)));
        end

        %Backward Pass

        %Momentum calculation
        momentum_zy = miu*delta_zy_old;
        momentum_xz = miu*delta_xz_old;

        %Output->Hidden
        do_k = (yk - ti).*(yk).*(1-yk);


        delta_zy = alpha.*z'.*do_k + momentum_zy;
        delta_zy_old = delta_zy;
        delta_zy_bias = alpha.*do_k + miu*delta_zy_bias_old;
        delta_zy_bias_old = delta_zy_bias;

        %Hidden->Input
        sigma_j = do_k * weight_zy';

        do_j = sigma_j.* z.*(1-z);

        delta_xz = alpha.* xi' .* do_j + momentum_xz;
        delta_xz_old = delta_xz;
        delta_xz_bias = alpha .* do_j + miu*delta_xz_bias_old;
        delta_xz_bias_old = delta_xz_bias;

        %Weight Update
        weight_zy = weight_zy - delta_zy;
        weight_xz = weight_xz - delta_xz;
        bias_zy = bias_zy - delta_zy_bias;
        bias_xz = bias_xz - delta_xz_bias;

        %Cost function
        error(1,n) = 0.5*sum((yk-ti).^2);

        %Momentum update
        delta_zy_old = delta_zy;
        delta_xz_old = delta_xz;
    end
    error_per_epoch(1,epoch_count) = sum(error)/input_row;

    if error_per_epoch(1,epoch_count) < error_target
            stop = 1;
    end

    epoch_count = epoch_count+1;

end
avgerrortrain = sum(error)/n_train;
% epoch = epoch - 1;
% figure;
plot(error_per_epoch);
```

```
% scatter(epoch,error_per_epoch)
```

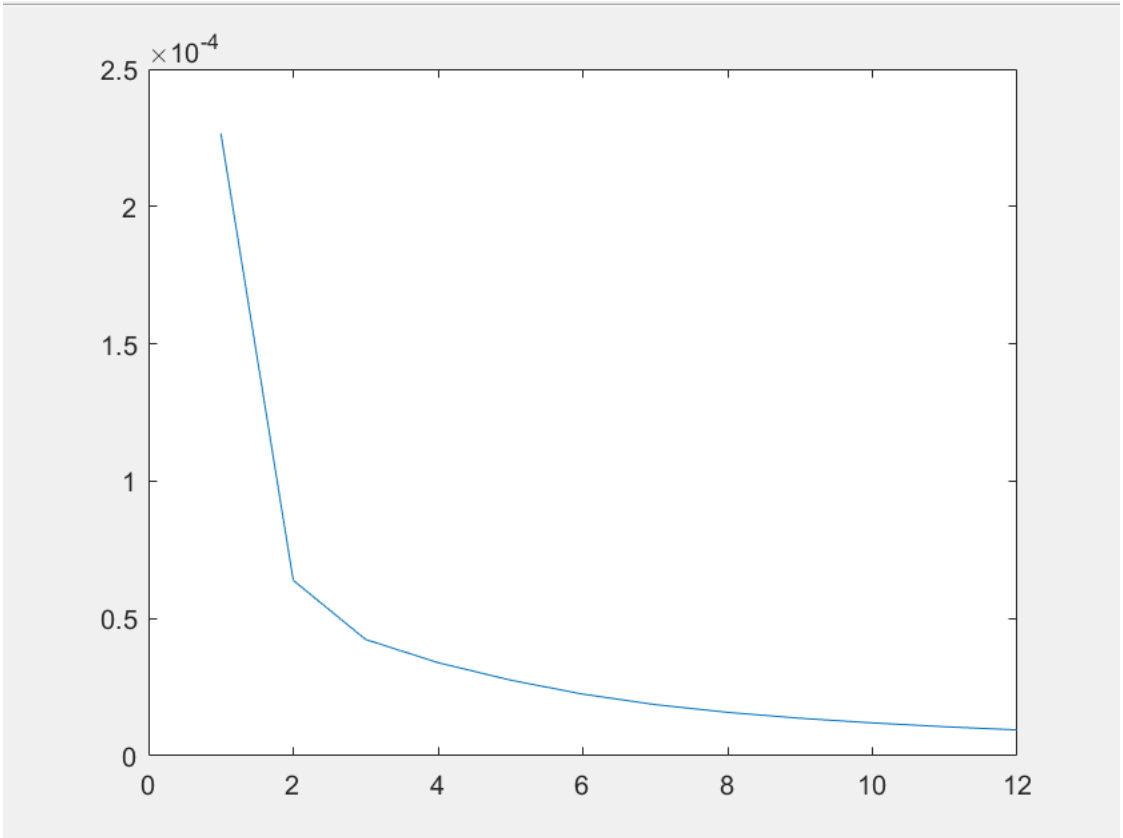
Pada setiap epoch, bobot pada jaringan akan diupdate. Semakin banyak neuron hidden layer yang digunakan, semakin kecil besar epoch yang dibutuhkan untuk mendapatkan nilai output yang sesuai dengan target. Hubungan antara error dan epoch kemudian di plot untuk dilihat penurunan nilai error pada jaringan. Berikut merupakan plot hubungan antara epoch dan error pada input acak.



Gambar 3.1. Grafik Hasil Learning Sistem Identifikasi Input Acak

Dapat dilihat dari grafik bahwa penurunan error sudah tidak mengalami perubahan yang signifikan setelah epoch 100. Sehingga dapat disimpulkan bahwa perubahan bobot lebih dominan terjadi pada epoch dibawah 100.

Berikut merupakan plot hubungan antara epoch dan error pada input sinusoidal



.

Gambar 3.2. Grafik Hasil Learning Sistem Identifikasi Input Sinusoidal

Dapat dilihat dari grafik bahwa penurunan error sudah tidak mengalami perubahan yang signifikan setelah epoch 3. Sehingga dapat disimpulkan bahwa perubahan bobot lebih dominan terjadi pada epoch dibawah 3. Epoch yang digunakan berjumlah sedikit karena ketika jaringan tetap melakukan proses pelatihan namun error sudah kecil, maka nilai error dapat naik lagi karena jaringan akan keluar dari konvergensi dan menjadi divergen.

3.3 Proses Pengujian

Data dari 1506 sampai 3010 kemudian digunakan untuk proses testing. Hal ini dilakukan untuk mengetahui peforma dari jaringan yang telah dibentuk pada proses pelatihan. Berikut merupakan program MATLAB untuk proses pengujian:

```
%% ANN Input Acak (testing)
% Data Input
input_testing = [y(1506:3009)' y(1505:3008)' u(1507:3010)
u(1506:3009) u(1505:3008)];
target_testing = y(1507:3010)';
[input_row_test,input_col_test] = size(input);
[target_row_test,target_col_test] = size(target);
n_test = size(input_testing);
n_test = n_test(2);
```

```
3:1506 %y(k)
2:1505 %y(k-1)
1:1504 %y(k-2)

1507:3010 %y(k)
1506:3009 %y(k-1)
1505:3008 %y(k-2)

input = normalize(input)

%% Hyperparameter & Declare Variable
%Variable
data_count_test = input_row_test;
true_count = 0;

for n = 1:data_count_test
    %Forward Pass
    %Input -> Hidden
    x_test = input_testing(n,:);
    t_test = target_testing(n,:);

    z_in_test = bias_xz + x_test*weight_xz;

    for m=1:z_size
        z_test(1,m) = 1/(1+exp(-z_in_test(1,m)));
    end

    %Hidden -> Output
    y_in_test = bias_zy + z_test*weight_zy;

    for l=1:y_size
        y_test(n,l) = 1/(1+exp(-y_in_test(1,l)));
    end

error_test(1,n) = 0.5*sum((y_test(n,1) - t_test).^2);
end

avgerrortest = sum(error_test)/n_test;
MSE_test = (sum((t_test-y_test).^2))/n_test;
disp("Error average test = "+ avgerrortest *100 +" %");
disp("MSE testing = "+ MSE_test*100 +" %");

figure;
plot(y_test,'o');
hold on
plot(target_testing,'x');
xlim([0 n_test]); xlabel('k'); ylabel('y(k)'); legend('Output
ANN','Output Plant');
hold off
```

Kemudian dari hasil luaran sistem, kita hitung Error Average dan Mean Squared Error (MSE) sebagai metric penilaian peforma dari jaringan kita. Dari hasil perhitungan, diperoleh nilai average error dan MSE sebagai berikut:

1. Input Acak

Error Average : 1.6075%

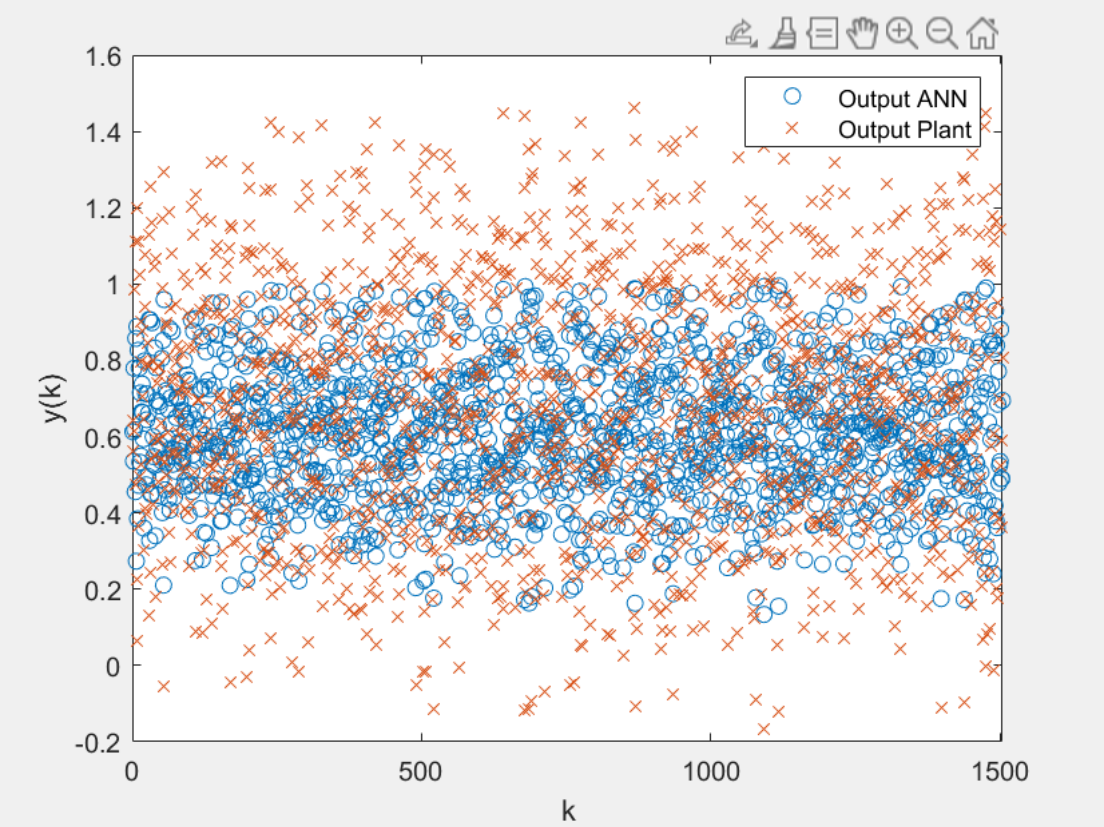
MSE : 7.0599%

2. Input Sinusoidal

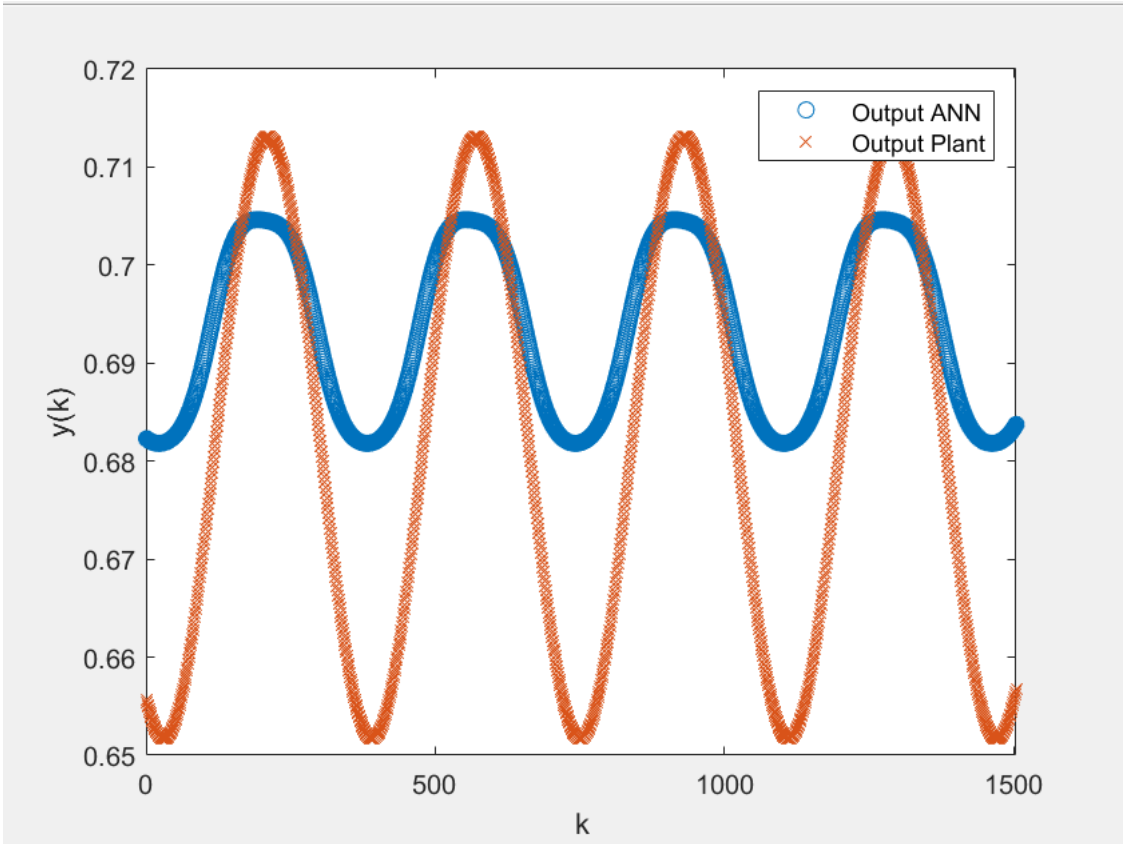
Error Average : 0.016096%

MSE : 0.14047%

Penulis juga membuat plot hubungan antara output ANN dengan output plant untuk melihat luaran dari sistem kita. Berikut merupakan plot output dari ANN dan plant pada input acak.



Gambar 3.3. Plot Output Input Acak



Gambar 3.4. Plot Output Input Sinusoidal

Dapat dilihat bahwa jaringan lebih mudah untuk mengidentifikasi sistem yang memiliki input sinusoidal dibandingkan input acak. Hal ini disebabkan karena dataset pada input sinusoidal memiliki bentuk yang lebih berpola sehingga jaringan lebih mudah mengikuti karena ANN pada dasarnya merupakan algoritma pattern recognition.

BAB IV
KESIMPULAN

Setelah melakukan eksperimen identifikasi sistem dengan input acak dan input sinusoidal, diperoleh kesimpulan sebagai berikut:

1. *Backpropagation* merupakan metode ANN yang merupakan metode pembelajaran sistem yang mengubah nilai bobot dari neuron relatif terhadap error yang diperoleh
2. ANN dapat digunakan untuk mengidentifikasi sistem dan pengendali pada sistem kendali
3. Nilai masukan yang lebih berpola teratur menghasilkan jaringan yang lebih baik dibandingkan nilai masukan yang acak
4. Jumlah neuron hidden layer berpengaruh terhadap hasil luaran ANN dimana semakin banyak jumlah neuron akan memberikan hasil yang lebih baik namun perlu diwaspadai pengaruh dari jumlah neuron yang terlalu banyak karena dapat menyebabkan overfitting

DAFTAR REFERENSI

[1] D. Wahyuningsih, I. Zuhroh, and - Zainuri, "Prediksi Inflasi Indonesia Dengan Model Artificial Neural Network," *J. Indones. Appl. Econ.*, vol. 2, no. 2, pp. 2–2008, 2008, doi: 10.21776/ub.jiae.2008.002.02.7.

[2] D. B. Kusumoputro, "BACKPROPAGATION," 2021.

[3] "Artificial_Neural_Network @ Www.Saedsayad.Com." [Online]. Available: http://www.saedsayad.com/artificial_neural_network.htm.

[4] P. D. B. Kusumoputro and U. Indonesia, "Penggunaan Artificial Neural Networks dalam Sistem Kendali."

Lampiran 1.

Berikut merupakan kode lengkap MATLAB untuk eksperimen berikut:

```
clc;
clear;
close all;

%% Random Input
rng(0,'twister');
u = (1-(-1)).*rand(3010,1)-1;

y=0;
for i = 2:length(u)
    if i == 1 then
        y(i)=0
    end
y(i)=(1/(1+y(i-1)^2)) + 0.25*u(i) - 0.300*u(i-1);
end

%% Sinusoidal Input
u_sinusoid = sind(0:(length(u)-1));

y_sinusoid = 0;
for i = 2:length(u_sinusoid)
    if i == 1 then
        y_sinusoid(i)=0
    end
y_sinusoid(i)=(1/(1+y_sinusoid(i-1)^2))+ 0.250*u_sinusoid(i)-
0.300*u_sinusoid(i-1);
end

%% Data Splitting

input = [y(2:1505)' y(1:1504)' u(3:1506) u(2:1505) u(1:1504)];
target = y(3:1506)';
[input_row,input_col] = size(input);
[target_row,target_col] = size(target);
n_train = size(input);
n_train = n_train(2);

%% Normalisasi
input = normalize(input)

%% Hyperparameter & Declare Variable

%Hyperparameter
x_size = input_col; %Input layer
z_size = 7;  %Hidden layer
y_size = target_col; %Output layer

alpha = 0.1;
epoch = 400;
miu   = 0.3;

%Variable
```

```
stop = 0;
error_target = 0.00001;
data_count = input_row;
epoch_count = 1;

delta_zy_old = 0;
delta_zy_bias_old = 0;
delta_xz_old = 0;
delta_xz_bias_old = 0;
%% Initialisasi

%Random Init
rng(200) %Seed
epsilon_init = 0.5; %Range random number

 beta = 0.7 * z_size^(1/x_size);
 weight_xz = rand(x_size, z_size) - epsilon_init;
 weight_zy = rand(z_size, y_size) - epsilon_init;
for i = 1:z_size
    norma(i) = sqrt(sum(weight_xz(:,i).^2));
    weight_xz(:,i) = beta*((weight_xz(:,i))/norma(i));
end

bias_xz = (rand(1, z_size) - epsilon_init) * beta;
bias_zy = rand(1, y_size) - epsilon_init;

%% Backpropagation
while stop == 0 && epoch_count <= epoch
    for n = 1:data_count
        %Forward Pass
        %Input -> Hidden
        xi = input(n,:);
        ti = target(n,:);

        z_in = bias_xz + xi*weight_xz;

        for m=1:z_size
            z(1,m) = 1/(1+exp(-z_in(1,m)));
        end

        %Hidden -> Output
        y_in = bias_zy + z*weight_zy;

        for l=1:y_size
           yk(n,l) = 1/(1+exp(-y_in(1,l)));
        end

        %Backward Pass

        %Momentum calculation
        momentum_zy = miu*delta_zy_old;
        momentum_xz = miu*delta_xz_old;

        %Output->Hidden
%        for l=1:y_size
          do_k = (yk(n,1) - ti).*(1+yk(n,1)).*(1-yk(n,1)).*0.5;
%        end
```

```
        delta_zy = alpha.*z'.*do_k + momentum_zy;
        delta_zy_old = delta_zy;
        delta_zy_bias = alpha.*do_k + miu*delta_zy_bias_old;
        delta_zy_bias_old = delta_zy_bias;

        %Hidden->Input
        sigma_j = do_k * weight_zy';
        do_j = sigma_j.* z.*(1-z).*0.5;


        delta_xz = alpha.* xi' .* do_j + momentum_xz;
        delta_xz_old = delta_xz;
        delta_xz_bias = alpha .* do_j + miu*delta_xz_bias_old;
        delta_xz_bias_old = delta_xz_bias;

        %Weight Update
        weight_zy = weight_zy - delta_zy;
        weight_xz = weight_xz - delta_xz;
        bias_zy = bias_zy - delta_zy_bias;
        bias_xz = bias_xz - delta_xz_bias;

        %Cost function
        error(n,1) = 0.5*sum((yk(n,1)-ti).^2);

        %Momentum update
        delta_zy_old = delta_zy;
        delta_xz_old = delta_xz;
    end
    error_per_epoch(1,epoch_count) = sum(error)/input_row;
    if error_per_epoch(1,epoch_count) < error_target
            stop = 1;
    end

    epoch_count = epoch_count+1;

end
avgerrortrain = sum(error)/n_train;
plot(error_per_epoch);
%% ANN Input Acak (testing)

% Data Input
input_testing = [y(1506:3009)' y(1505:3008)' u(1507:3010)
u(1506:3009) u(1505:3008)];
target_testing = y(1507:3010)';
[input_row_test,input_col_test] = size(input);
[target_row_test,target_col_test] = size(target);
n_test = size(input_testing);
n_test = n_test(2);

3:1506 %y(k)
2:1505 %y(k-1)
1:1504 %y(k-2)

1507:3010 %y(k)
1506:3009 %y(k-1)
1505:3008 %y(k-2)
```

```
%% Normalisasi
input = normalize(input)
%% Hyperparameter & Declare Variable

%Variable
data_count_test = input_row_test;
true_count = 0;

for n = 1:data_count_test
    %Forward Pass
    %Input -> Hidden
    x_test = input_testing(n,:);
    t_test = target_testing(n,:);

    z_in_test = bias_xz + x_test*weight_xz;

    for m=1:z_size
        z_test(1,m) = 1/(1+exp(-z_in_test(1,m)));
    end

    %Hidden -> Output
    y_in_test = bias_zy + z_test*weight_zy;

    for l=1:y_size
        y_test(n,l) = 1/(1+exp(-y_in_test(1,l)));
    end

error_test(1,n) = 0.5*sum((y_test(n,1) - t_test).^2);

end

avgerrortest = sum(error_test)/n_test;
MSE_test = (sum((t_test-y_test).^2))/n_test;
disp("Error average test = "+ avgerrortest *100 +" %");
disp("MSE testing = "+ MSE_test*100 +" %");

figure;
plot(y_test,'o');
hold on
plot(target_testing,'x');
xlim([0 n_test]); xlabel('k'); ylabel('y(k)'); legend('Output
ANN','Output Plant');
hold off
```