Merancang Pengendali Pressure Process Rig Menggunakan Neural Network Berbasis Error dengan Metode Pembelajaran Backpropagation

Designing a Pressure Process Rig Controller Using Error-Based Neural Network with Backpropagation Learning Method

Muhammad Ashari[*] [1], Benyamin Kusumoputro[2], Dian Putra Saragih[3], Selly Annisa Binti Zulkarnain[4], Saras Pratama[5]

[1]*Department of Electrical Engineering, Faculty of Engineering, Universitas Negeri Medan, Indonesia*
[2]*Department of Electrical Engineering, Faculty of Engineering, Universitas Indonesia, Indonesia*
[3]*Department of Electrical Engineering, Faculty of Engineering, Universitas Negeri Medan, Indonesia*
[4]*Department of Electrical Engineering, Faculty of Engineering, Universitas Negeri Medan, Indonesia*
[5]*Department of Electrical Engineering, Faculty of Engineering, Universitas Negeri Medan, Indonesia*
Email: [1]ashari@unimed.ac.id, [2]kusumo@ee.ui.ac.id, [3]dian@unimed.ac.id, [4]sellyannisa@unimed.ac.id, [5]saraspratama@unimed.ac.id
Received 30 June 2025; Revised 25 July 2025; Accepted 28 July 2025

Abstrak - Jaringan Saraf Tiruan (NN) merupakan sebuah paradigma dalam pemrosesan informasi yang dimodelkan berdasarkan sistem *neuron*, menyerupai cara otak manusia memproses informasi. Penelitian ini membahas penerapan jaringan saraf dengan algoritma pembelajaran *backpropagation* yang diimplementasikan dalam simulasi sebagai pengendali suatu sistem. *Pressure Process Rig 38-714* digunakan sebagai model *plant* dalam penelitian ini. Pengendali berbasis Jaringan Saraf Tiruan ini beroperasi dengan menghitung kesalahan antara keluaran aktual sistem dan masukan referensi yang telah ditentukan, dimana sinyal hasil selisih antara keluaran *plant* dan sinyal referensi menjadi masukan utama bagi sistem kendali. Penelitian ini memperkenalkan pendekatan pengendalian berbasis eror dengan pelatihan dua tahap, yakni konfigurasi seri-paralel dan paralel, serta disertai metode faktor pengali untuk menjaga kestabilan sinyal kendali dalam batas kerja *plant*. Berbagai eksperimen dilakukan untuk mengevaluasi kinerja sistem, termasuk pengujian dengan dan tanpa gangguan, serta pembandingan terhadap metode *Direct Inverse Control Neural Network* (DIC NN). Hasil pengujian menunjukkan bahwa pendekatan ini memberikan solusi adaptif dan tahan terhadap eror, dengan performa dan ketahanan yang lebih baik dalam kondisi terpengaruh gangguan dibandingkan metode DIC NN. Keberhasilan sistem diukur dari kesesuaian antara keluaran *plant* dengan sinyal referensi dan kemampuan sistem kendali dalam meredam gangguan secara efektif.

Kata kunci: Pengendali, *Neural Network, Backpropagation*

Abstract - *Artificial Neural Networks (NN) represent a paradigm in information processing modeled after the neural system, similar to how the human brain processes information. This study discusses the application of neural networks using the backpropagation learning algorithm, implemented in simulation as a controller for a system. The Pressure Process Rig 38-714 is used as the plant model in this research. The NN-based controller operates by calculating the error between the actual system output and the predefined reference input, where the error signal—the difference between the plant's output and the reference signal—becomes the main input for the control system. This research introduces an error-based control approach with a two-stage training process, namely series-parallel and parallel configurations, accompanied by a scaling factor method to maintain the stability of the control signal within the plant's operating limits. Various experimental tests were conducted to evaluate the system's performance, including testing with and without disturbances, as well as comparing it with the Direct Inverse Control Neural Network (DIC NN) method. The results show that this approach provides an adaptive and*

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381

robust solution to pressure system dynamics, with better performance and resilience under disturbance conditions compared to the DIC NN method. The system's success is measured by the similarity between the plant's output and the reference signal and the control system's ability to effectively mitigate disturbances.

Keywords: *Controller, Neural Network, Backpropagation.*

1. PENDAHULUAN

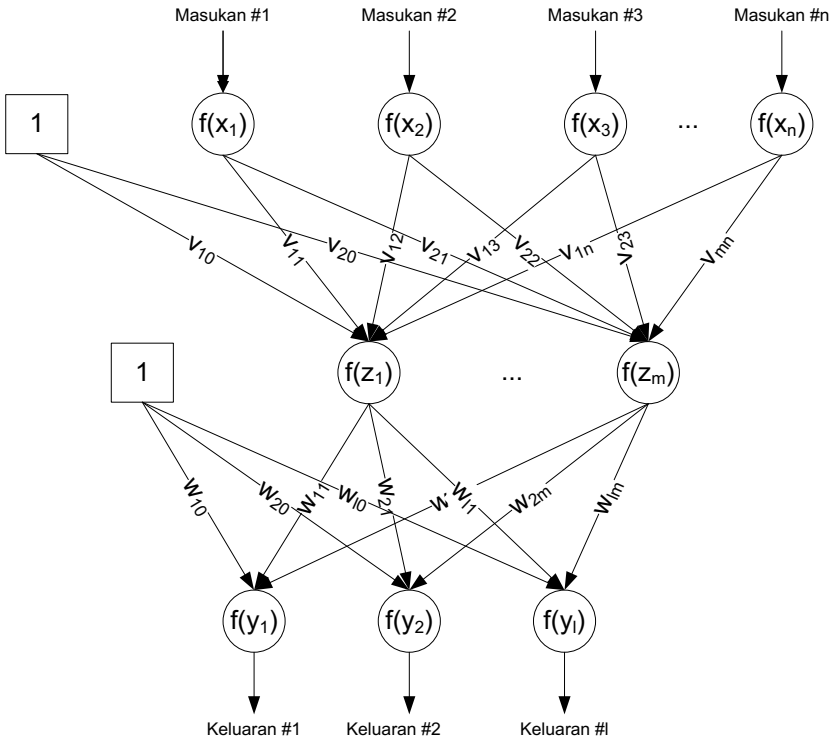
Jaringan saraf, atau Neural Network (NN), merupakan salah satu teknik pembelajaran mesin yang meniru cara kerja neuron biologis dalam memahami dan mengolah informasi. Paradigma ini memainkan peran penting dalam berbagai aplikasi, termasuk kontrol sistem industri, berkat kemampuan adaptif dan fleksibilitasnya dalam menghadapi data nonlinier serta kompleksitas sistem industri yang semakin meningkat[1][2]. Dalam penerapannya, metode pembelajaran backpropagation adalah salah satu teknik yang paling umum digunakan untuk melatih jaringan saraf, yang memungkinkan penyesuaian bobot neuron berdasarkan sejauh mana keluaran yang dihasilkan menyimpang dari sinyal referensi atau target [3][4].

Metode *backpropagation* merupakan algoritma pelatihan fundamental untuk jaringan saraf tiruan yang bekerja berdasarkan prinsip *supervised learning* [5]. Proses ini terdiri dari dua fase utama: *forward propagation* dan *backward propagation*. Pada fase *forward*, data masukan diproses melalui setiap lapisan jaringan, di mana setiap neuron menghitung jumlah terbobot dari inputnya, kemudian menerapkan fungsi aktivasi untuk menghasilkan keluaran. Keluaran jaringan kemudian dibandingkan dengan target aktual menggunakan fungsi *loss* (misalnya *Mean Squared Error*) untuk menghitung besarnya kesalahan [6]. Fase ini bersifat deterministik dan berfungsi untuk mengevaluasi kinerja jaringan terhadap data pelatihan [7].

Pada fase *backward propagation*, kesalahan yang dihitung dipropagasikan mundur melalui jaringan untuk menghitung turunan parsial (*gradien*) fungsi *loss* terhadap setiap bobot jaringan menggunakan aturan rantai (*chain rule*) dari kalkulus diferensial [8]. Gradien ini menunjukkan arah dan besaran penyesuaian yang diperlukan untuk meminimalkan kesalahan. Bobot kemudian diperbarui menggunakan metode optimasi seperti *Gradient Descent* ($W = W - \alpha \cdot \nabla E$), di mana α adalah *learning rate* yang mengontrol kecepatan pembelajaran [6]. Proses ini diulang secara iteratif hingga konvergensi tercapai, baik berdasarkan jumlah *epoch* maksimum atau toleransi kesalahan yang ditentukan [9]. Kelebihan *backpropagation* terletak pada kemampuannya menangani jaringan dengan banyak lapisan (*deep learning*) [10], meskipun memiliki kelemahan seperti risiko terjebak di *local minima* dan ketergantungan pada inisialisasi bobot awal serta pemilihan *hyperparameter* yang tepat [11].

Peneliti menggunakan metode *backpropagation* karena metode ini telah teruji secara luas sebagai algoritma pelatihan yang stabil dan efisien untuk sistem kontrol dinamik dengan kompleksitas menengah [12]. Metode ini memberikan keunggulan dalam implementasi sistem kontrol industri berbasis *Pressure Process Rig* seperti: interpretabilitas parameter, komputasi yang ringan, dan konvergensi yang dapat diprediksi [13]. Penggunaan *backpropagation* dalam *neural network* dimulai dari ide dasar untuk memperkecil kesalahan keluaran. Proses ini melibatkan tiga langkah kunci: pertama, menghitung selisih antara keluaran yang diinginkan dan keluaran aktual; kedua, menyesuaikan bobot *neuron* berdasarkan perhitungan tersebut; dan ketiga, mempropagasi sinyal kesalahan kembali melalui jaringan untuk mengoptimalkan bobot di semua lapisan [3][14]. Hasil penelitian menunjukkan bahwa penggunaan jaringan saraf berbasis metode *backpropagation* dalam sistem kontrol, seperti pada *Pressure Process Rig*, menghasilkan kinerja yang lebih baik jika dibandingkan dengan pengendali konvensional yang sering kali tidak mampu mengatasi kompleksitas serta ketidaklinieran dalam sistem [15][16].

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381

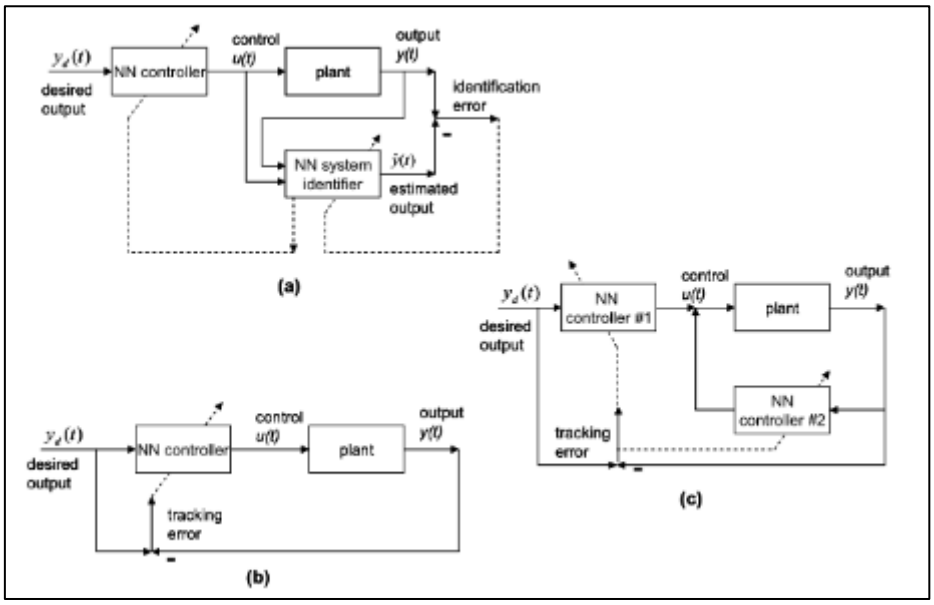


Gambar 1. Rancangan sebuah jaringan syaraf tiruan

Sistem kendali berbasis jaringan saraf mampu menyesuaikan parameternya secara otomatis dalam menghadapi perubahan karakteristik sistem atau gangguan dari luar. Hal ini menjadi keunggulan signifikan dibandingkan dengan pengendali PID konvensional, di mana perubahan karakteristik sistem memerlukan recalibrasi manual yang kompleks [17]. Selain itu, kemampuan jaringan saraf dalam memproses data dalam jumlah besar dan mengadaptasi algoritma kendalinya sesuai dengan data yang baru, menjadikannya solusi yang efektif dalam menciptakan sistem industri yang lebih efisien dan responsif terhadap kebutuhan pasar yang terus berkembang [18].

Penelitian menunjukkan bahwa ketahanan sistem yang menggunakan jaringan saraf, terutama yang dilatih dengan backpropagation, mampu menangani gangguan eksternal secara signifikan, sehingga meningkatkan akurasi dan stabilitas keluaran sistem kendali [1][19]. Kesuksesan dalam uji coba dapat diukur melalui kesamaan antara keluaran yang dihasilkan dan sinyal referensi, serta kemampuannya beradaptasi dengan gangguan [20][21]. Sebagai contoh, dalam pengendalian sistem yang kompleks, jaringan saraf memberikan fleksibilitas yang lebih dibandingkan dengan metode konvensional dan menjadi alat yang berharga dalam penerapan teknologi terkini di industri [22]. Dengan ketidakpastian dan kompleksitas yang terus bertambah dalam sistem industri modern, *neural network* seperti yang diterapkan dalam kontrol berbasis *backpropagation* menawarkan pendekatan yang adaptif dan efektif untuk mengatasi tantangan tersebut, sembari menyediakan platform untuk pengembangan lebih lanjut dalam teknologi kontrol otomatis dan optimasi proses industri [23][24].

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381



Gambar 2. Topologi kendali NN. (a) Skema *indirect*. (b) Skema *direct*. (c) Skema *feedback*/*feedforward*

Kebanyakan penelitian di bidang terkait lebih menekankan pada aspek prediksi [25][26][27], klasifikasi [28][29][30], atau pengembangan algoritma pelatihan jaringan syaraf tiruan seperti *Levenberg–Marquardt* [3] dan *spatio-temporal backpropagation* [15][31][32], tanpa menerapkannya secara langsung sebagai pengendali berbasis eror pada sistem fisik. Beberapa studi memang membahas kontrol sistem dengan jaringan saraf tiruan [25][33][34], namun tidak mengintegrasikan skema pelatihan dua tahap (seri-paralel dan paralel), tidak menyertakan mekanisme *online tuning* berbasis *backpropagation* yang sederhana namun efektif, serta tidak mengevaluasi respon sistem terhadap gangguan menggunakan parameter performa transien standar seperti *rise time* atau *overshoot*. Penelitian ini mengisi gap tersebut dengan mengembangkan pengendali NN berbasis eror pada *Pressure Process Rig* 38-714, dilengkapi metode faktor pengali untuk menjaga kestabilan keluaran dalam batas kerja, dan menunjukkan performa yang lebih adaptif dan tangguh terhadap gangguan dibanding metode *Direct Inverse Control NN* [18][29].

2. METODE PENELITIAN

2.1. Pengolahan Data

Percobaan menggunakan sistem PPR sebagai representasi model sistem yang dibentuk menggunakan metode *neural network*. NN *plant* tersebut akan dipakai untuk melakukan simulasi terhadap NN pengendali bebasis eror. Data yang digunakan diperoleh dari percobaan PPR selama lima detik dengan *sampling time* 0.001 detik sehingga didapatkan data sebanyak lima ribu data. Data masukan dan keluaran yang digunakan adalah data yang telah dikonversi menjadi tegangan 0.4-2 volt. Salah satu kekurangan NN adalah sulitnya melakukan proses pembelajaran jika antara dimensi satu dan dimensi yang lain dalam satu data mempunyai *range* yang lebar. Sehingga, sebelum digunakan untuk proses pembelajaran, set data masukan dan keluaran diproses dalam dua tahap: *zscore* dan normalisasi.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381

Gambar 3. *Pressure Process Rig* 38-714

Metode *zscore* adalah metode untuk menyeragamkan antara data satu dengan data lain yang mempunyai range berbeda-beda agar mempunyai jangkauan dan rata-ratanya seragam. Metode *zscore* menghitung nilai rata-rata dan standar deviasi dari masing-masing dimensi data awal. Kemudian, data tersebut akan dikurangi dengan rata-ratanya agar nilai rata-rata seluruh dimensi menjadi nol. Setelah dikurangi data masih memiliki nilai jangkauan yang berbeda-beda. Oleh karena itu, data kemudian dibagi dengan standar deviasi agar jangkauan data menjadi lebih seragam. Secara umum persamaan untuk fungsi *zscore* adalah sebagai berikut:

$$Z = \frac{X - \bar{X}}{\sigma} \qquad (1)$$

Z = set data baru
x = set data lama
$\bar{X}$ = nilai rata rata data
σ = standar deviasi data

Langkah berikutnya adalah normalisasi data sehingga nilai maksimum data menjadi 0.8 dan nilai minimumnya menjadi minus 0.8. Pengecilan *range* data ini bertujuan untuk mempermudah dalam proses pembelajaran karena data dengan *range* yang kecil dan seragam lebih mudah diproses dibandingkan dengan data yang memiliki *range* yang besar. Secara umum persamaan untuk fungsi normalisasi data adalah sebagai berikut:

$$x_{baru} = (T_{max} - T_{min}) * \left(\frac{x - x_{min}}{x_{max} - x_{min}}\right) + T_{min} \qquad (2)$$

x_{baru} = set data baru
x = set data lama
x_{min} = data minimum dari set data lama
x_{max} = data maksimum dari set data lama
T_{min} = data minimum dari set data baru
T_{max} = data maksimum dari set data baru

Untuk pembelajaran NN pengendali berbasis eror, nilai eror diperoleh dengan menggunakan hasil kurang dari nilai keluaran referensi yang sekarang y(t) dengan nilai keluaran referensi yang sebelumnya y(t-1) seperti pada persamaan berikut

$$\Delta y(t) = y(t) - y(t-1) \qquad (3)$$

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381

Dengan Δy(t) sebagai nilai masukan, maka keluaran dari NN pengendali adalah nilai yang memiliki bobot yang sama dengan Δy(t), yaitu Δu(t)

$$\Delta u(t) = u(t) - u(t-1) \qquad (4)$$

2.2. Identifikasi Plant

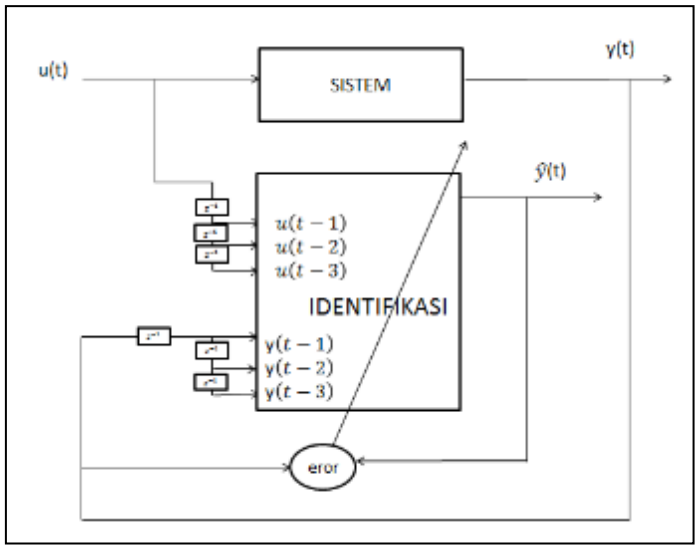
Proses pembelajaran identifikasi NN menggunakan konfigurasi seri-paralel (Gambar 4). Pada konfigurasi ini, nilai keluaran *plant* yang lalu digunakan dalam proses pembelajaran identifikasi. Karena *plant* dianggap stabil secara BIBO, semua sinyal yang digunakan dalam proses identifikasi juga ikut terikat. Penggunaan nilai keluaran yang sebenarnya juga menghilangkan error dalam pengukuran. Dengan pertimbangan diatas, konfigurasi seri-paralel menjadi konfigurasi yang sering dipakai dalam identifikasi sistem nonlinear. [8] Sistem nonlinear yang dalam percobaan ini menggunakan PPR dapat dimisalkan dalam persamaan sebagai berikut

$$\hat{y}[t] = f\big(y[t-1], \ldots, y[t-n_y], u[t-1], \ldots, u[t-n_u]\big) \qquad (5)$$

dengan yˆadalah keluaran dari NN, y adalah keluaran dari sistem, u adalah masukan dari sistem, dan n_y dan n_u adalah *lag*, atau *operator delay* bagi masing-masing keluaran dan masukan sistem. Sedangkan fungsi f adalah fungsi non-linear sistem. Persamaan 5 dapat disebut sebagai *non-linear mapping* dari ruang dengan jumlah dimensi n_y+ n_u ke dalam ruang satu dimensi. *Mapping* dapat diperkirakan dengan menggunakan NN dengan metode pembelajaran *backpropagation*.

Jumlah *operator delay* yang dipakai dalam identifikasi NN pada percobaan ini adalah tiga untuk n_y dan empat untuk n_u. NN hasil pembelajaran tersebut nantinya akan dipakai sebagai pengganti *plant* dalam proses *tuning* pengendali. Sistem lalu akan menggunakan data keluaran dari hasil percobaan sebagai nilai referensi untuk mengubah bobot NN. Proses pembelajaran identifikasi NN *plant* menggunakan konfigurasi seri-paralel (Gambar 4) dengan persamaan sebagai berikut

$$\hat{y}(t) = f[u(t-1), u(t-2), u(t-3), y(t-1), y(t-2), y(t-3)] \qquad (6)$$



Gambar 4. Skematik pembelajaran identifikasi *plant*

Parameter yang perlu diperhatikan pada pembelajaran adalah

Neuron input : 6 buah

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381

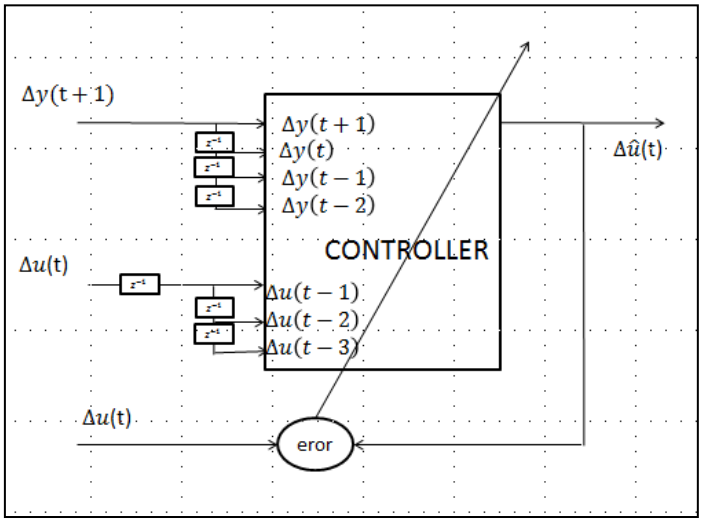
Neuron hidden : 4 buah
Neuron output : 1 buah
Alpha : 0.2
Momentum : 0

2.3. Pelatihan Pengendali

2.3.1. Tahap 1: Pelatihan Pengendali dengan Masukan Eror

Pengendali bersasis NN yang akan dibuat adalah pengendali yang bekerja dengan nilai masukan berupa eror dari sistem. Ketika pengendali di-*cascade* dengan NN *plant*, nilai keluaran dari sistem yang digabung akan sama dengan nilai *setpoint*-nya. Nonlinearitas dari pengendali akan menghilangkan nonlienaritas pada NN *plant*. Pembelajaran tahap pertama NN pengendali menggunakan konfigurasi seri paralel dengan persamaan masukan sebagai berikut:

$$\Delta\hat{u}(t) = f[\Delta y(t+1), \Delta y(t), \Delta y(t-1), \Delta y(t-2), \Delta u(t-1), \Delta u(t-2), \Delta u(t-3)] \quad (7)$$



Gambar 5. Skematik pembelajaran NN pengendali tahap 1

Parameter yang perlu diperhatikan pada pembelajaran adalah
Neuron input : 7 buah
Neuron hidden : 10 buah
Neuron output : 1 buah
Alpha : 0.2
Momentum : 0

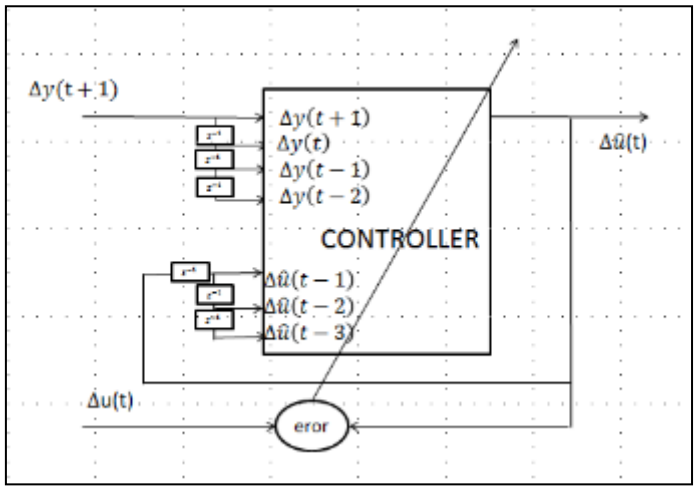
3.3.2. Tahap 2: Pelatihan Pengendali dengan Konfigurasi Paralel

Pembelajaran tahap kedua NN pengendali menggunakan konfigurasi paralel dengan persamaan sebagai berikut:

$$\Delta\hat{u}(t) = f[\Delta y(t+1), \Delta y(t), \Delta y(t-1), \Delta y(t-2), \Delta\hat{u}(t-1), \Delta\hat{u}(t-2), \Delta\hat{u}(t-3)] \quad (8)$$

Parameter yang perlu diperhatikan pada pembelajaran adalah
Neuron input : 7 buah
Neuron hidden : 10 buah
Neuron output : 1 buah
Alpha : 0.2
Momentum : 0.5

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381

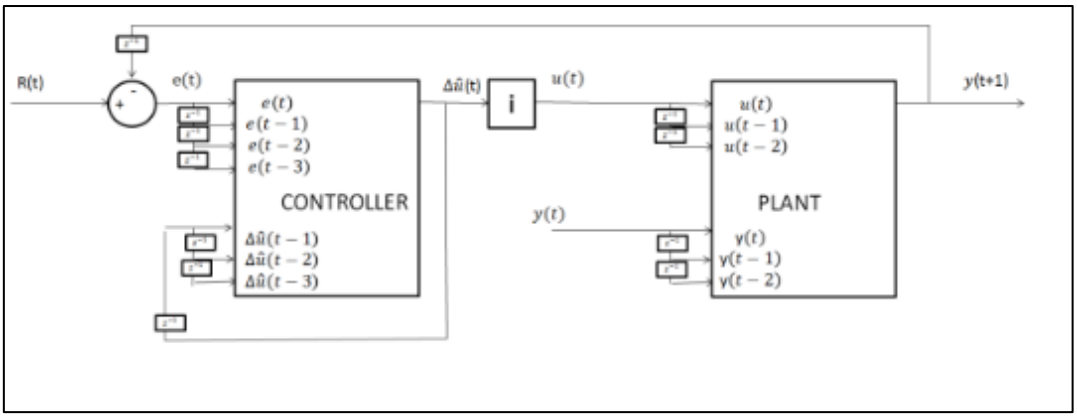


Gambar 6. Skematik pembelajaran NN pengendali tahap 2

2.4. Pengujian Pengendali

2.4.1. Pengujian tanpa Online Tuning

Pada percobaan ini, NN *plant* dan NN pengendali hasil pembelajaran akan di-*cascade* sesuai dengan skematik pada Gambar. NN pengendali diberi masukan $e(t)$ berupa hasil selisih dari nilai target $R(t)$ dan keluaran dari NN *plant* $y(t)$. Seperti pada tahap sebelumnya, nilai keluaran NN pengendali $\Delta\hat{u}(t)$ kemudian akan dijumlahkan dengan nilai sebelumnya $\hat{u}(t-1)$ untuk mendapatkan $\hat{u}(t)$ yang akan dipakai sebagai masukan terhadap NN *plant*. Nilai keluaran NN *plant* y(t+1) akan dipakai sebagai *feedback* untuk dikurangkan dengan R(t) untuk mendapatkan nilai eror e(t). y(t+1) juga dipakai untuk dibandingkan dengan nilai keluaran *plant* yang sebenarnya.



Gambar 7. Skematik NN pengendali berbasis eror

Pengujian kedua menggunakan data *step* dengan nilai terendah 0 dan nilai tertinggi 0.5, untuk memastikan bahwa NN pengendali mampu bekerja pada data yang memiliki nilai yang konstan. Data *step* yang digunakan hanya 200 data, tidak sebanyak data pembelajaran karena hanya dipakai untuk pengecekan. Spesifikasi respon transien yang akan diperhatikan adalah sebagai berikut

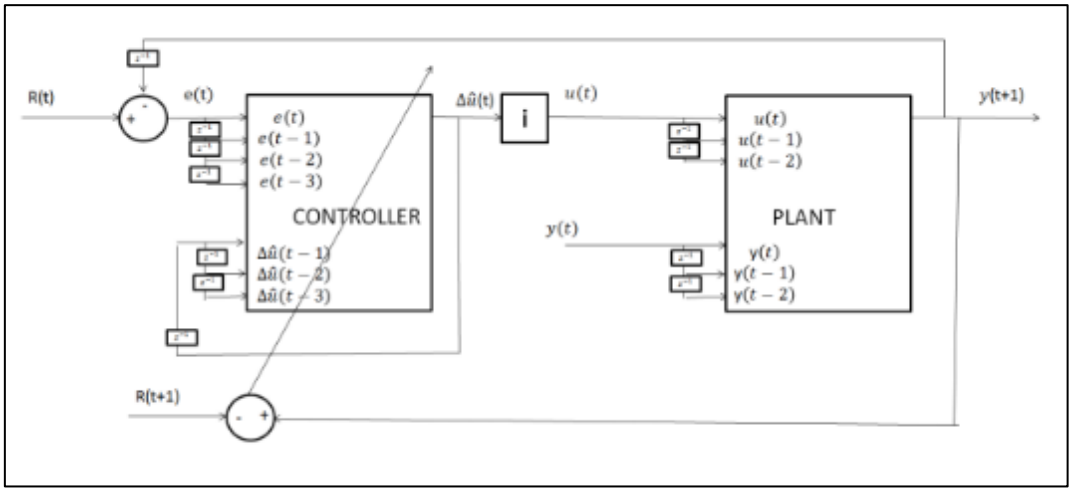
- *Delay time* (td) adalah waktu yang dibutuhkan oleh respon untuk mencapai setengah dari nilai akhir
- *Rise time* (tr) adalah waktu yang dibutuhkan oleh respon untuk naik dari 0% sampai 90%

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381

- *Peak time* (tp) adalah waktu yang dibutuhkan oleh respon untuk mencapai puncak pertama *overshoot*
- *Maximum overshoot* (Mp) adalah nilai maksimum dari kurva respon jika dibandingkan dengan nilai referensi
- *Settling time* (ts) adalah waktu yang dibutuhkan oleh respon untuk mencapai dan dengan stabil berada di atau mendekati nilai akhir (biasanya memiliki toleransi sebesar 2%)

2.4.2. Pengujian dengan Online Tuning

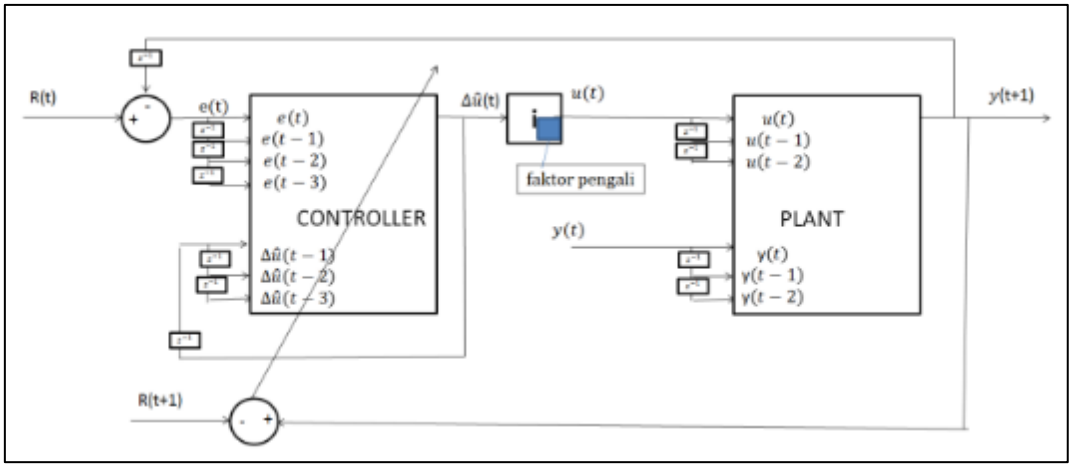
Untuk mengantisipasi adanya gangguan dari luar dan untuk memperbagus nilai keluaran dari *plant*, NN pengendali akan diberikan *tuning* secara *online*, dimana bobot NN pengendali akan selalu diubah disetiap iterasi pengolahan data (Gambar 8). Untuk mendapatkan hasil yang paling bagus, dilakukan percobaan variasi terhadap nilai *alpha* dan momentum.



Gambar 8. Skematik NN pengendali dengan *tuning* secara *online*

2.4.3. Menggunakan Metode Faktor Pengali

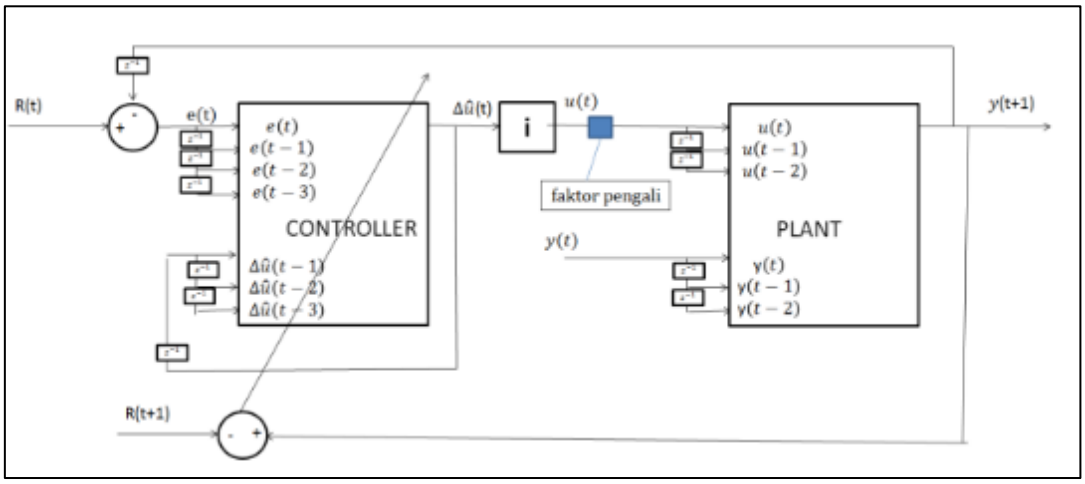
Percobaan berikutnya adalah dengan menggunakan metode faktor pengali kepada $\Delta\hat{u}(t)$ sebelum digunakan sebagai masukan ke NN *plant* (Gambar 9) dengan harapan nilai $\hat{u}(t)$ yang dipakai berada pada nilai batas kerja. Pada percobaan pertama, faktor pengali yang dipakai adalah 0.5,



Gambar 9. Skematik NN pengendali *online tuning* dengan metode faktor pengali

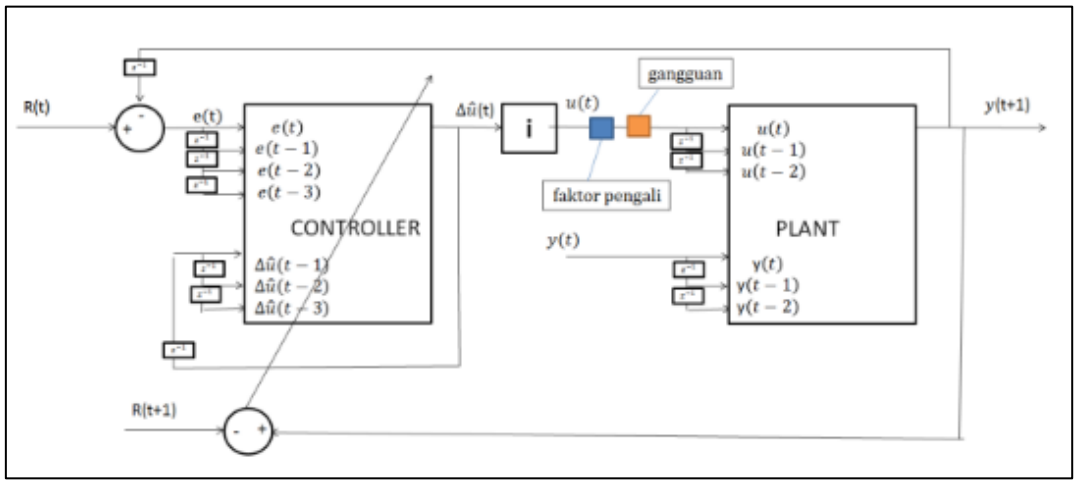
e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381

Percobaan berikutnya adalah memberikan faktor pengali di $\hat{u}(t)$ sebelum dipakai sebagai masukan ke NN *plant* (Gambar 10). Berbeda dengan percobaan sebelumnya, pada percobaan ini nilai $\hat{u}(t)$ di setiap iterasi akan dikalikan dengan faktor pengali. Dengan nilai faktor pengali yang sama, dilakukan pengujian berikutnya dengan data *step*.



Gambar 10. Skematik NN pengendali *online tuning* dengan metode faktor pengali

Pengujian berikutnya adalah dengan memberi faktor gangguan pada $\hat{u}(t)$ (Gambar 11) untuk melihat apakah NN pengendali mampu mengatasi gangguan yang diberikan dan meredam eror. Gangguan yang diberikan berupa paduan data *step* dan data *random* dengan *range* nilai -0.2 sampai 0.2. Faktor pengali yang digunakan sama dengan yang sebelumnya, 0.7.



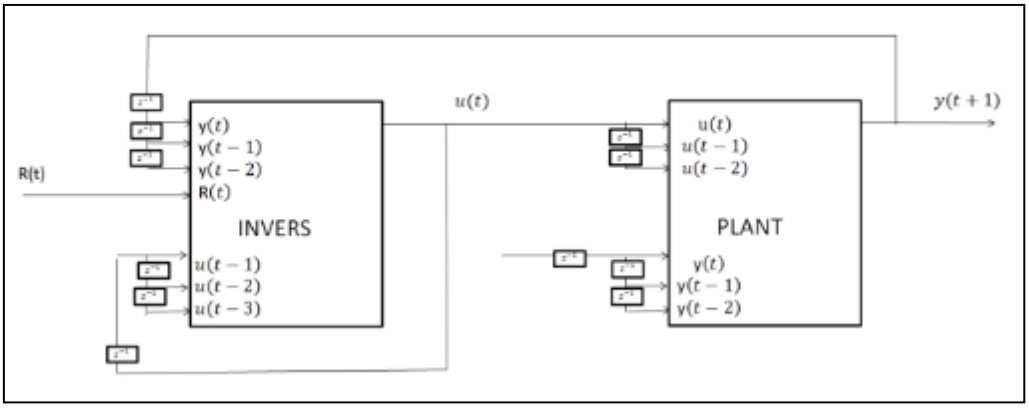
Gambar11. Skematik NN pengendali *online tuning* dengan gangguan

Pengujian berkikutnya adalah pengujian dengan masukan *step* yang diberi gangguan, namun dengan gangguan yang berbeda dengan gangguan yang diberikan ketika diuji dengan data pembelajaran. Gangguan yang diberikan adalah gangguan *step* dengan nilai gangguannya adalah 0.2 atau 40% dari nilai referensi masukannya.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381

2.5. Perbandingan dengan Pengendali DIC NN

Pada percobaan ini akan dilakukan perbandingan performa dari NN pengendali berbasis eror dengan pengendali *feedback* yang lain. Pada percobaan ini pengendali yang digunakan sebagai perbandingan adalah *Direct Inverse Control* (DIC). DIC yang dipakai adalah hasil dari pembelajaran NN dengan metode *backpropagation*. *Direct Inverse Control* (DIC) adalah pengendali dengan konsep yang paling sederhana dan fundamental pada pengendali berbasis NN. Pengendali *inverse* menghasilkan *inverse* dari *plant*. Nilai keluaran *plant* digunakan sebagai nilai masukan kepada pengendali. Keluaran NN dibandingkan dengan sinyal pembelajaran (nilai masukan *plant*) dan nilai erornya dipakai untuk melatih NN. [9]



Gambar 12. Skematik *Direct Inverse Control Neural Network*

2.5.1. Pengujian DIC NN tanpa Gangguan

Pengujian menggunakan data pembelajaran dan data step.

2.5.2. Pengujian DIC NN dengan Gangguan

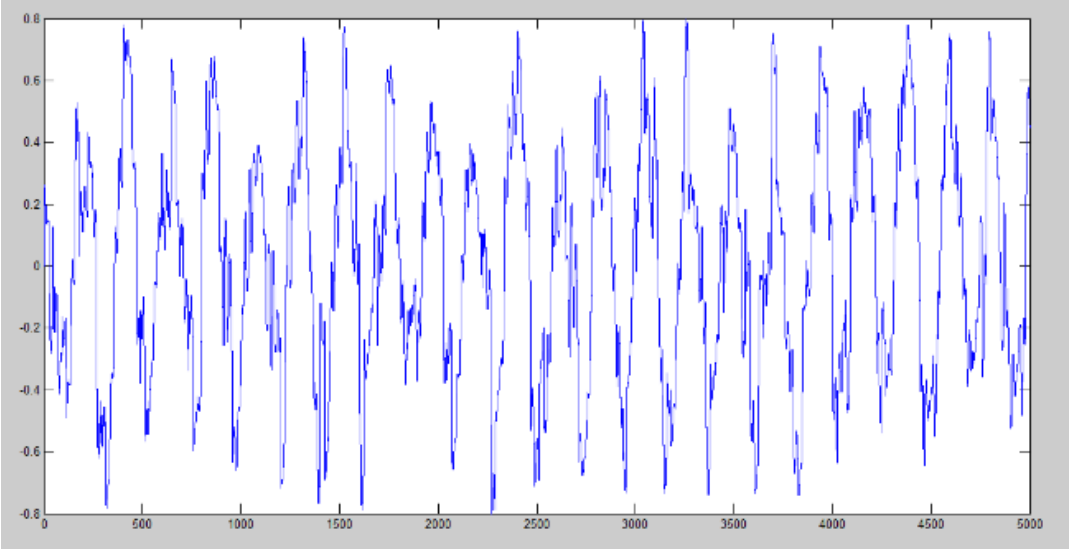
Pengujian berikutnya adalah dengan memberikan gangguan pada DIC NN untuk melihat performa dari NN pengendali tersebut. Gangguan yang diberikan dibatasi dengan *range* data -0.2 sampai 0.2. Gangguan yang diberikan adalah variasi dari *step* dan *random*. Pengujian awal menggunakan data pembelajaran, dimana gangguan yang diberikan adalah gabungan dari data *step* dan data *random*. Pada pengujian ini, nilai *alpha* dan momentum disamakan dengan pengendali NN berbasis eror, dengan *alpha* 0.2 dan momentum 0. Pengujian berikutnya adalah dengan memberikan gangguan kepada sistem. Gangguan yang digunakan adalah data *step* dengan nilai terendah 0 dan nilai tertinggi 0.5, untuk memastikan bahwa NN pengendali mampu bekerja pada data yang memiliki nilai yang konstan.

3. HASIL DAN PEMBAHASAN

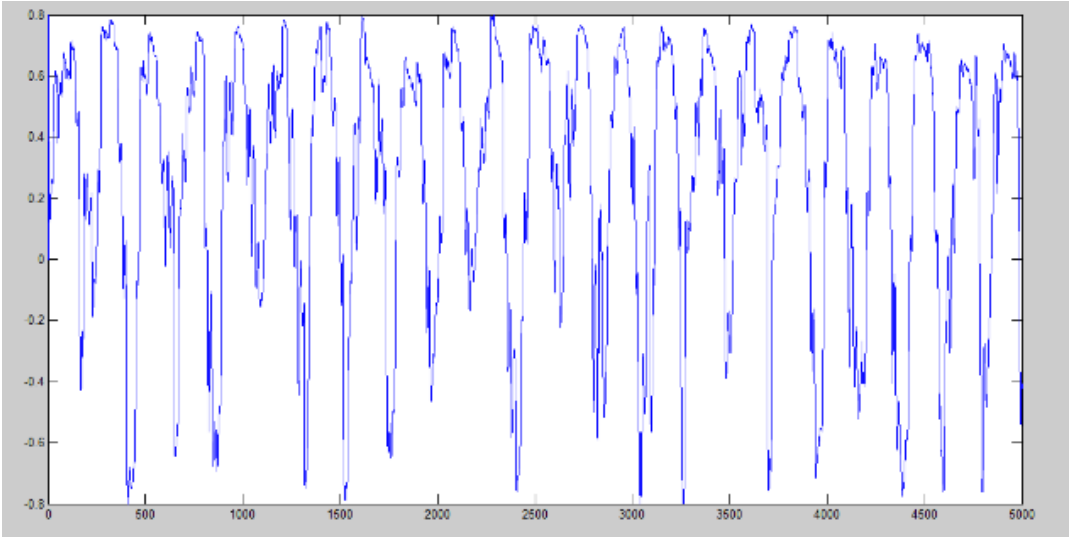
3.1. Pengolahan Data

Setelah mendapatkan 5000 data dari Pressure Process Rig, data tersebut kemudian diproses dengan metode zscore dan normalisasi. Hasil pengolahan data yang akan dipakai di percobaan ini ditampilkan pada Gambar 13 dan Gambar 14 berikut.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381



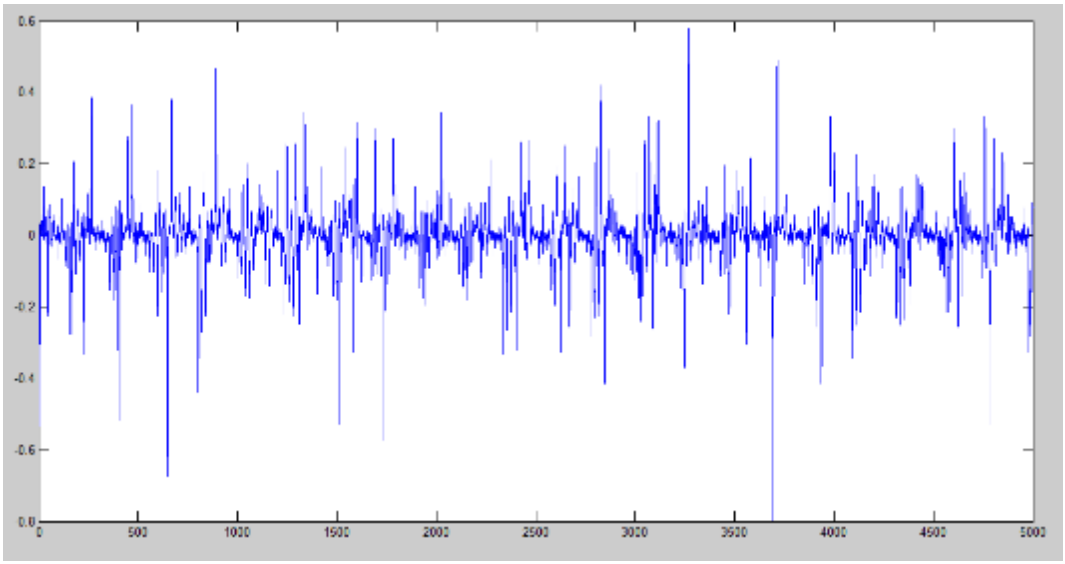
Gambar 13. Plot data masukan setelah dinormalisasi



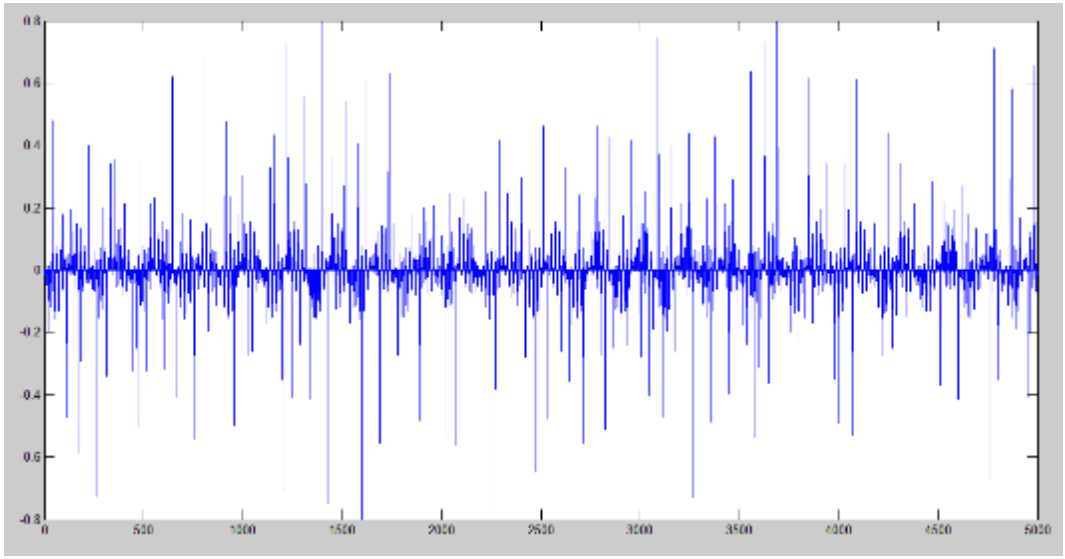
Gambar 14. Plot data keluaran setelah dinormalisasi

Sebelum dipakai untuk pembelajaran, data diproses dengan metode faktor pengali untuk mengatur nilai maksimum dan nilai minimum data menjadi 0.8 dan -0.8. Pengolahan $\Delta y(t)$ menggunakan metode faktor pengali karena perhitungannya lebih cepat dibandingkan dengan metode normalisasi dan metode faktor pengali juga dapat diaplikasikan ketika pengujian secara online. Nilai terbesar $\Delta y(t)$ adalah 0.3212 dan terkecilnya adalah -0.4455, sehingga faktor pengali yang dipakai adalah 0.8/0.4455. $\Delta y(t)$. Seperti $\Delta y(t)$, $\Delta u(t)$ juga akan diproses dengan faktor pengali. Nilai terbesar $\Delta u(t)$ adalah 0.6808 dan terkecilnya adalah -0.6839, sehingga faktor pengali yang dipakai adalah 0.8/0.6839. $\Delta u(t)$.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381



Gambar 15. Plot data Δy(t) setelah menggunakan faktor pengali

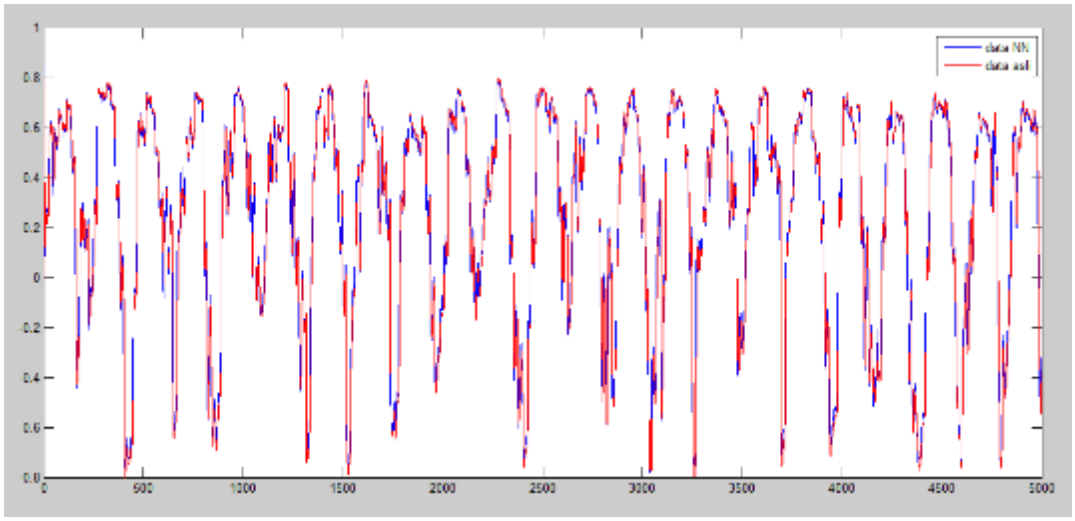


Gambar 16. Plot data Δu(t) setelah menggunakan faktor pengali

3.2. Identifikasi Plant

Pembelajaran dilakukan sebanyak 7415 epoh dengan eror rata-rata pembelajaran adalah 2.24E-04. NN *plant* hasil pembelajaran kemudian dibandingkan hasilnya dengan data keluaran sistem y(t). Dari Gambar 17, terlihat bahwa nilai keluaran NN mampu mengikuti data keluaran sistem dengan cukup baik.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381

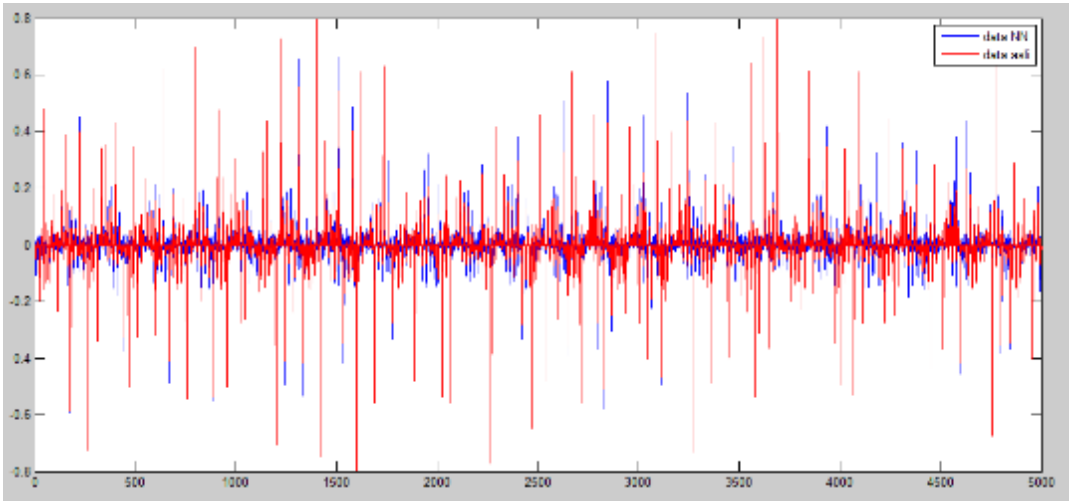


Gambar 17. Plot perbandingan keluaran NN *plant* dengan data keluaran sistem y(t)

3.3. Pelatihan Pengendali

3.3.1. Tahap 1: Pelatihan Pengendali dengan Masukan Eror

Pembelajaran dilakukan sebanyak 10000 epoh dengan eror rata-rata pembelajaran adalah 9.0815E-04. NN pengendali hasil pembelajaran kemudian dibandingkan hasilnya dengan nilai referensi Δu(t). Dari Gambar 18, terlihat bahwa nilai keluaran NN pengendali masih belum mampu mengikuti nilai referensi dengan sepenuhnya dimana pada beberapa titik khususnya ketika terjadi spike, masih banyak yang belum serupa.

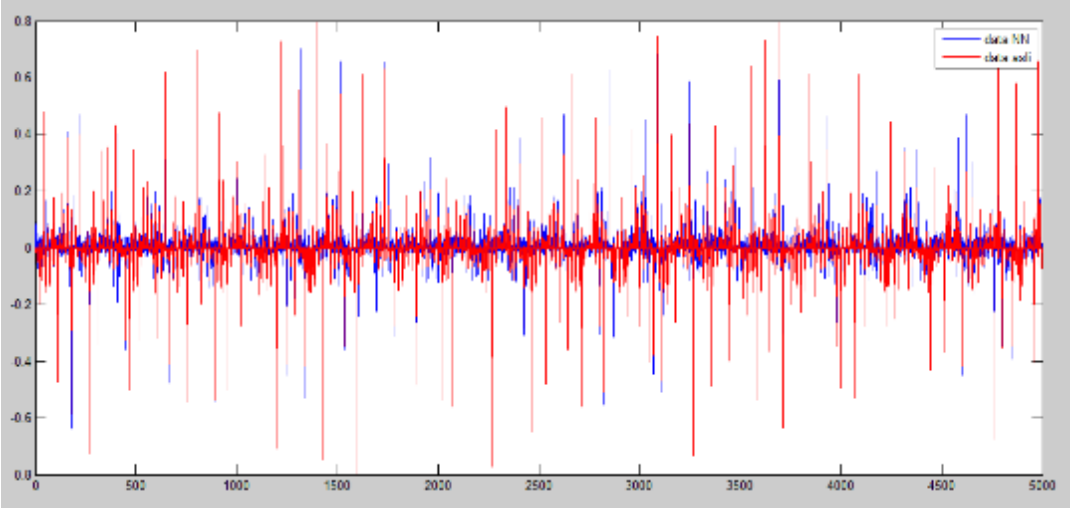


Gambar 18. Plot perbandingan keluaran NN pengendali dengan $\Delta u(t)$

3.3.2. Tahap 2: Pelatihan Pengendali dengan Konfigurasi Paralel

Pembelajaran dilakukan sebanyak 10000 epoh dengan eror rata-rata pembelajaran adalah 0.001. NN pengendali hasil pembelajaran kemudian dibandingkan hasilnya dengan nilai referensi. Dari Gambar 19, terlihat bahwa nilai keluaran NN pengendali walaupun masih belum sepenuhnya mampu mengikuti nilai referensi Δu(t), tapi sudah cukup untuk bisa dipakai sebagai pengendali.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381

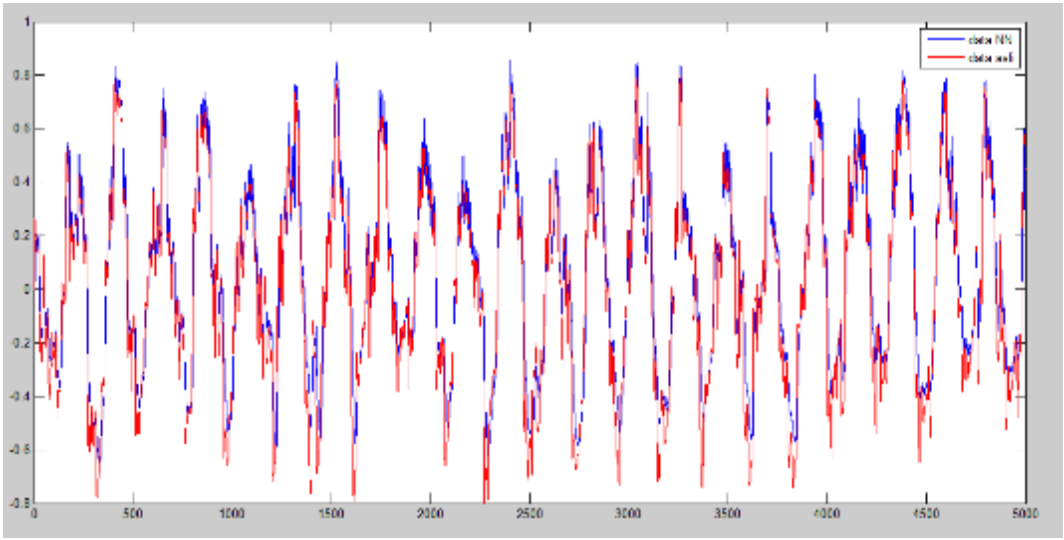


Gambar 19. Plot perbandingan keluaran NN pengendali dengan $\Delta u(t)$

3.4. Pengujian Pengendali

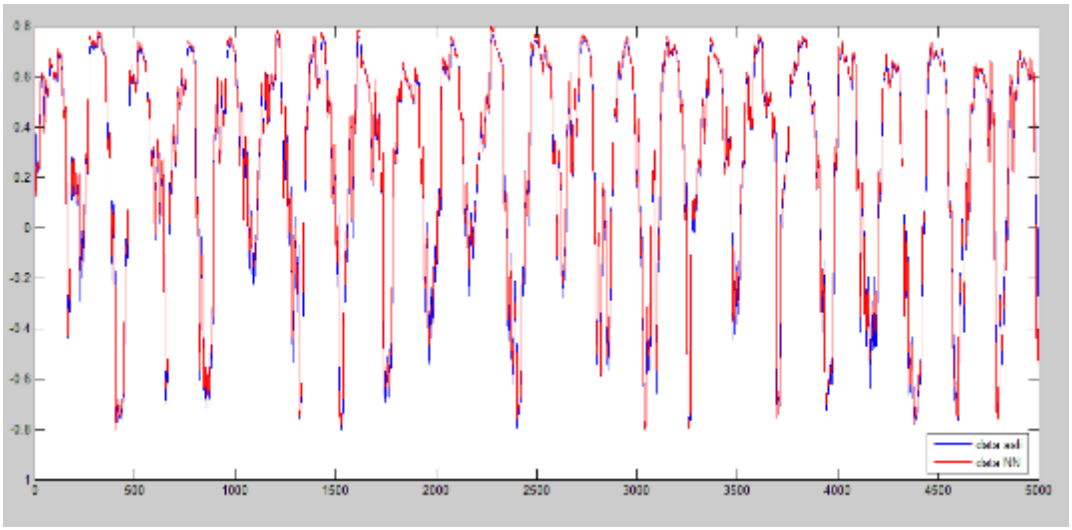
3.4.1. Pengujian tanpa Online Tuning

Pengujian pertama menggunakan data yang dipakai dalam pembelajaran. Pada Gambar 21, terlihat bahwa nilai keluaran NN *plant* sudah mirip dengan nilai keluaran *plant* dengan eror rata-rata 8.787E-04. Pada percobaan ini didapatkan hasil yang bagus, hal ini dikarenakan nilai $e(t)$ yang akan berubah sesuai dengan nilai keluaran NN *plant* $y(t+1)$.



Gambar 20. Plot perbandingan keluaran NN pengendali $\hat{u}(t)$ dengan data masukan sistem $u(t)$ pengujian tanpa *online tuning* dengan data pembelajaran

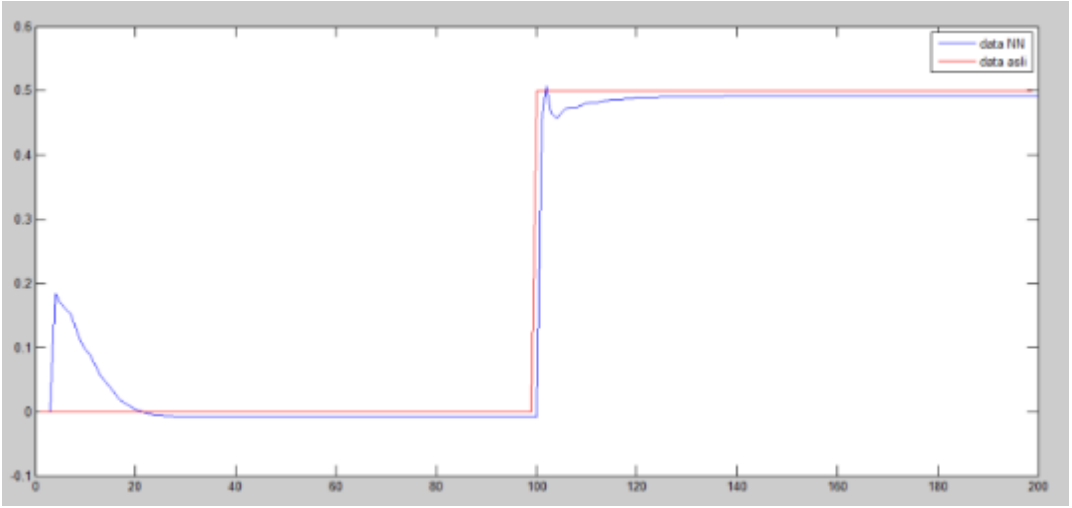
e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381



Gambar 21. Plot perbandingan keluaran NN *plant* dengan data keluaran sistem $\boldsymbol{y(t+1)}$ pengujian tanpa *online tuning* dengan data pembelajaran

Untuk pengujian kedua, pada Gambar 22, terlihat bahwa keluaran NN pengendali perlu 20 iterasi untuk dapat menyesuaikan dengan data referensinya. Ketika nilai masukan menjadi 0.5(data ke 101), pengendali NN terlihat berusaha untuk mengikutinya, dan kembali stabil lagi pada data ke 120. Spesifikasi respon transien untuk percobaan ini adalah

- td = 0.5 detik
- tr = 0.9 detik
- tp = 2 detik
- Mp = 1.3%
- ts = 20



Gambar 22. Plot perbandingan keluaran NN *plant* dengan data keluaran sistem $\boldsymbol{y(t+1)}$ pengujian tanpa *online tuning* dengan data

3.4.2. Pengujian dengan Online Tuning

Untuk mendapatkan hasil yang paling bagus, dilakukan percobaan variasi terhadap nilai *alpha* dan momentum (Tabel 1). Pada Tabel, terlihat bahwa nilai eror yang paling kecil (eror =

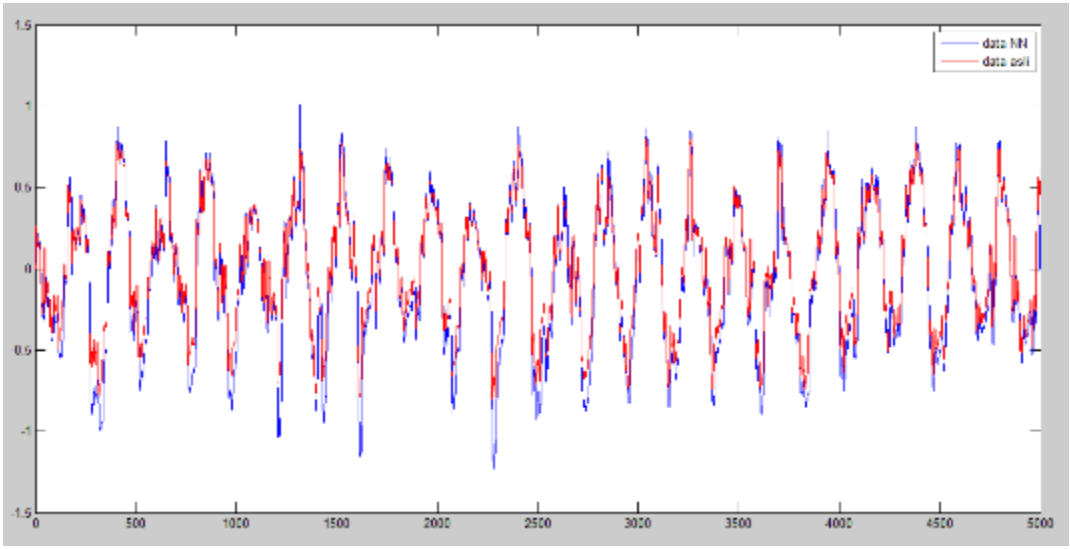
e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381

4.42E-04) didapatkan dengan alpha = 0.2 dan momentum = 0. Perbandingan nilai keluaran dari NN pengendali $\hat{u}(t)$ dengan data masukan sistem $u(t)$ ditampilkan pada Gambar 23, sedangkan perbandingan dari nilai keluaran NN *plant* dengan data keluaran sistem ditampilkan pada Gambar 24. Pada Gambar 23, terlihat bahwa nilai $\hat{u}(t)$ keluar dari nilai batas kerja yang ditetapkan (pada percobaan ini nilai batas kerja diasumsikan berkisar antara satu dan minus satu). Hal ini akan berdampak buruk jika diterapkan pada *plant* yang sebenarnya, sehingga nilai $\hat{u}(t)$ harus diubah sedemikian rupa sehingga tidak keluar dari nilai batas kerja.

Tabel 1. Percobaan pembelajaran NN pengendali secara *online* dengan variasi *alpha* dan momentum

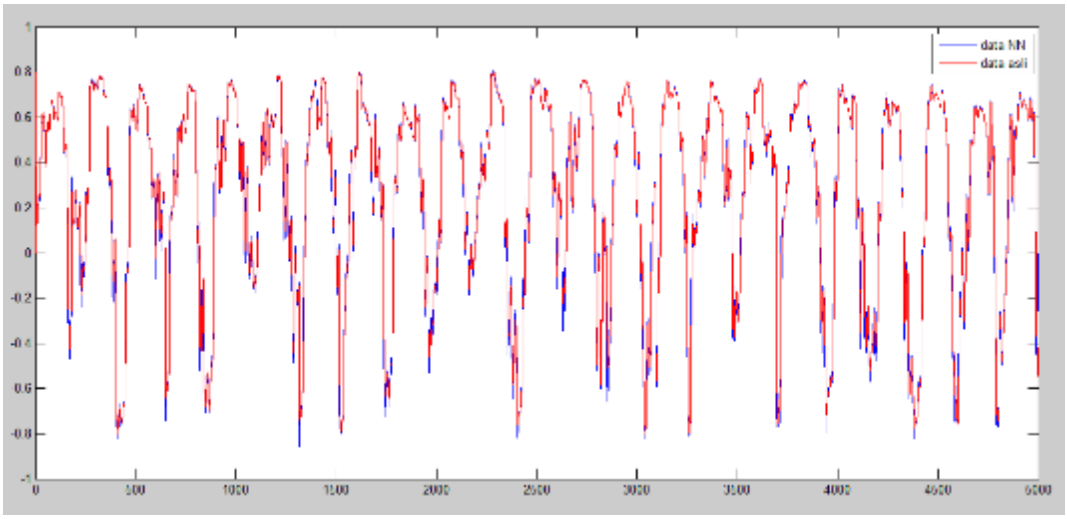
ALPHA	EROR RATA-RATA
0	8.79E-04
0.1	4.44E-04
0.2	**4.42E-04**
0.3	4.52E-04
0.4	4.59E-04
0.5	4.66E-04
0.6	4.80E-04
0.7	5.13E-04
0.8	5.68E-04
0.9	6.02E-04

MOMENTUM	EROR RATA-RATA
0	4.42E-04
0.1	4.46E-04
0.2	4.54E-04
0.3	4.62E-04
0.4	4.64E-04
0.5	4.74E-04
0.6	4.82E-04
0.7	5.48E-04
0.8	8.04E-04
0.9	2.30E-03



Gambar 23. Plot perbandingan keluaran NN pengendali $\hat{\boldsymbol{u}}(\boldsymbol{t})$ dengan data masukan sistem $\boldsymbol{u}(\boldsymbol{t})$ pengujian dengan *online tuning* dengan data pembelajaran

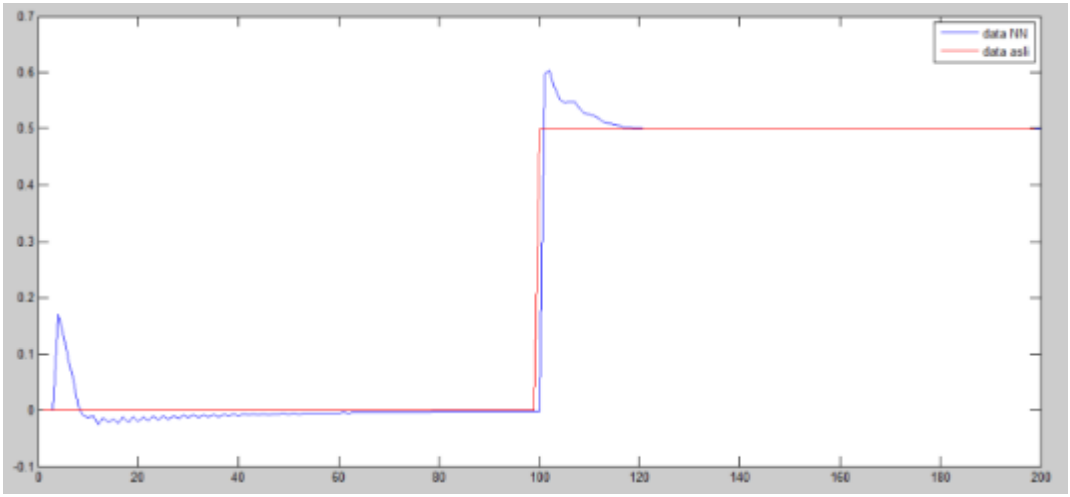
e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381



Gambar 24. Plot perbandingan keluaran NN *plant* dengan data keluaran sistem $\boldsymbol{y(t+1)}$ pengujian dengan *online tuning* dengan data pembelajaran

Dengan nilai *alpha* dan momentum yang sama, dilakukan pengujian berikutnya dengan data *step*. Pada Gambar 25, terlihat bahwa keluaran NN pengendali perlu 10 iterasi untuk dapat menyesuaikan dengan data referensinya. Ketika nilai masukan menjadi 0.5(data ke 101), pengendali NN terlihat berusaha untuk mengikutinya, dan kembali stabil lagi pada data ke 120. *Overshoot* berada pada t = 102 dengan nilai *overshoot* = 0.6042. Spesifikasi respon transien untuk percobaan ini adalah

- td = 0.3 detik
- tr = 0.6 detik
- tp = 2 detik
- Mp = 20.84%
- ts = 20 detik

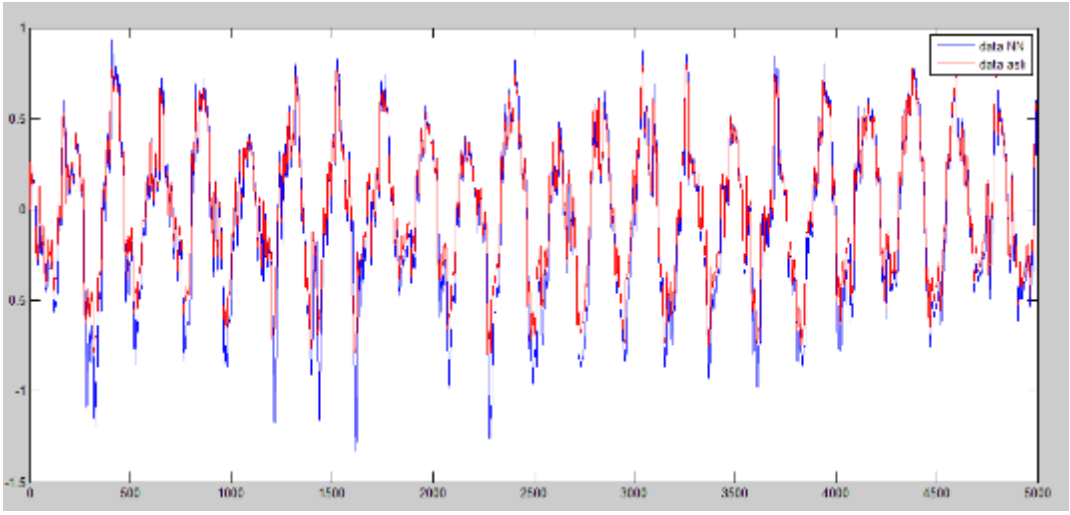


Gambar 25. Plot perbandingan keluaran NN *plant* dengan data keluaran sistem $\boldsymbol{y(t+1)}$ pengujian dengan *online tuning* dengan data *step*

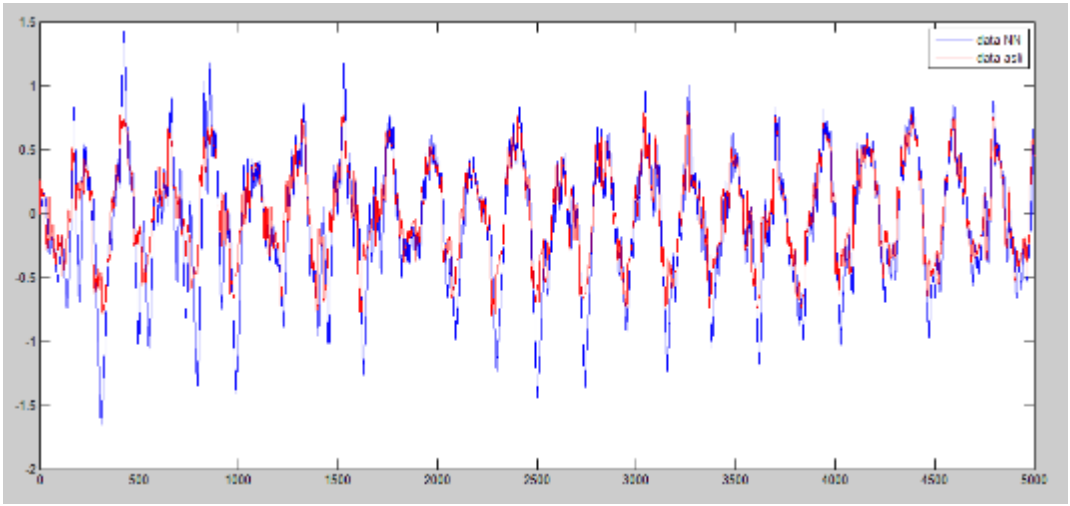
e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381

3.4.3. Menggunakan Metode Faktor Pengali

Nilai $\hat{u}(t)$ setelah diberi faktor pengali dibandingkan dengan $u(t)$ pada Gambar 26, dan seperti yang terlihat, tidak berbeda dengan percobaan yang sebelumnya, nilai $\hat{u}(t)$ masih belum berada dalam nilai batas kerja NN *plant*. Faktor pengali kemudian diturunkan sampai dengan 0.1, dengan nilai eror rata-rata keluaran NN *plant* adalah 0.0044. Dapat dilihat bahwa semakin kecil faktor pengali maka eror pada NN *plant* akan semakin besar, dan juga nilai keluaran NN pengendali semakin buruk dibandingkan dengan nilai masukan *plant*nya(Gambar 27).



Gambar 26. Plot perbandingan keluaran NN pengendali $\hat{u}(t)$ dengan data masukan sistem $u(t)$ pengujian menggunakan faktor pengali 0.5 dengan data pembelajaran

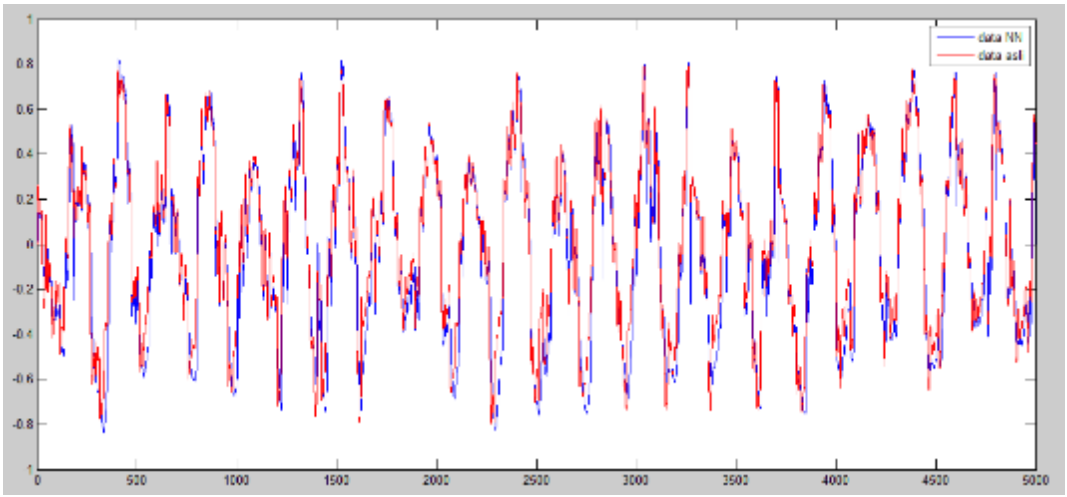


Gambar 27. Plot perbandingan keluaran NN pengendali $\hat{u}(t)$ dengan data masukan sistem $u(t)$ pengujian menggunakan faktor pengali 0.1 dengan data pembelajaran

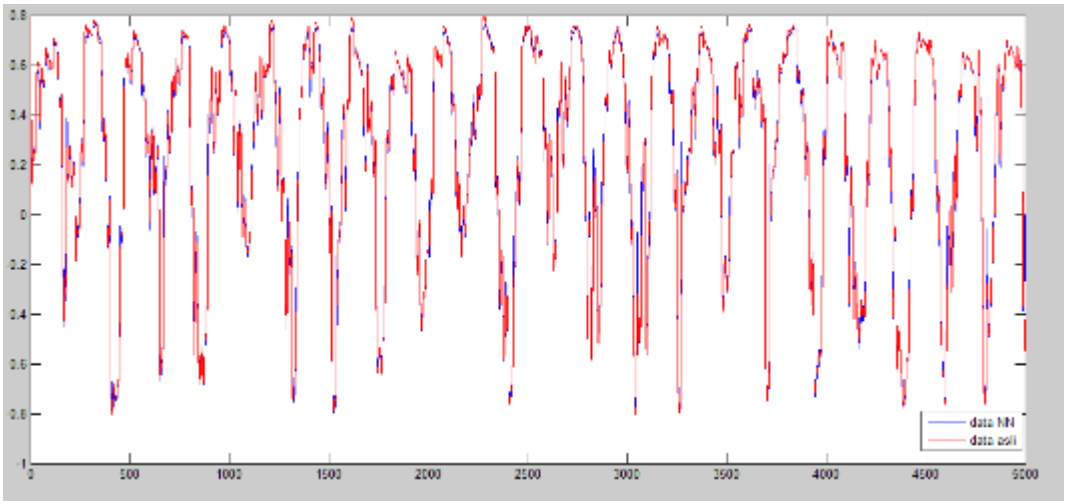
Setelah dilakukan percobaan variasi nilai faktor pengali, didapatkan bahwa dengan nilai 0.7 akan mendapatkan hasil yang baik dan mengembalikan nilai keluaran NN pengendali dalam *range* -1 dan 1. Nilai eror rata rata keluaran NN *plant* adalah 7.7161E-04. Nilai eror hasil percobaan ini memang tidak sebagus nilai eror sebelum menggunakan metode faktor pengali (4.02E-04), namun masih bisa ditolerir. Pada perbandingan antara keluaran NN pengendali $\hat{u}(t)$

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381

dan data masukan sistem $u(t)$ (Gambar 28) terlihat bahwa $\hat{u}(t)$ tidak memiliki nilai yang berada di luar nilai batas kerja.



Gambar 28. Plot perbandingan keluaran NN pengendali $\hat{\boldsymbol{u}}(\boldsymbol{t})$ dengan data masukan sistem $\boldsymbol{u}(\boldsymbol{t})$ pengujian menggunakan faktor pengali 0.7 dengan data pembelajaran

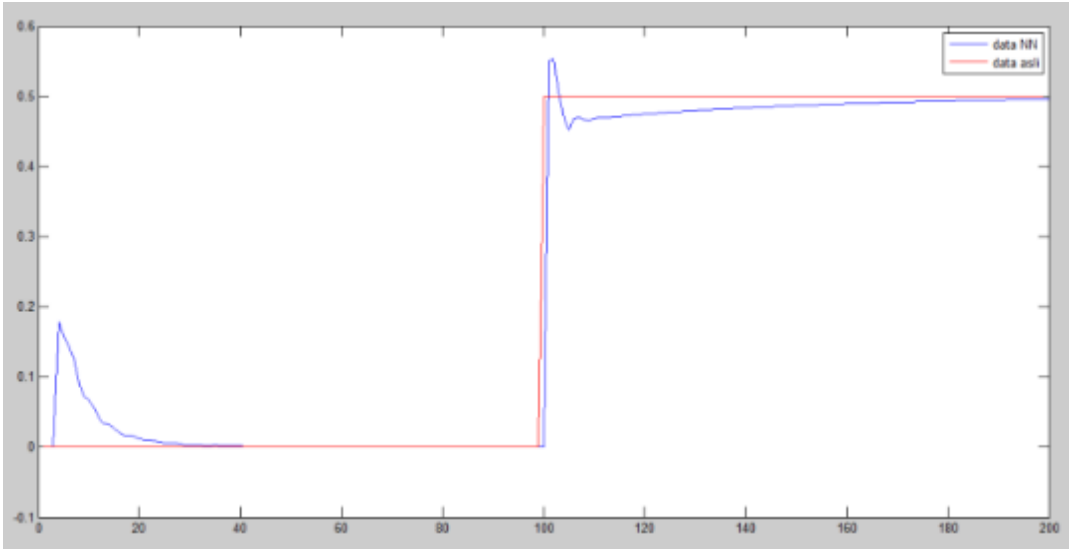


Gambar 29. Plot perbandingan keluaran NN *plant* dengan data keluaran sistem $\boldsymbol{y}(\boldsymbol{t}+\mathbf{1})$ pengujian menggunakan faktor pengali 0.7 dengan data pembelajaran

Dalam pengujian menggunakan data *step*, pada Gambar 30, terlihat bahwa keluaran NN pengendali perlu 20 iterasi untuk dapat menyesuaikan dengan data referensinya. Ketika nilai masukan menjadi 0.5(data ke 101), pengendali NN terlihat berusaha untuk mengikutinya, dan kembali stabil lagi pada data ke 140. *Overshoot* berada pada t = 102 dengan nilai *overshoot* = 0.5523. Spesifikasi respon transien untuk percobaan ini adalah

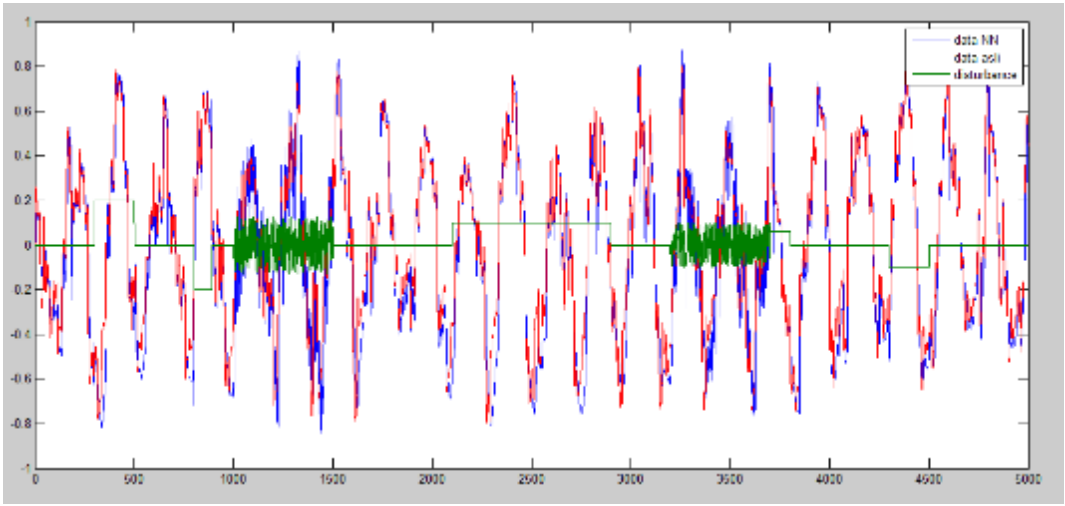
- td = 0.3 detik
- tr = 0.7 detik
- tp = 2 detik
- Mp = 10.46%
- ts = 40

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381



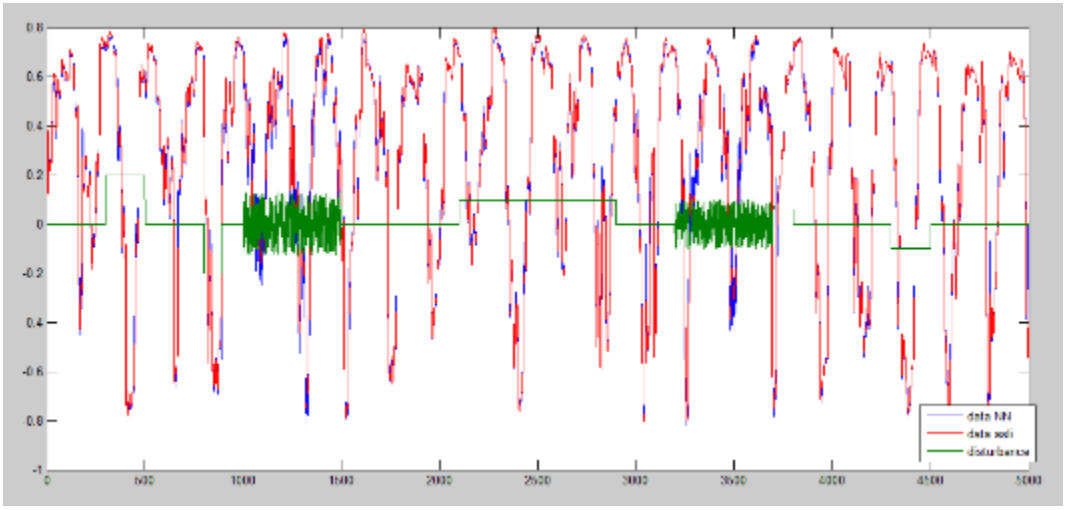
Gambar 30. Plot perbandingan keluaran NN *plant* dengan data keluaran sistem $\boldsymbol{y(t+1)}$ pengujian menggunakan faktor pengali 0.7 dengan data *step*

Pada pengujian dengan memberi faktor gangguan pada $\hat{u}(t)$, eror nilai keluaran NN *plant* adalah 0.001. Plot gangguan beserta pebandingan nilai keluaran NN pengendali $\hat{u}(t)$ dengan data masukan sistem $u(t)$ ditunjukkan pada Gambar 31, sedangkan perbandingan nilai keluaran NN *plant* dengan data keluaran sistem ditunjukkan pada Gambar 32. Pada Gambar 31 dan 32 terlihat bahwa nilai keluaran NN tidak ada yang keluar dari nilai batas kerja dan mampu mengikuti nilai referensinya dengan baik. Ketika gangguan yang diberikan adalah data *random*, terlihat bahwa respon data keluaran NN pada Gambar 31 dan 32 mengalami gejolak, namun masih dapat kembali mengikuti data referensinya. Sehingga dapat disimpulkan bahwa NN pengendali berbasis eror dengan *online tuning* dapat mengatasi faktor gangguan dengan range 0.2 sampai -0.2.



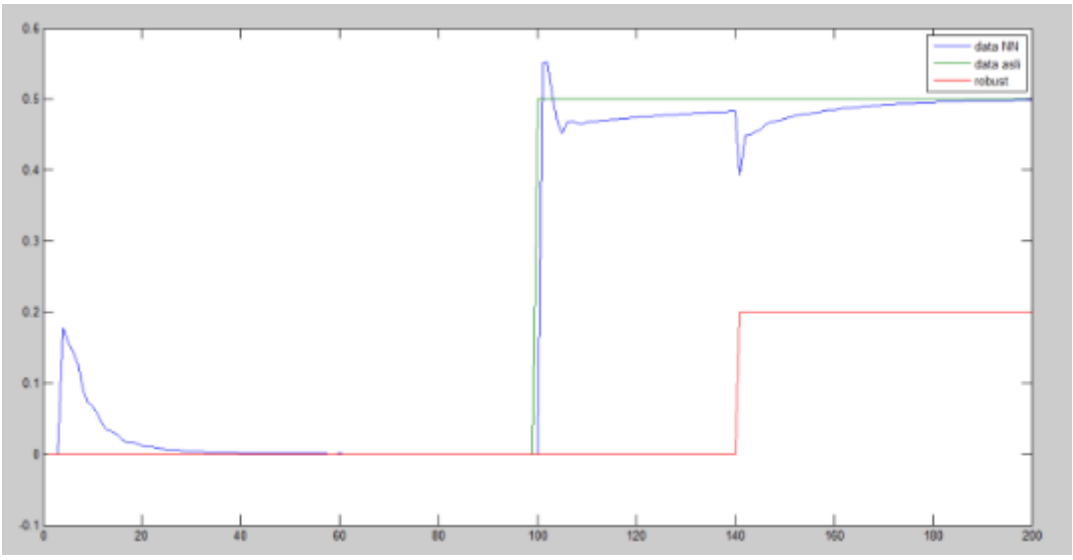
Gambar 31. Plot perbandingan keluaran NN pengendali $\boldsymbol{\hat{u}(t)}$ dengan data masukan sistem $\boldsymbol{u(t)}$ pengujian menggunakan faktor pengali dengan data pembelajaran ketika diberi gangguan

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381



Gambar 32. Plot perbandingan keluaran NN *plant* dengan data keluaran sistem $\boldsymbol{y(t+1)}$ pengujian menggunakan faktor pengali dengan data pembelajaran ketika diberi gangguan

Pada pengujian dengan masukan *step* yang diberi gangguan (Gambar 33), ketika gangguan mulai berpengaruh pada NN, nilai keluaran NN turun sampai 0.3933 atau 21.34% dari nilai referensi masukannya. Setelah itu, nilai keluaran NN mulai bergerak menuju nilai referensi masukannya.



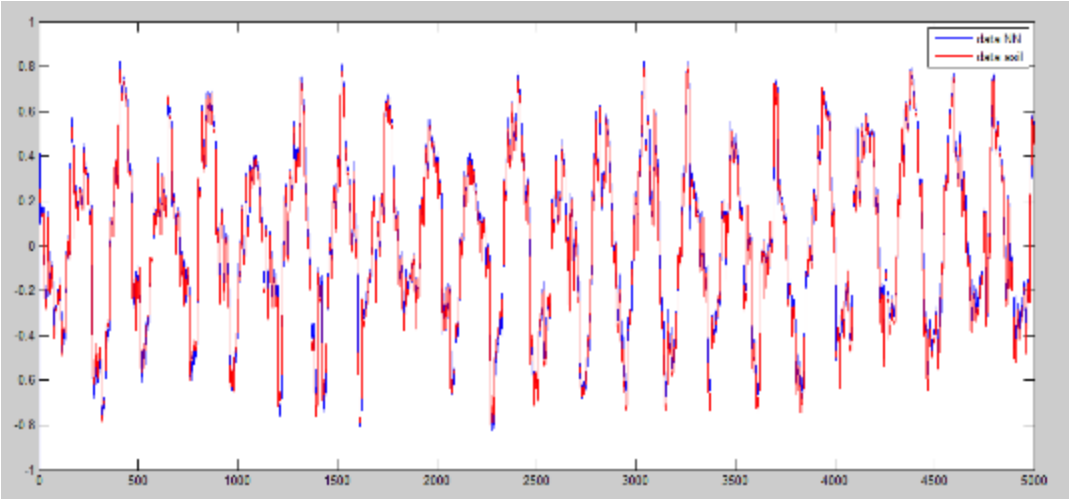
Gambar 33. Plot perbandingan keluaran NN *plant* dengan data keluaran sistem $\boldsymbol{y(t+1)}$ pengujian menggunakan faktor pengali dengan data *step* ketika diberi gangguan

3.5. Perbandingan dengan Pengendali DIC NN

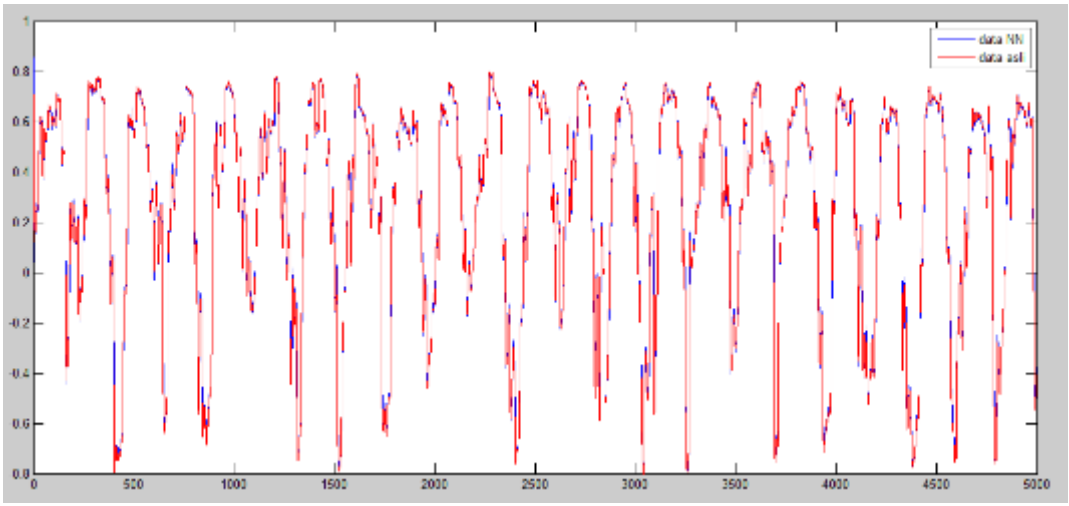
3.5.1. Pengujian DIC NN tanpa Gangguan

Pengujian pertama menggunakan data pembelajaran, hasil pengujian terlihat pada Gambar 34 dan Gambar 35, dengan nilai eror rata-rata keluaran NN *plant* 4.6721E-04. Pada Gambar 34 terlihat bahwa keluaran NN *plant* mampu mengikuti nilai referensinya dengan baik.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381



Gambar 34. Plot perbandingan keluaran NN inverse $\widehat{\boldsymbol{u}}(\boldsymbol{t})$ dengan data masukan sistem $\boldsymbol{u}(\boldsymbol{t})$ pengujian DIC NN menggunakan data pembelajaran



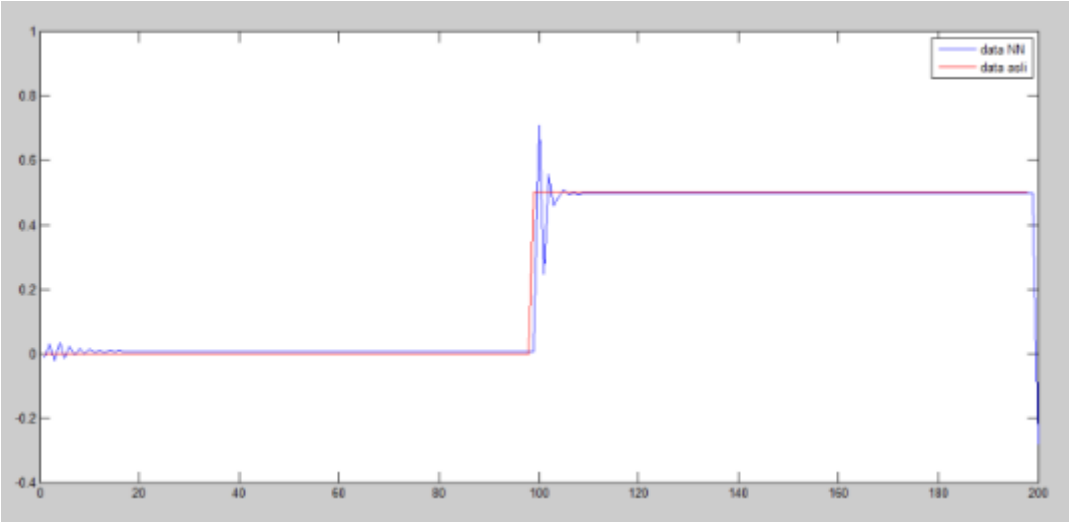
Gambar 35. Plot perbandingan keluaran NN *plant* $\boldsymbol{y}(\boldsymbol{t})$ dengan data keluaran sistem $\boldsymbol{y}(\boldsymbol{t})$ pengujian DIC NN menggunakan data pembelajaran *plant*

Pada pengujian berikutnya dengan data *step* (Gambar 36), ketika nilai masukan menjadi 0.5(data ke 101), pengendali NN terlihat berusaha untuk mengikutinya, dan kembali stabil lagi pada data ke 107. *Overshoot* berada pada t = 101 dengan nilai *overshoot* = 0.7112.

Spesifikasi respon transien untuk percobaan ini adalah

- td = 0.3 detik
- tr = 0.7 detik
- tp = 2 detik
- Mp = 10.46%
- ts = 40

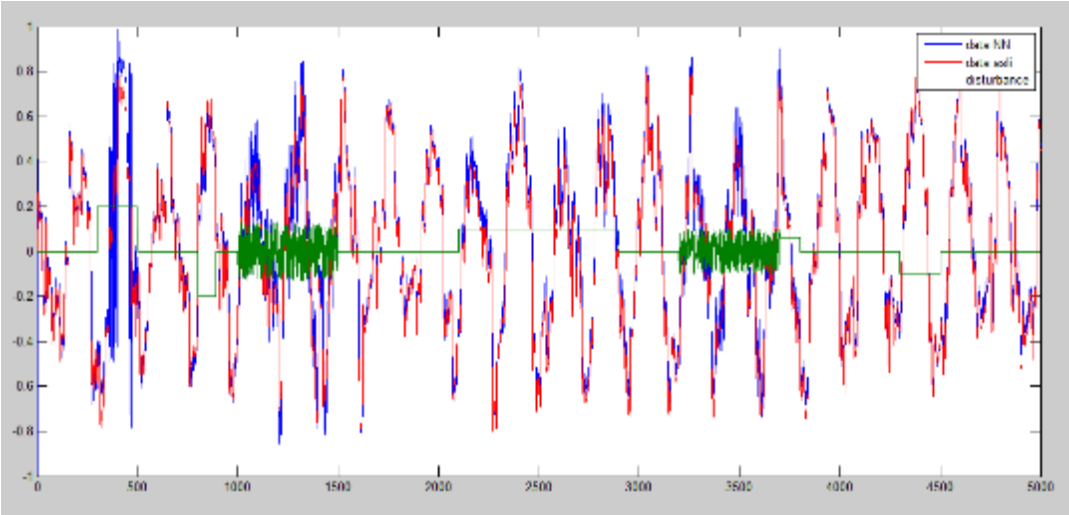
e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381



Gambar 36. Plot perbandingan keluaran NN *inverse* $\hat{\boldsymbol{u}}(\boldsymbol{t})$ dengan data masukan sistem $\boldsymbol{u}(\boldsymbol{t})$ pengujian pengendali *inverse* menggunakan data *step*

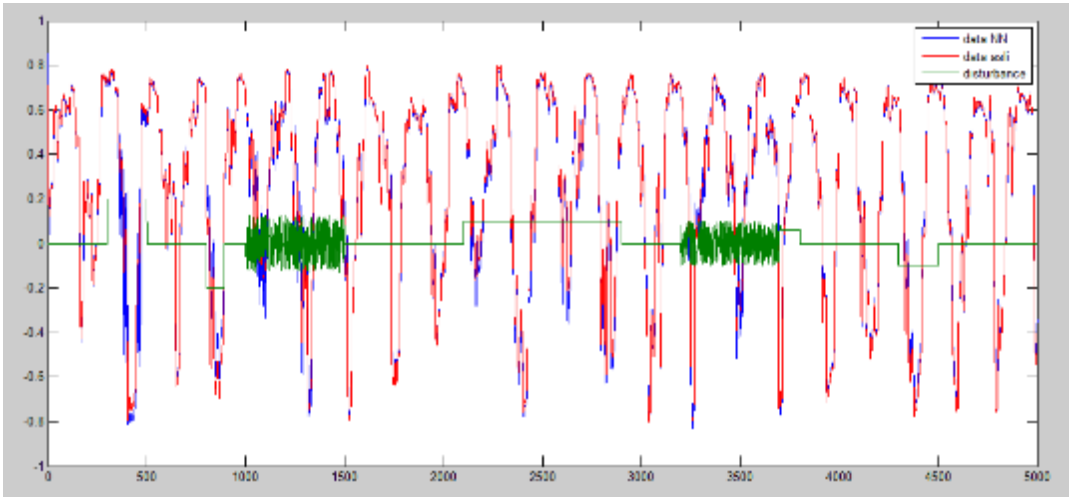
3.5.2. Pengujian DIC NN dengan Gangguan

Pada pengujian dengan memberikan gangguan pada DIC NN, hasil pengujian ditunjukkan pada Gambar 37 dan Gambar 38, dengan eror rata-rata keluaran NN *plant* 0.0016. Pada Gambar 37, terlihat bahwa ketika terdapat gangguan yang masuk, keluaran DIC NN mengalami kesulitan untuk mengikuti nilai referensinya.



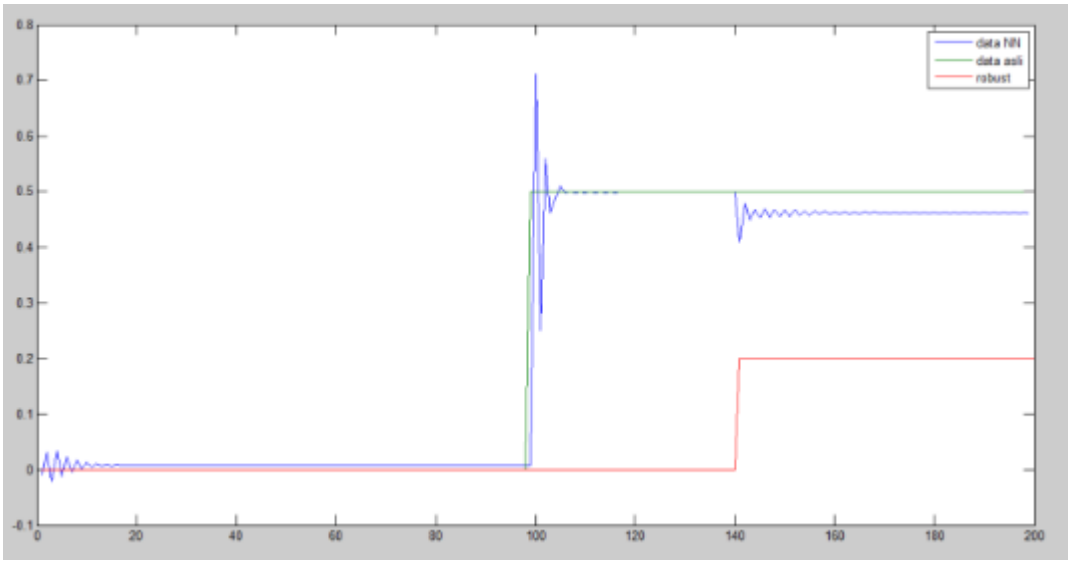
Gambar 37. Plot perbandingan keluaran NN *inverse* $\hat{\boldsymbol{u}}(\boldsymbol{t})$ dengan data masukan sistem $\boldsymbol{u}(\boldsymbol{t})$ pengujian DIC NN dengan data pembelajaran ketika diberi gangguan

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381



Gambar 38. Plot perbandingan keluaran NN *plant* dengan data keluaran sistem $\boldsymbol{y(t+1)}$ pengujian DIC NN dengan data pembelajaran ketika diberi gangguan

Pada pengujian dengan memberikan gangguan kepada sistem (Gambar 39), terlihat bahwa ketika diberi gangguan pada NN, keluaran NN tidak dapat kembali ke nilai referensi masukannya.



Gambar 39. Plot perbandingan keluaran NN *inverse* $\boldsymbol{\hat{u}(t)}$ dengan data masukan sistem $\boldsymbol{u(t)}$ pengujian pengendali *inverse* menggunakan data *step* ketika diberi gangguan

3.5.3. Perbandingan Performa

Pada subbab, akan dibandingkan performa antara NN pengendali berbasis eror dan DIC sebelum dan setelah diberi gangguan (*disturbance*) ketika menggunakan data pembelajaran dalam pengujian. Tabel hasil perbandingan performa dapat dilihat pada Tabel 2. Pada tabel terlihat bahwa pada perbandingan performa tanpa diberi gangguan, DIC NN memiliki nilai eror yang lebih kecil (4.6721E-04), namun ketika terdapat gangguan, eror pada DIC NN naik secara drastis (0.0016), melewati eror NN pengendali berbasis eror (0.001). Dari percobaan ini dapat

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381

disimpulkan bahwa NN pengendali berbasis eror lebih tahan terhadap gangguan jika dibandingkan dengan *Direct Inverse Control* NN.

Tabel 2. Perbandingan performa DIC NN dengan NN pengendali berbasis eror

	Tanpa gangguan	Dengan gangguan
DIC NN	4.8721E-04	0.0016
NN pengendali berbasis eror	7.7161E-04	0.001

4. KESIMPULAN

1. NN pengendali berbasis eror bekerja dengan data masukan berupa selisih dari data masukan referensi dengan data keluaran dari *plant*
2. Metode pembelajaran *backpropagation* efektif untuk melatih NN pengendali berbasis eror
3. Normalisasi data pelatihan dan metode faktor pengali mejadi satu dan minus satu mempermudah pencapaian konvergensi eror pada pembelajaran
4. Penggunaan NN pengendali pada pengujian *online* masih belum baik karena nilai keluarannya berada diluar nilai batas kerja (satu dan minus satu), sehingga perlu penanganan untuk memperbaikinya, yaitu dengan cara menggunakan metode faktor pengali agar nilai masukan ke NN *plant* berada di dalam nilai batas kerja
5. NN pengendali berbasis eror lebih tahan terhadap gangguan jika dibandingkan dengan *Direct Inverse Control* NN

DAFTAR PUSTAKA

[1] X. H. Wang, S. Wu, and Z. X. Jiao, "RBF-elman neural network control on electro-hydraulic load simulator," *Appl. Mech. Mater.*, vol. 433–435, pp. 1054–1060, 2013, doi: 10.4028/www.scientific.net/AMM.433-435.1054.

[2] M. H. Abd Wahab, A. Johari, T. H. Tg. Othman, H. F. Hanafi, and A. H. Ariffin, "Design and Development of Neural Network Simulator," *Malaysian J. Inf. Commun. Technol.*, pp. 39–49, 2016, doi: 10.53840/myjict1-1-75.

[3] S. Sapna, "Backpropagation Learning Algorithm Based on Levenberg Marquardt Algorithm," pp. 393–398, 2012, doi: 10.5121/csit.2012.2438.

[4] S. Nuraitul Janah, T. Hastono, N. Nurmala, and M. Nor Azhari Azman, "Implementation Network Nerves Imitation Using the Backpropagation Method For Prediction Sale Groceries," *Appl. Sci. Technol. Reaserch J.*, vol. 2, no. 2, pp. 56–66, 2023, doi: 10.31316/astro.v2i2.5796.

[5] T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, and G. Hinton, "Backpropagation and the brain," *Nat. Rev. Neurosci.*, vol. 21, no. 6, pp. 335–346, 2020, doi: 10.1038/s41583-020-0277-3.

[6] S. Ozbay, "Modified Backpropagation Algorithm with Multiplicative Calculus in Neural Networks," *Elektron. ir Elektrotechnika*, vol. 29, no. 3, pp. 55–61, 2023, doi: 10.5755/j02.eie.34105.

[7] S. Fan and Y. Zhao, "Analysis of des Plaintext Recovery Based on BP Neural Network," *Secur. Commun. Networks*, vol. 2019, 2019, doi: 10.1155/2019/9580862.

[8] A. Nikov and S. Stoeva, "Quick fuzzy backpropagation algorithm," *Neural Networks*, vol. 14, no. 2, pp. 231–244, 2001, doi: 10.1016/S0893-6080(00)00085-X.

[9] T. Korkobi, M. Djemel, and M. Chtourou, "Stability analysis of neural networks-based system identification," *Model. Simul. Eng.*, vol. 2008, 2008, doi: 10.1155/2008/343940.

[10] R. N. Singarimbun, O. E. Putra, N. L. W. S. R. Ginantra, and M. P. Dewi, "Backpropagation Artificial Neural Network Enhancement using Beale-Powell Approach Technique," *J. Phys. Conf. Ser.*, vol. 2394, no. 1, 2022, doi: 10.1088/1742-6596/2394/1/012007.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381

[11] R. Khalil and M. Younis, "Image Compression Technique Using a Hierarchical Neural Network," *AL-Rafidain J. Comput. Sci. Math.*, vol. 3, no. 2, pp. 99–112, 2006, doi: 10.33899/csmj.2006.164055.
[12] M. N. Nazarov, "Neural networks with dynamical coefficients and adjustable connections on the basis of integrated backpropagation," *Vestn. Udmurt. Univ. Mat. Mekhanika, Komp'yuternye Nauk.*, vol. 28, no. 2, pp. 260–274, 2018, doi: 10.20537/vm180212.
[13] A. R. Bohari, "Meningkatkan Kinerja Backpropagation Neural Network Menggunakan Algoritma Adaptif," *IMTechno J. Ind. Manag. Technol.*, vol. 3, no. 1, pp. 58–63, 2022, doi: 10.31294/imtechno.v3i1.1043.
[14] A. Agrawal, D. Mungra, and A. Thakkar, "Sentiment Analysis: An empirical comparison between various training algorithms for Artificial Neural Network," *Int. J. Innov. Comput. Appl.*, vol. 11, no. 1, p. 1, 2020, doi: 10.1504/ijica.2020.10026717.
[15] G. Shen, D. Zhao, and Y. Zeng, "Backpropagation with biologically plausible spatiotemporal adjustment for training deep spiking neural networks," *Patterns*, vol. 3, no. 6, 2022, doi: 10.1016/j.patter.2022.100522.
[16] T. Vesselenyi, S. Dziţac, I. Dziţac, and M.-J. Manolescu, "Fuzzy and Neural Controllers for a Pneumatic Actuator," *Int. J. Comput. Commun. Control*, vol. 2, no. 4, p. 375, 2007, doi: 10.15837/ijccc.2007.4.2368.
[17] M. N. F. Hidayat, "Prediction of Patients' Illness Based on Average Temperature and Rainfall In Az-Zainiyah Clinic Using Backpropagation Method," *TRILOGI J. Ilmu Teknol. Kesehatan, dan Hum.*, vol. 2, no. 3, pp. 239–251, 2021, doi: 10.33650/trilogi.v2i3.3169.
[18] M. Wahyudi, Firmansyah, Lise Pujiastuti, and Solikhun, "Application of Neural Network Variations for Determining the Best Architecture for Data Prediction," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 6, no. 5, pp. 742–748, 2022, doi: 10.29207/resti.v6i5.4356.
[19] "Optimization of breast cancer classification using feature selection on neural network," *J. Soft Comput. Explor.*, vol. 3, no. 2, 2022, doi: 10.52465/joscex.v3i2.78.
[20] T. Senjyu, Y. Morishima, T. Yamashita, K. Uezato, and H. Fujita, "Recurrent neural network supplementary stabilization controller for automatic voltage regulator and governor," *Electr. Power Components Syst.*, vol. 31, no. 7, pp. 693–707, 2003, doi: 10.1080/15325000390203683.
[21] I. K. A. G. Wiguna, P. Sugiartawan, I. G. I. Sudipa, and I. P. Y. Pratama, "Sentiment Analysis Using Backpropagation Method to Recognize the Public Opinion," *IJCCS (Indonesian J. Comput. Cybern. Syst.*, vol. 16, no. 4, p. 423, 2022, doi: 10.22146/ijccs.78664.
[22] O. A. Amalia and A. H. M. Putri, "Comparison of backpropagation artificial neural network and SARIMA in predicting the number of railway passengers," *J. Phys. Conf. Ser.*, vol. 1663, no. 1, 2020, doi: 10.1088/1742-6596/1663/1/012033.
[23] A. Rahmawati Sunaryo, N. Abelia, T. Hastono, and E. H. Pratisto, "Network Nerves Mock Backpropagation Prediction Graduation Student Elementary School With Practice Values Exam," *Appl. Sci. Technol. Reaserch J.*, vol. 2, no. 2, pp. 73–80, 2023, doi: 10.31316/astro.v2i2.5798.
[24] P. Kim, J. Lee, and C. S. Shin, "Classification of walking environments using deep learning approach based on surface emg sensors only," *Sensors*, vol. 21, no. 12, 2021, doi: 10.3390/s21124204.
[25] X. Fu, S. Li, M. Fairbank, D. C. Wunsch, and E. Alonso, "Training Recurrent Neural Networks with the Levenberg-Marquardt Algorithm for Optimal Control of a Grid-Connected Converter," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 26, no. 9, pp. 1900–1912, 2015, doi: 10.1109/TNNLS.2014.2361267.
[26] P. R. Vlachas *et al.*, "Backpropagation algorithms and Reservoir Computing in Recurrent Neural Networks for the forecasting of complex spatiotemporal dynamics," *Neural Networks*, vol. 126, pp. 191–217, 2020, doi: 10.1016/j.neunet.2020.02.016.
[27] S. E. Mohammed, D. Al-Bayati, and Y. J. Tawfeeq, "A New Model for Predicting Surface Pump Pressure of Drilling Rig Using Artificial Neural Network," *Pet. Chem.*, vol. 64, no. 7, pp. 747–755, 2024, doi: 10.1134/S0965544124050141.
[28] M. Arifeen, A. Petrovski, M. J. Hasan, I. Kotenko, M. Sletov, and P. Hassard, "DataDRILL:

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381

Formation Pressure Prediction and Kick Detection for Drilling Rigs," *arXiv Prepr. arXiv2409.19724*, 2024.

[29] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Front. Neurosci.*, vol. 12, no. MAY, 2018, doi: 10.3389/fnins.2018.00331.

[30] A. Zribi, M. Chtourou, and M. Djemel, "A new PID neural network controller design for nonlinear processes," *J. Circuits, Syst. Comput.*, vol. 27, no. 4, 2018, doi: 10.1142/S0218126618500652.

[31] F. Bonassi, M. Farina, J. Xie, and R. Scattolini, "On Recurrent Neural Networks for learning-based control: Recent results and ideas for future developments," *J. Process Control*, vol. 114, pp. 92–104, 2022, doi: 10.1016/j.jprocont.2022.04.011.

[32] E. R. W. van Doremaele, T. Stevens, S. Ringeling, S. Spolaor, M. Fattori, and Y. van de Burgt, "Hardware implementation of backpropagation using progressive gradient descent for in situ training of multilayer neural networks," *Sci. Adv.*, vol. 10, no. 28, 2024, doi: 10.1126/sciadv.ado8999.

[33] C. Jia *et al.*, "Prediction of Drilling Efficiency for Rotary Drilling Rig Based on an Improved Back Propagation Neural Network Algorithm," *Machines*, vol. 12, no. 7, 2024, doi: 10.3390/machines12070438.

[34] E. A. Muravyova, M. I. Sharipov, and K. R. Zubaydullina, "Development of neural network controller for control of debutanization process in rectification column of gas-fractionating plant," *J. Phys. Conf. Ser.*, vol. 2373, no. 5, 2022, doi: 10.1088/1742-6596/2373/5/052013.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v24i3.13381