UNIVERSITAS INDONESIA

SISTEM KENDALI BERBASIS *ARTIFICIAL NEURAL NETWORK* DENGAN METODE *DIRECT INVERSE CONTROL*

TUGAS MAKALAH UAS

SISTEM BERBASIS PENGETAHUAN

DEVIN EZEKIEL PURBA

2106701583

FAKULTAS TEKNIK

PROGRAM STUDI TEKNIK ELEKTRO

DEPOK

DESEMBER 2024

BAB 1
PENDAHULUAN

1.1. Latar Belakang

Sistem dinamik memiliki peran yang sangat penting dalam berbagai bidang, mulai dari kontrol industri, otomasi, hingga pemodelan fenomena alam yang kompleks. Sistem-sistem ini, yang dapat berupa mesin-mesin industri, kendaraan otomatis, maupun perangkat elektronik lainnya, beroperasi secara berkesinambungan dan membutuhkan kontrol yang tepat agar tetap dapat bekerja secara optimal. Oleh karena itu, diperlukan metode kontrol yang mampu menangani kompleksitas sistem dinamik dengan presisi tinggi, terutama untuk sistem yang memiliki sifat non-linear.

Pemodelan dan identifikasi sistem dinamik menjadi langkah awal yang esensial dalam memahami dan mengendalikan sistem-sistem tersebut. Pemodelan sistem dinamik melibatkan pembuatan model matematis yang merepresentasikan perilaku dan respons sistem terhadap berbagai masukan. Sedangkan identifikasi sistem adalah proses menentukan parameter-parameter dalam model tersebut berdasarkan data yang diperoleh dari pengamatan atau pengujian sistem nyata. Kedua proses ini merupakan dasar dalam pengembangan sistem kendali yang efektif.

Dalam beberapa dekade terakhir, neural network atau jaringan saraf tiruan telah menunjukkan potensi besar dalam pemodelan dan pengendalian sistem dinamik yang kompleks. Neural network memiliki kemampuan adaptasi yang tinggi dalam mempelajari hubungan non-linear antara variabel masukan dan keluaran berdasarkan data. Kemampuan ini menjadikan neural network sebagai alat yang sangat kuat untuk menggantikan atau melengkapi metode konvensional, terutama ketika menghadapi sistem yang sulit dimodelkan secara matematis.

Salah satu pendekatan dalam kendali sistem berbasis neural network adalah metode Direct Inverse Control (DIC). Metode ini memanfaatkan neural network untuk mempelajari model invers dari sistem, di mana model invers ini digunakan secara langsung sebagai pengendali. Dengan kata lain, neural network dilatih untuk memprediksi masukan yang diperlukan berdasarkan keluaran yang diinginkan.

Pendekatan ini memiliki keunggulan dalam menyederhanakan desain pengendali, karena tidak memerlukan perancangan model sistem secara eksplisit.

Metode Direct Inverse Control sangat relevan untuk sistem dengan karakteristik umpan balik (feedback), di mana keluaran sistem saat ini (y[k]) tidak hanya dipengaruhi oleh masukan saat ini (u[k]), tetapi juga oleh keluaran sebelumnya (y[k-1]) dan masukan pada waktu sebelumnya (u[k-1]). Kompleksitas ini membutuhkan pendekatan kendali yang canggih, seperti yang ditawarkan oleh neural network. Dengan kemampuan belajar dan generalisasi yang dimiliki neural network, metode ini dapat mengatasi tantangan pengendalian sistem non-linear secara efektif.

Dalam laporan ini, akan dibahas bagaimana neural network diterapkan pada metode Direct Inverse Control untuk mengendalikan sistem dinamik non-linear. Fokus utamanya adalah pada pemanfaatan neural network untuk mempelajari hubungan antara masukan dan keluaran sistem, serta bagaimana model invers yang dihasilkan dapat digunakan untuk mencapai kontrol yang optimal. Diharapkan, penelitian ini dapat memberikan kontribusi signifikan dalam pengembangan teknik kontrol berbasis pembelajaran mesin untuk aplikasi di berbagai bidang, seperti otomasi, robotika, dan energi terbarukan.

Dengan memahami dan mengeksplorasi potensi metode Direct Inverse Control berbasis neural network, laporan ini bertujuan untuk memberikan wawasan baru dalam bidang kendali adaptif yang berbasis data. Penerapan pendekatan ini diharapkan mampu meningkatkan efisiensi dan akurasi dalam pengendalian sistem dinamik yang kompleks.

1.2. Rumusan Masalah

1.2.1. Bagaimana cara menerapkan metode Direct Inverse Control menggunakan neural network untuk mengendalikan sistem dinamik diskrit?

1.2.2. Bagaimana neural network mempelajari hubungan input dan output pada sistem dengan karakteristik umpan balik?

1.2.3. Seberapa akurat metode Direct Inverse Control berbasis neural network dalam memprediksi masukan yang dibutuhkan untuk mencapai keluaran sistem yang diinginkan?

1.2.4. Apakah metode Direct Inverse Control berbasis neural network efektif untuk mengendalikan sistem non-linear dan kompleks?

1.3. Tujuan

1.3.1. Menerapkan metode Direct Inverse Control untuk mengendalikan sistem dinamik diskrit berbasis neural network.

1.3.2. Memahami mekanisme pembelajaran neural network dalam mempelajari hubungan antara input dan output pada sistem dengan sifat umpan balik.

1.3.3. Mengevaluasi tingkat akurasi metode Direct Inverse Control berbasis neural network dalam menghasilkan masukan yang sesuai dengan keluaran yang diinginkan.

1.3.4. Memberikan solusi kendali yang efektif untuk sistem non-linear dan kompleks menggunakan metode Direct Inverse Control.

1.4. Pembatasan Masalah

1.4.1. Sistem yang dikendalikan dibatasi pada sistem dinamik diskrit dengan persamaan yang memiliki sifat umpan balik, di mana keluaran saat ini bergantung pada keluaran dan masukan sebelumnya.

1.4.2. Neural network yang digunakan terbatas pada model jaringan saraf tiruan feedforward dengan metode pembelajaran supervised learning.

1.4.3. Data yang digunakan untuk pelatihan dan pengujian neural network diambil dari simulasi sistem yang telah ditentukan, tanpa melibatkan data nyata dari sistem fisik.

1.4.4. Evaluasi kinerja neural network akan difokuskan pada tingkat akurasi prediksi dan kemampuan model dalam mempelajari pola hubungan input-output saja

1.4.5. Laporan ini hanya membahas penerapan metode Direct Inverse Control untuk pengendalian sistem, tanpa melakukan evaluasi lebih lanjut pada desain pengendali yang kompleks.

BAB 2
STATE OF THE ART

2.1. Artificial Neural Network (ANN)

Artificial Neural Network (ANN) adalah model komputasi yang terinspirasi oleh cara kerja otak manusia. ANN terdiri dari neuron-neuron buatan yang dihubungkan oleh bobot (weights) yang dapat diatur. ANN dapat belajar dari data yang diberikan melalui proses pelatihan (training), dan berfungsi untuk mengenali pola atau memetakan input ke output tertentu.

Secara umum, struktur ANN dapat dibagi menjadi tiga bagian utama:

- **Lapisan Input (Input Layer):** Lapisan ini menerima data mentah dari luar jaringan dan meneruskannya ke lapisan berikutnya.
- **Lapisan Tersembunyi (Hidden Layers):** Lapisan ini melakukan pemrosesan terhadap data dan mengekstrak fitur-fitur yang relevan. Lapisan ini dapat terdiri dari satu atau lebih lapisan.
- **Lapisan Output (Output Layer):** Lapisan ini menghasilkan prediksi akhir berdasarkan pemrosesan yang dilakukan oleh lapisan-lapisan tersembunyi.

Pada setiap neuron, output dihasilkan melalui persamaan:

$$a_j = f\left(\sum_{i=1}^{n} w_{ij} \cdot x_i + b_j\right)$$

di mana:

- w_{ij} adalah bobot antara neuron ke-iii di lapisan sebelumnya dengan neuron ke-jjj di lapisan saat ini
- x_i adalah input
- b_j adalah bias
- f adalah fungsi aktivasi

2.2. Supervised Learning

Supervised learning adalah metode pembelajaran mesin di mana model dilatih menggunakan data input-output yang sudah diberi label. Pada ANN, data input dipasangkan dengan output yang diharapkan. Selama proses pelatihan, jaringan belajar untuk memetakan input ke output yang benar dengan meminimalkan error (loss) antara output jaringan dan output yang diharapkan.

Contoh loss function yang sering digunakan adalah Mean Squared Error (MSE):

$$J = \frac{1}{N}\sum_{k=1}^{N}(y_k - \widehat{y_k})^2$$

di mana:

- J adalah nilai Loss atau error
- y_k adalah nilai output yang diharapkan
- $\widehat{y_k}$ adalah nilai output yang diprediksi oleh jaringan

2.3. Forward Propagation

Forward propagation adalah proses di mana data input diteruskan melalui jaringan, dari lapisan input ke lapisan output, untuk menghasilkan prediksi akhir. Pada setiap lapisan, hasil kalkulasi berupa dot product antara input dan bobot, ditambahkan dengan bias, kemudian diproses melalui fungsi aktivasi.

Persamaan forward propagation:

$$z_j = \sum_{i=1}^{n} w_{ij} \cdot x_i + b_j$$

$$a_j = f(z_j)$$

di mana:

- z_j adalah input bersih yang masuk ke neuron
- a_j adalah output neuron setelah fungsi aktivasi
- f adalah fungsi aktivasi seperti sigmoid atau tanh

Untuk aplikasi sistem kendali, fungsi aktivasi yang biasanya digunakan adalah **fungsi tanh (hyperbolic tangent).** Fungsi tanh memiliki range output antara -1 dan 1, yang cocok untuk sistem kendali yang umumnya bekerja pada rentang nilai negatif dan positif. Persamaan fungsi tanh adalah:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

2.4. Backward Propagation

Backward propagation adalah langkah penting dalam pelatihan *neural network* yang digunakan untuk memperbarui parameter-parameter jaringan, yaitu *weights* dan *biases*, sehingga jaringan dapat belajar dari kesalahan yang terjadi selama proses pelatihan. Proses ini memungkinkan jaringan saraf tiruan untuk meminimalkan fungsi kerugian (loss function) dengan cara menghitung gradien dan memperbarui parameter menggunakan algoritma optimisasi, seperti *Gradient Descent*.

Pada dasarnya, *backward propagation* bekerja dengan cara sebagai berikut:

1. Mencari *error* atau kesalahan dari prediksi jaringan dibandingkan dengan nilai sebenarnya.
2. Menghitung gradien (turunan) dari fungsi kerugian terhadap setiap *weight* dan *bias* dengan menggunakan *Chain Rule* dari kalkulus.
3. Memperbarui *weights* dan *biases* berdasarkan gradien tersebut untuk meminimalkan kesalahan pada iterasi berikutnya.

Persamaan pembaruan bobot yang menggunakan algoritma **Gradient Descent Optimizer**:

$$w_{ij} = w_{ij} - \alpha \frac{\partial J}{\partial w_{ij}}$$

$$b_j = b_j - \alpha \frac{\partial J}{\partial b_j}$$

Di mana:

- w_{ij} adalah *weight* yang menghubungkan neuron ke-i pada lapisan sebelumnya dengan neuron ke-j pada lapisan saat ini

- b_j adalah *bias* neuron ke-j
- α adalah *learning rate*, yang menentukan seberapa besar pembaruan yang dilakukan
- $\frac{\partial J}{\partial w_{ij}}$ adalah gradien dari fungsi kerugian J terhadap *weight* w_{ij}.
- $\frac{\partial J}{\partial b_j}$ adalah gradien dari fungsi kerugian J terhadap *bias* b_j

Gradien tersebut diperoleh dengan menggunakan turunan parsial dari fungsi kerugian dan menerapkan *Chain Rule*. Perhitungan gradien ini memungkinkan jaringan untuk mengetahui arah di mana kesalahan dapat diminimalkan dan oleh karena itu, dapat memperbarui parameter-parameter jaringan untuk menghasilkan prediksi yang lebih akurat. Selama proses *backward propagation*, turunan dari *activation function* memainkan peran penting. Turunan ini diperlukan untuk menghitung gradien dari fungsi kerugian terhadap *weights* dan *biases*. Misalnya, ketika menggunakan fungsi aktivasi *tanh*, turunan dari fungsi *tanh* digunakan untuk mengukur sensitivitas kesalahan terhadap perubahan pada neuron. Turunan dari fungsi *tanh* adalah:

$$f'(x) = 1 - \tanh^2(x)$$

Turunan ini digunakan dalam proses *backward propagation* untuk menghitung perubahan yang perlu diterapkan pada setiap parameter jaringan. Jika keluaran dari neuron adalah A dan fungsi aktivasi adalah f, maka turunan dari keluaran neuron tersebut adalah $f'(Z)$, di mana Z adalah nilai sebelum diterapkan fungsi aktivasi:

$$dZ = dA \cdot f'(Z)$$

di sini:

- dZ adalah gradien dari fungsi kerugian terhadap Z,
- dA adalah gradien dari fungsi kerugian terhadap keluaran neuron, dan
- $f'(Z)$ adalah turunan dari fungsi aktivasi (misalnya *tanh*).

2.5. Normalisasi Min-Max Scaler

Normalisasi adalah teknik untuk menskalakan data sehingga berada dalam rentang tertentu, misalnya [0, 1] atau [-1, 1]. Hal ini membantu dalam mempercepat proses pelatihan karena memastikan bahwa nilai input berada dalam rentang yang konsisten. Persamaan normalisasi:

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

Denormalisasi adalah proses mengembalikan data yang telah dinormalisasi ke skala aslinya:

$$x = x_{norm} \cdot (\max(x) - \min(x)) + \min(x)$$

2.6. Inisialisasi Bobot Nguyen-Widrow

Metode Nguyen-Widrow adalah teknik inisialisasi bobot untuk mempercepat proses pelatihan jaringan. Inisialisasi yang baik dapat membantu jaringan untuk konvergen lebih cepat. Persamaan Nguyen-Widrow:

$$W = \beta \frac{W_{random}}{||W\_random||}$$

dengan $\beta = 0.7 \cdot (output\ size)^{\frac{1}{input\ size}}$

2.7. L2 Regularization

Overfitting terjadi ketika model belajar terlalu baik terhadap data pelatihan sehingga tidak mampu menangani data baru dengan baik. Underfitting terjadi saat model gagal belajar pola-pola penting dari data pelatihan.

L2 Regularization adalah teknik untuk mencegah overfitting dengan menambahkan penalti terhadap bobot besar. Fungsi loss dengan L2 Regularization adalah:

$$J_{total} = J_{original} + \lambda \sum_j w_j^2$$

Dimana λ adalah parameter regularisasi.

2.8. Identifikasi Sistem dengan Metode Output Feedback Secara Vektoral pada ANN

Metode ini menggunakan feedback dari output sistem sebagai input tambahan ke jaringan. ANN menerima input dari beberapa langkah waktu sebelumnya, memungkinkan untuk memodelkan sistem dinamis:

$$y[k] = f(X[k])$$

$$X[k] = (y[k-1], y[k-2], u[k], u[k-1], u[k-2])$$

Metode ini membuat jaringan dapat belajar pola feedback dan memprediksi output berdasarkan input masa lalu yang umumnya digunakan dalam sistem kendali. Hal ini membuat pelatihan dalam matriks tidak memungkinkan karena data input yang dilakukan umpan balik akan berubah terhadap waktu dan terhadap sistemnya sehingga input data pada sistem harus berupa vektoral.

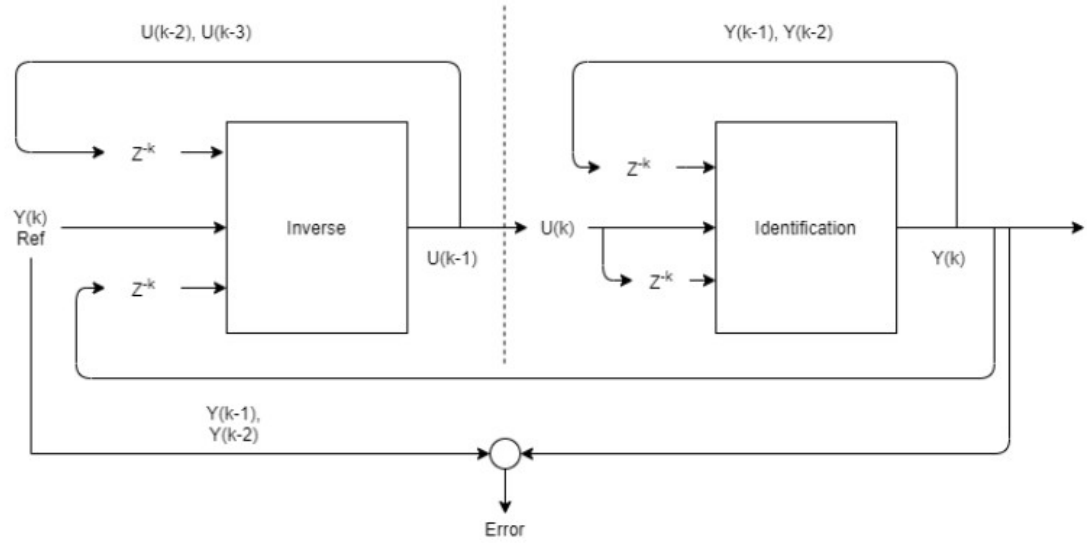
2.9. Direct Inverse Control Berbasis ANN

Metode ini menggunakan feedback dari output pengendali sebagai input tambahan ke jaringan. ANN menerima input dari beberapa langkah waktu sebelumnya, memungkinkan untuk memodelkan sistem dinamis:

$$u[k] = f(X[k])$$

$$X[k] = (y[k-1], y[k-2], y[k], u[k-1], u[k-2])$$

Direct Inverse Control (DIC) adalah pendekatan kendali di mana model invers sistem dinamik dipelajari dan digunakan sebagai pengendali. Pada metode ini, neural network (ANN) dilatih untuk memetakan hubungan antara keluaran sistem yang diinginkan (desired output) dengan masukan sistem yang diperlukan untuk mencapainya. ANN berfungsi sebagai model invers sistem, yang berarti neural network menentukan masukan sistem u[k] berdasarkan keluaran yang diinginkan.



BAB 3
METODOLOGI

3.1. Data Collection

Pada tahap Pengumpulan Data dalam program ini, proses dimulai dengan mengimpor beberapa pustaka penting seperti numpy untuk operasi numerik, matplotlib.pyplot untuk visualisasi grafik, pandas untuk manipulasi data (meskipun tidak digunakan secara eksplisit pada bagian ini), seaborn untuk memperindah tampilan grafik, serta sklearn.preprocessing dan scipy.signal untuk tugas penskalaan dan pemrosesan sinyal. Langkah pertama adalah menetapkan random seed menggunakan np.random.seed(42) untuk memastikan hasil yang konsisten setiap kali program dijalankan. Selanjutnya, program mendefinisikan variabel waktu dengan k_max = 10000 yang menunjukkan jumlah total sampel data. Array t dibuat menggunakan np.linspace(0, k_max, k_max), yang menghasilkan nilai yang terdistribusi secara linear dari 0 hingga k_max. Variabel t ini berfungsi sebagai sumbu waktu untuk semua grafik. Program kemudian menghasilkan sinyal input acak u(k) menggunakan u = 2 * np.random.uniform(-5, 5, k_max). Sinyal ini memiliki rentang nilai antara -10 hingga 10, yang mencerminkan berbagai kondisi yang bisa memengaruhi sistem, memberikan variasi data yang kaya. Setelah itu, array y diinisialisasi dengan y = np.zeros(k_max), yang berfungsi untuk menyimpan nilai keluaran sistem pada setiap indeks waktu k. Pada tahap ini, array y diisi dengan nilai nol sebagai langkah awal. Penghitungan keluaran sistem y(k) dilakukan melalui sebuah loop, di mana nilai y(k) ditentukan berdasarkan nilai y(k-1) (keluaran sebelumnya) serta input saat ini u(k) dan input sebelumnya u(k-1). Persamaan yang digunakan adalah:

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

Persamaan ini menggambarkan sistem dinamik non-linear di mana keluaran saat ini y(k) dipengaruhi oleh keluaran dan input dari langkah-langkah sebelumnya. Untuk melengkapi pengumpulan data, program membuat beberapa array yang berisi shifted values atau nilai geser, yaitu y(k-1), y(k-2), u(k-1), dan u(k-2). Array ini dibuat dengan tujuan untuk menyimpan keluaran dan input dari satu hingga dua langkah sebelumnya, yang penting untuk analisis data deret waktu. Proses ini dilakukan dengan menggeser data dalam array:

- y_k_1[1:] = y[:-1] menyimpan nilai y(k-1)
- y_k_2[2:] = y[:-2] menyimpan nilai y(k-2)
- Array u_k_1 dan u_k_2 dibuat dengan metode serupa untuk input u(k-1) dan u(k-2).

Sebagai langkah akhir dari pengumpulan data, program membuat visualisasi menggunakan grid 3x2 untuk menunjukkan hubungan input-output. Pada baris pertama, program menampilkan grafik u(k) dan y(k) untuk memperlihatkan sinyal input dan output saat ini. Baris kedua menampilkan u(k-1) dan y(k-1) untuk menunjukkan bagaimana sinyal input dan output di masa lalu berperilaku, sementara baris ketiga menggambarkan u(k-2) dan y(k-2) yang menunjukkan keadaan input dan output pada dua langkah sebelumnya. Setiap grafik dilengkapi dengan judul, label sumbu, dan legenda untuk memperjelas interpretasi data. Terakhir, plt.tight_layout() digunakan untuk memastikan tata letak grafik tidak saling tumpang tindih, dan plt.show() menampilkan seluruh grafik tersebut. Tahap pengumpulan data ini bertujuan untuk menghasilkan data sintetik yang mencerminkan perilaku sistem dinamik. Dengan menciptakan sinyal input acak dan menghitung keluaran berdasarkan model non-linear, serta membuat visualisasi yang informatif, program ini menyediakan data dasar yang dapat digunakan untuk analisis lebih lanjut atau pemodelan.

3.2. Data Preprocessing

Pada tahap Pra-pemrosesan Data dalam program ini, proses dimulai dengan mengatur data mentah yang telah dikumpulkan menjadi bentuk yang siap untuk digunakan dalam pelatihan model. Langkah pertama adalah menggabungkan beberapa variabel menjadi satu matriks fitur (X). Matriks X ini terdiri dari data yang telah dihasilkan sebelumnya, yaitu y(k-1), y(k-2), u(k), u(k-1), dan u(k-2), yang disusun menggunakan np.array([y_k_1, y_k_2, u, u_k_1, u_k_2]).T. Variabel target y juga diubah menjadi bentuk kolom dengan y.reshape(-1, 1), sementara sinyal input u disusun dengan cara yang sama.

Selanjutnya, proses normalisasi dilakukan untuk memastikan bahwa semua fitur berada dalam rentang nilai yang sama, yang sangat penting untuk mempercepat konvergensi selama pelatihan model dan meningkatkan kinerja. Program menggunakan MinMaxScaler dari pustaka sklearn.preprocessing untuk merubah nilai fitur ke dalam rentang antara -1 dan 1. Langkah ini bertujuan untuk menghindari dominasi fitur

dengan skala besar dan memastikan bahwa semua fitur memiliki dampak yang seimbang pada model. Normalisasi dilakukan pada variabel input u, u(k-1), dan u(k-2), serta pada output y, y(k-1), dan y(k-2). Setelah fitting menggunakan data input asli, program melakukan transformasi untuk mengubah data ke dalam skala yang dinormalisasi. Matriks fitur yang telah dinormalisasi, X_norm, kemudian disusun menggunakan array np.array([y_k_1_norm.flatten(), y_k_2_norm.flatten(), u_norm.flatten(), u_k_1_norm.flatten(), u_k_2_norm.flatten()]).T. Data ini kini siap untuk digunakan dalam pelatihan dan validasi model.

Sebagai langkah berikutnya, program melakukan pembagian data menjadi dua bagian, yaitu data pelatihan dan data validasi. Sebanyak 80% dari total data digunakan sebagai data pelatihan (X_train dan y_train), sedangkan 20% sisanya digunakan untuk validasi (X_val dan y_val). Pembagian ini juga diterapkan pada data yang telah dinormalisasi (X_train_norm, y_train_norm, X_val_norm, dan y_val_norm). Dengan demikian, program memastikan bahwa model dapat dilatih dengan baik menggunakan data yang cukup banyak, sementara validasi dengan data yang terpisah akan membantu menilai kemampuan generalisasi model pada data baru yang tidak dilihat selama proses pelatihan. Tahap pra-pemrosesan data ini memastikan bahwa data input dan output berada dalam format yang sesuai, dinormalisasi, dan tersegmentasi dengan baik untuk melatih dan memvalidasi model secara efektif, sehingga meningkatkan kualitas dan keakuratan proses pembelajaran mesin.

3.3. Training dan Validasi Tahap 1 (Plant)

Pada tahap Training dan Validasi Tahap 1, program menggunakan kelas MLP (Multi-Layer Perceptron) untuk membangun, melatih, dan memvalidasi model jaringan saraf tiruan yang dirancang untuk memprediksi perilaku sistem berdasarkan input-output yang telah dinormalisasi. Kelas MLP diinisialisasi dengan beberapa parameter, seperti jumlah lapisan (layers), fungsi aktivasi (activations), jenis optimizer (optimizer), laju pembelajaran (learning_rate), dan parameter regulasi L2 (l2_lambda). Pada awalnya, bobot diinisialisasi menggunakan metode Nguyen-Widrow yang membantu mempercepat konvergensi dengan mengatur bobot awal secara efisien. Selain itu, bias diinisialisasi dengan nilai nol.

Selama proses forward pass, input diteruskan melalui setiap lapisan jaringan untuk menghasilkan output prediksi, di mana setiap input dikalikan dengan bobot,

ditambahkan bias, dan diproses menggunakan fungsi aktivasi seperti tanh atau linear. Kemudian, pada backward pass, program menghitung gradien kerugian menggunakan metode backpropagation untuk menentukan perubahan yang diperlukan pada bobot dan bias, sambil menerapkan regulasi L2 untuk mengurangi risiko overfitting. Pembaruan bobot dilakukan menggunakan algoritma optimasi yang dipilih seperti adam, rmsprop, atau SGD, yang mempertimbangkan parameter seperti momentum dan rata-rata kuadrat tertimbang.

Selama proses pelatihan, model dilatih melalui beberapa epoch. Pada setiap epoch, prediksi dihasilkan dari data pelatihan menggunakan forward pass, kemudian gradien dihitung melalui backward pass, dan bobot diperbarui. Jika data validasi tersedia, model juga dievaluasi pada data tersebut, dan nilai kerugian serta akurasi disimpan untuk memantau kinerja selama pelatihan. Program juga menyediakan visualisasi berupa grafik yang menampilkan perbandingan kerugian dan akurasi antara data pelatihan dan validasi, menggunakan skala logaritmik pada sumbu epoch untuk memudahkan pemantauan perubahan tren selama pelatihan.

Setelah pelatihan selesai, model digunakan untuk prediksi pada data baru atau data yang telah dilatih, di mana hasil prediksi dibandingkan dengan nilai aktual untuk melihat seberapa baik model dapat menggeneralisasi pola yang dipelajari. Program juga menyajikan grafik perbandingan antara nilai prediksi dan nilai aktual untuk data pelatihan dan validasi, memberikan gambaran visual tentang akurasi model dalam mereplikasi dinamika sistem yang dipelajari. Secara keseluruhan, proses ini bertujuan untuk menghasilkan model jaringan saraf tiruan yang stabil, akurat, dan mampu menggeneralisasi dengan baik pada data yang belum pernah dilihat sebelumnya.

3.4. Training dan Validasi Tahap 2 (Plant)

Pada tahap Training dan Validasi Tahap 2, program menggunakan kelas MLPVectoral untuk melatih dan memvalidasi model jaringan saraf tiruan yang dirancang dengan mekanisme umpan balik (feedback). Kelas ini mirip dengan MLP pada tahap pertama, tetapi dengan penekanan lebih pada integrasi feedback dari prediksi sebelumnya untuk meningkatkan akurasi prediksi berurutan. Kelas MLPVectoral diinisialisasi dengan parameter seperti jumlah lapisan (layers), fungsi aktivasi (activations), jenis optimizer (optimizer), laju pembelajaran (learning_rate), dan parameter regulasi L2 (l2_lambda). Kelas ini juga memungkinkan untuk

menggunakan bobot dan bias yang telah ditentukan sebelumnya, yang dapat menghemat waktu pelatihan jika model sudah pernah dilatih. Salah satu perbedaan utama dari tahap pertama adalah penggunaan mekanisme umpan balik dalam proses pelatihan. Pada bagian ini, selama proses training, nilai prediksi yang dihasilkan (y_pred) pada setiap langkah digunakan sebagai input untuk langkah berikutnya (y(k-1) dan y(k-2)). Hal ini menciptakan dinamika yang lebih realistis karena model dapat belajar dari pola temporal yang ada dalam data, menjadikannya lebih efektif dalam memodelkan sistem yang membutuhkan prediksi berurutan.

Selama proses forward pass, model menerapkan fungsi aktivasi seperti tanh atau linear pada setiap lapisan. Input diolah dengan bobot dan bias, kemudian diaktifkan untuk menghasilkan output pada lapisan berikutnya. Ketika masuk ke dalam backward pass, gradien kerugian dihitung dan digunakan untuk meng-update bobot dan bias melalui algoritma optimasi seperti adam, rmsprop, atau SGD, serupa dengan metode pada tahap pertama. Namun, perhitungan gradien di sini memperhitungkan pengaruh feedback, yang membantu dalam melatih model untuk menangkap pola sekuensial dalam data. Selama proses pelatihan, model dilatih dalam beberapa epoch, dan untuk setiap epoch, data training diolah dengan feedback yang memperbarui input berdasarkan hasil prediksi sebelumnya. Setelah menyelesaikan satu epoch, model juga mengevaluasi kinerja pada data validasi untuk menghitung metrik seperti kerugian dan akurasi, yang kemudian dicatat untuk pemantauan. Program juga menyediakan visualisasi untuk melihat tren kerugian dan akurasi dari pelatihan dan validasi, yang membantu dalam memahami apakah model mengalami overfitting atau underfitting.

Perbedaan mendasar antara tahap pertama dan tahap kedua adalah bahwa MLP pada tahap pertama melakukan pelatihan tanpa feedback eksplisit, artinya setiap prediksi independen dari prediksi sebelumnya. Sementara itu, MLPVectoral pada tahap kedua memanfaatkan mekanisme umpan balik yang memungkinkan model untuk belajar dari urutan data, meningkatkan kemampuan prediksi dalam situasi di mana terdapat pola temporal. Dengan demikian, tahap kedua lebih cocok untuk masalah yang memerlukan analisis sekuensial atau prediksi yang bergantung pada kondisi sebelumnya, memberikan hasil yang lebih akurat dan konsisten pada data yang memiliki ketergantungan waktu. Training dan validasi tahap kedua dengan MLPVectoral meningkatkan kemampuan model dalam mengidentifikasi pola sekuensial dengan menggunakan feedback, sehingga lebih unggul dalam menangani

data berurutan dan menghasilkan prediksi yang lebih cermat dibandingkan dengan model yang dilatih pada tahap pertama tanpa feedback.

3.5. Testing Hasil Identifikasi Sistem (Plant)

Pada tahap Testing Hasil Identifikasi Sistem, program bertujuan untuk mengevaluasi kemampuan model dalam memprediksi keluaran sistem berdasarkan berbagai sinyal input baru. Tahap ini melibatkan pembuatan sinyal uji, seperti gelombang sinusoidal, kotak, dan segitiga, dengan parameter tertentu (frekuensi dan amplitudo) yang kemudian digunakan sebagai input untuk sistem. Setiap sinyal input diolah untuk menghasilkan keluaran sistem y_test dengan menggunakan persamaan yang sama seperti yang dipakai saat pelatihan. Data y_test dan u_test ini kemudian dinormalisasi menggunakan MinMaxScaler yang telah dilatih dengan data pelatihan, memastikan skala yang konsisten antara data uji dan data pelatihan. Setelah normalisasi, matriks input X_test dibentuk dari nilai-nilai y(k-1), y(k-2), u(k), u(k-1), dan u(k-2), yang kemudian digunakan untuk menghasilkan prediksi keluaran menggunakan model nn (dari Tahap 1) dan nnv (dari Tahap 2).

Program membuat prediksi keluaran (y_pred) dan membandingkannya dengan nilai y_test aktual untuk menilai akurasi model. Selain itu, hasil uji divisualisasikan dalam grafik yang menampilkan perbandingan antara sinyal input dengan keluaran yang dihasilkan, serta perbandingan antara nilai aktual dan prediksi dari model Tahap 1 dan Tahap 2. Penggunaan tiga jenis sinyal input ini memungkinkan pengujian yang lebih komprehensif terhadap model, memastikan model tidak hanya mengenali satu pola tetapi dapat menggeneralisasi dan mengidentifikasi berbagai pola dinamis. Grafik perbandingan membantu memberikan gambaran visual tentang seberapa akurat model dalam mereplikasi perilaku sistem yang sebenarnya. Secara keseluruhan, tahap ini penting untuk menilai efektivitas model yang dilatih dan memastikan bahwa model mampu memberikan prediksi yang konsisten dan akurat untuk berbagai skenario input.

3.6. Training dan Validasi Tahap 1 (Controller)

Pada tahap Training dan Validasi Tahap 1, program menggunakan kelas MLP (Multi-Layer Perceptron) untuk membangun, melatih, dan memvalidasi model neural network yang dirancang sebagai inverse controller. Neural network ini dirancang untuk mempelajari hubungan antara keluaran yang diinginkan (referensi y_ref) dan masukan kendali (u[k]) yang diperlukan untuk mencapai keluaran tersebut pada plant. Kelas

MLP diinisialisasi dengan parameter seperti jumlah lapisan (layers), fungsi aktivasi (activations), jenis optimizer (optimizer), laju pembelajaran (learning_rate), dan parameter regulasi L2 (l2_lambda). Bobot awal diinisialisasi menggunakan metode Nguyen-Widrow, sementara bias diatur ke nol untuk mempercepat konvergensi. Selama proses forward pass, referensi keluaran y_ref[k] dan keluaran plant sebelumnya y[k-1], y[k-2] diteruskan melalui setiap lapisan neural network untuk menghasilkan sinyal kendali u[k]. Neural network memproses input dengan mengalikan bobot, menambahkan bias, dan menerapkan fungsi aktivasi seperti tanh atau linear. Pada backward pass, program menghitung gradien kerugian menggunakan metode backpropagation untuk mengoptimalkan bobot dan bias, dengan regulasi L2 untuk mengurangi risiko overfitting. Pembaruan bobot dilakukan menggunakan algoritma optimasi seperti adam, rmsprop, atau SGD.

Selama pelatihan, model dilatih melalui beberapa epoch. Pada setiap epoch, neural network menghasilkan sinyal kendali u[k] berdasarkan y_ref dan memverifikasi apakah plant menghasilkan keluaran y[k] yang mendekati referensi y_ref. Data validasi juga digunakan untuk mengevaluasi kinerja model dan mencatat kerugian serta akurasi, yang divisualisasikan dalam grafik untuk memantau tren pelatihan dan validasi. Setelah pelatihan selesai, model digunakan untuk menghasilkan sinyal kendali u[k] berdasarkan referensi keluaran baru. Hasil prediksi dibandingkan dengan sinyal kendali aktual untuk memverifikasi keakuratan dan kemampuan model dalam mempelajari pola yang relevan. Grafik perbandingan antara referensi y_ref, keluaran plant y[k], dan sinyal kendali u[k] memberikan gambaran visual tentang kinerja model dalam mengontrol plant.

3.7. Training dan Validasi Tahap 2 (Controller)

Pada tahap Training dan Validasi Tahap 2, program menggunakan kelas MLPVectoral untuk melatih dan memvalidasi inverse controller dengan mekanisme feedback. Neural network ini tidak hanya menerima referensi y_ref, tetapi juga mengintegrasikan keluaran plant sebelumnya sebagai bagian dari input. Input neural network di tahap ini adalah X[k] = (y_ref[k], y[k-1], y[k-2], u[k-1], u[k-2]).

Selama proses pelatihan, referensi keluaran y_ref[k] dan keluaran plant sebelumnya digunakan untuk menghasilkan sinyal kendali u[k], yang kemudian diteruskan ke plant untuk memvalidasi apakah keluaran plant mendekati referensi. Dalam mekanisme ini,

sinyal kendali u[k] dari prediksi sebelumnya juga digunakan sebagai input ke neural network pada langkah berikutnya, menciptakan umpan balik yang realistis dan memungkinkan model menangkap pola temporal.

Proses forward pass dan backward pass serupa dengan tahap pertama, tetapi perhitungan gradien dan pembaruan bobot mempertimbangkan pengaruh feedback. Selama pelatihan, kerugian pada data pelatihan dan validasi dicatat untuk mengevaluasi kinerja model. Visualisasi berupa grafik kerugian dan akurasi membantu memantau overfitting atau underfitting. Tahap ini dirancang untuk meningkatkan kemampuan neural network dalam menangkap pola sekuensial, menjadikannya lebih efektif dalam memprediksi sinyal kendali yang tepat untuk mencapai keluaran yang diinginkan.

3.8. Testing Hasil Identifikasi Sistem (Controller)

Pada tahap ini, model inverse controller yang telah dilatih diuji pada berbagai sinyal referensi y_ref, seperti gelombang sinusoidal, kotak, dan segitiga dengan berbagai parameter (frekuensi dan amplitudo). Sinyal referensi ini digunakan untuk menghasilkan sinyal kendali u[k] melalui neural network, yang kemudian diterapkan pada plant untuk menghasilkan keluaran plant y[k].

Hasil uji divisualisasikan dalam grafik yang membandingkan sinyal referensi y_ref, keluaran plant y[k], dan sinyal kendali u[k] yang dihasilkan oleh neural network. Evaluasi dilakukan untuk melihat seberapa baik model dapat mengendalikan plant sehingga keluaran plant mendekati sinyal referensi dalam berbagai skenario. Tahap ini menilai keakuratan dan kemampuan generalisasi model dalam menerapkan Direct Inverse Control, memastikan model stabil, akurat, dan mampu mengendalikan sistem dengan pola dinamis yang beragam.

3.9. Cascade Inverse Controller dengan Plant Identifikasi Sistem

Pendekatan *Cascade Inverse Controller* dengan plant hasil identifikasi sistem merupakan strategi kontrol yang dirancang untuk meningkatkan performa dan robustness sistem kendali pada plant non-linear dengan dinamika kompleks. Sistem ini mengadopsi konfigurasi pengendalian berlapis, di mana terdapat dua pengendali yang bekerja secara berurutan. Pengendali pertama, atau *controller primer*, berfungsi untuk menghasilkan sinyal referensi berdasarkan perbedaan antara setpoint dan keluaran plant. Sinyal referensi ini kemudian diproses oleh pengendali kedua, yaitu *inverse*

controller, yang bertanggung jawab untuk menghasilkan sinyal masukan u[k]u[k]u[k] yang diperlukan agar keluaran plant dapat mengikuti referensi secara akurat.

Plant yang digunakan dalam sistem ini adalah model plant hasil identifikasi sistem berbasis neural network yang telah dilatih untuk merepresentasikan hubungan input-output plant secara akurat. Pada konfigurasi ini, *inverse controller* juga menggunakan neural network yang telah dilatih untuk memodelkan invers plant. Dengan demikian, *cascade control* memungkinkan sistem untuk menangani karakteristik plant yang kompleks dan non-linear, sekaligus meningkatkan respons sistem terhadap gangguan atau perubahan kondisi dinamis.

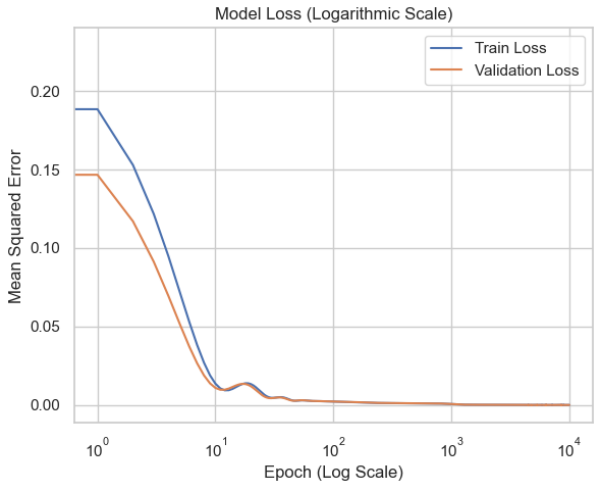
Keunggulan utama dari pendekatan ini terletak pada robustness dan akurasi yang lebih tinggi. Dengan adanya dua lapisan kontrol, sistem mampu meredam gangguan pada tingkat tinggi melalui *controller primer*, sementara *inverse controller* memastikan plant mengikuti referensi secara presisi. Selain itu, pendekatan ini memungkinkan sistem untuk menangkap dinamika non-linear plant, berkat kemampuan neural network dalam mempelajari pola kompleks. Pada tahap implementasi, sistem ini melibatkan beberapa langkah utama, yaitu identifikasi plant menggunakan neural network, desain *inverse controller* untuk memodelkan invers plant, integrasi *controller primer* untuk menghasilkan referensi keluaran, serta simulasi dan evaluasi performa sistem menggunakan berbagai jenis input, seperti *step input* dan *sinusoidal input*.

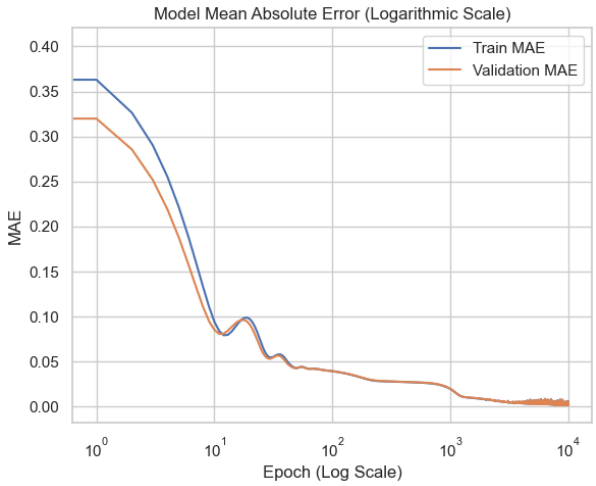
Evaluasi sistem dilakukan berdasarkan metrik kontrol seperti *overshoot*, waktu penyesuaian (*settling time*), kesalahan steady-state, serta kemampuan sistem dalam mengikuti trajektori setpoint dinamis. Hasil pengujian menunjukkan bahwa pendekatan ini mampu memberikan respons yang lebih stabil, adaptif, dan presisi, terutama pada plant dengan karakteristik non-linear yang rumit. Dengan memanfaatkan kombinasi antara *cascade control* dan plant hasil identifikasi sistem berbasis neural network, pendekatan ini memberikan solusi yang efektif untuk aplikasi kendali yang membutuhkan stabilitas, akurasi, dan fleksibilitas tinggi dalam berbagai kondisi operasi.

BAB 4
HASIL DAN ANALISIS

4.1. Sistem Identifikasi Plant (2 Hidden Layer, 64 dan 32 Neuron)

4.1.1. Hasil Tahap 1



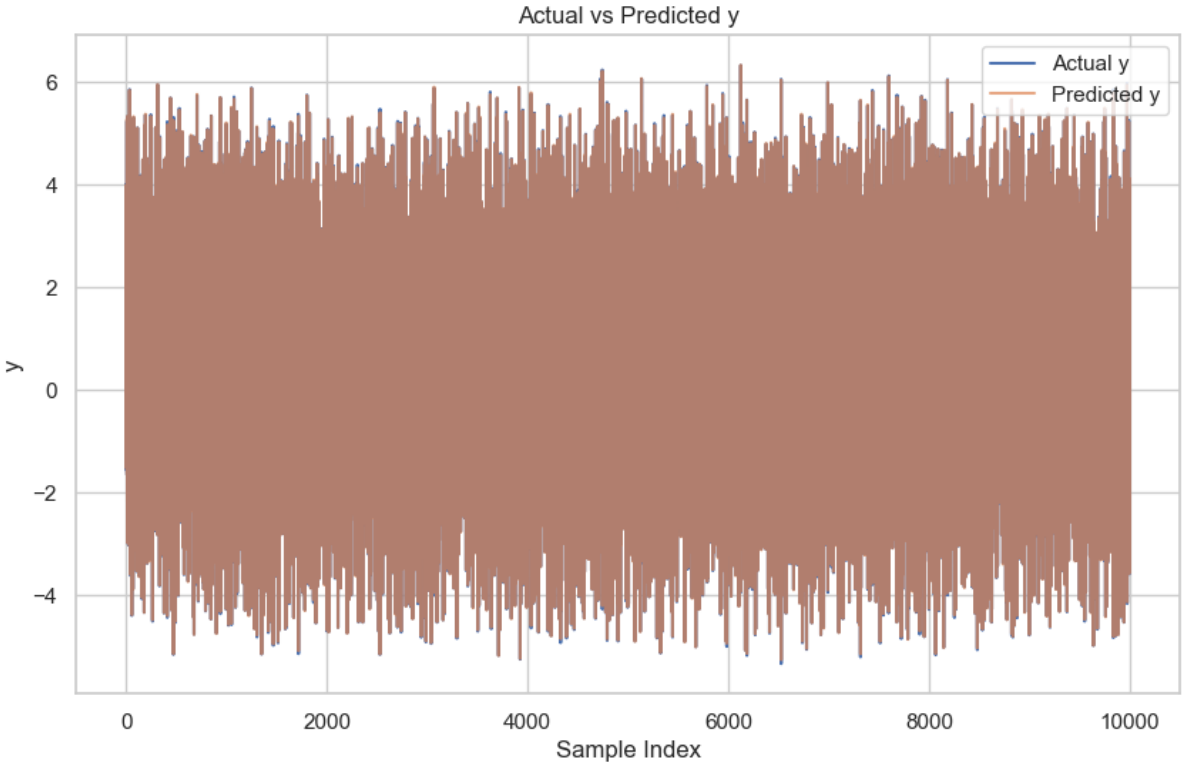


loss: 6.9188e-06

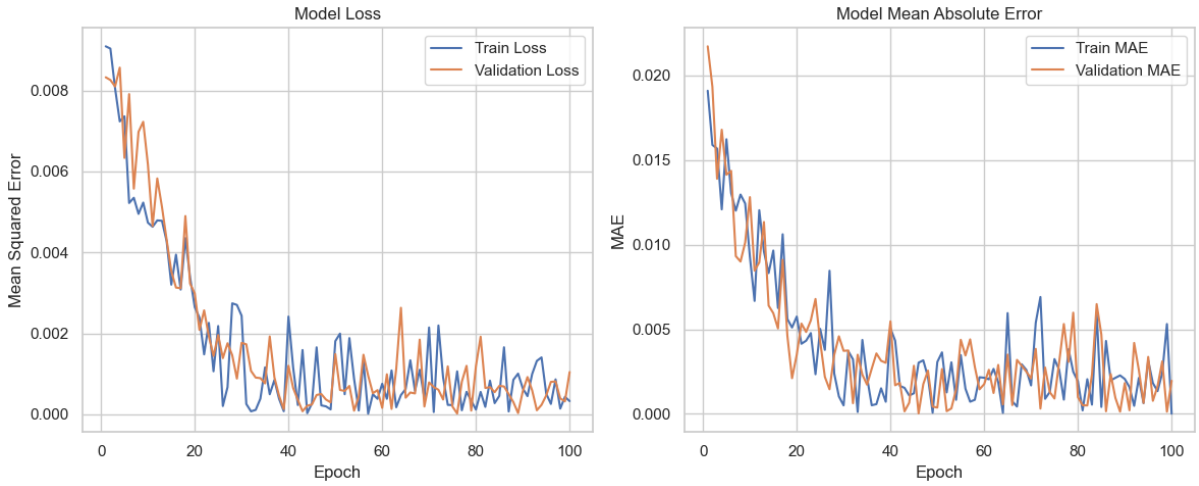
mean_absolute_error: 0.0020

val_loss: 6.5215e-06

val_mean_absolute_error: 0.0020



4.1.2. Hasil Tahap 2

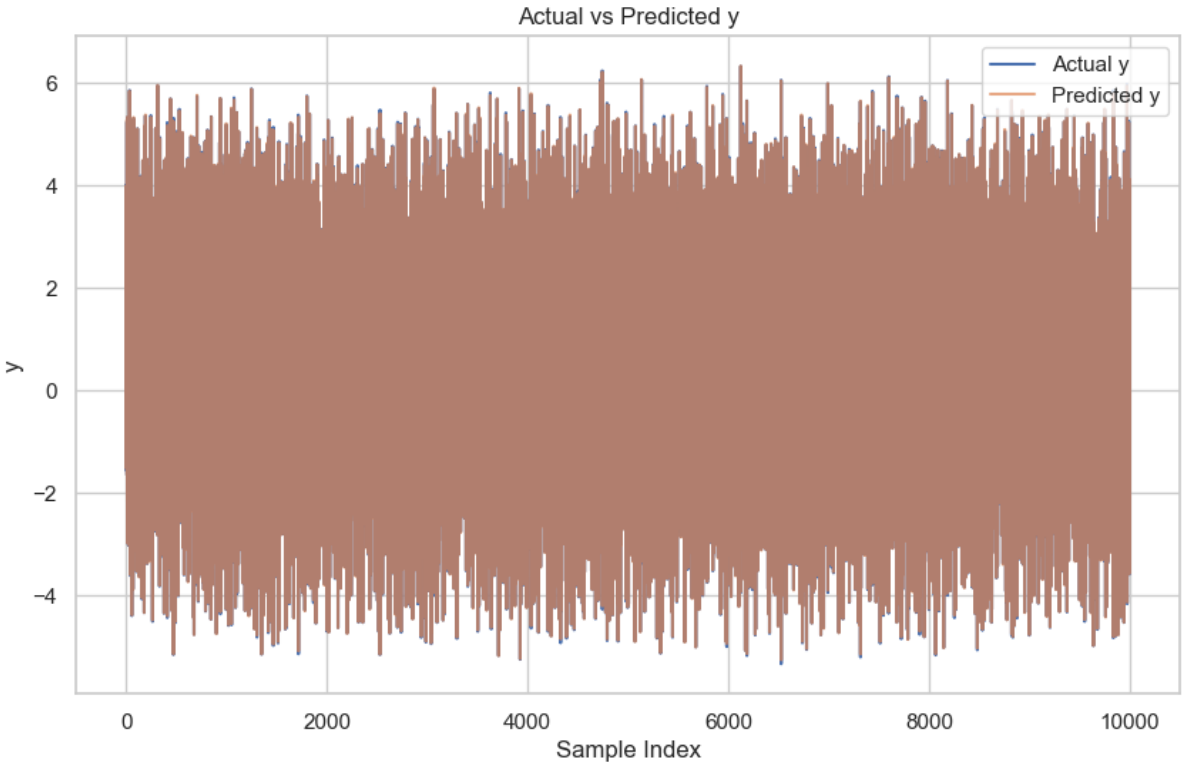


loss: 4.2132e-04

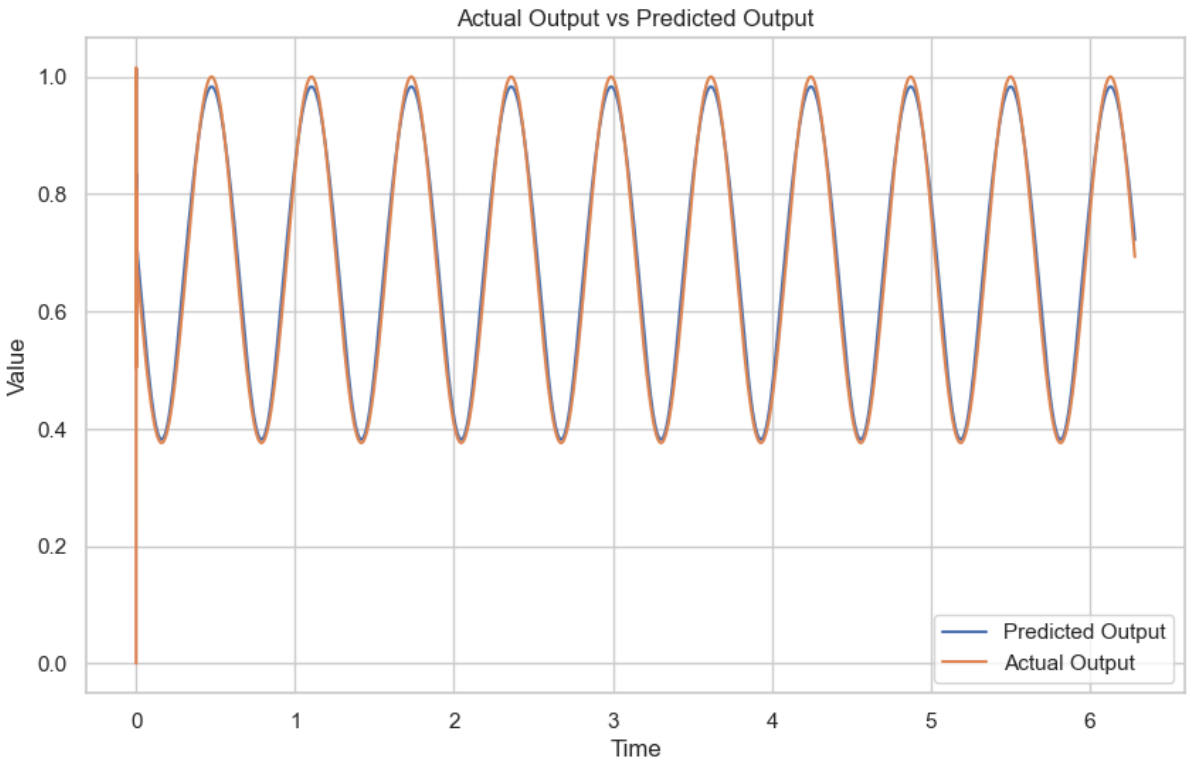
mean_absolute_error: 0.013

val_loss: 4.0216e-04

val_mean_absolute_error: 0.012

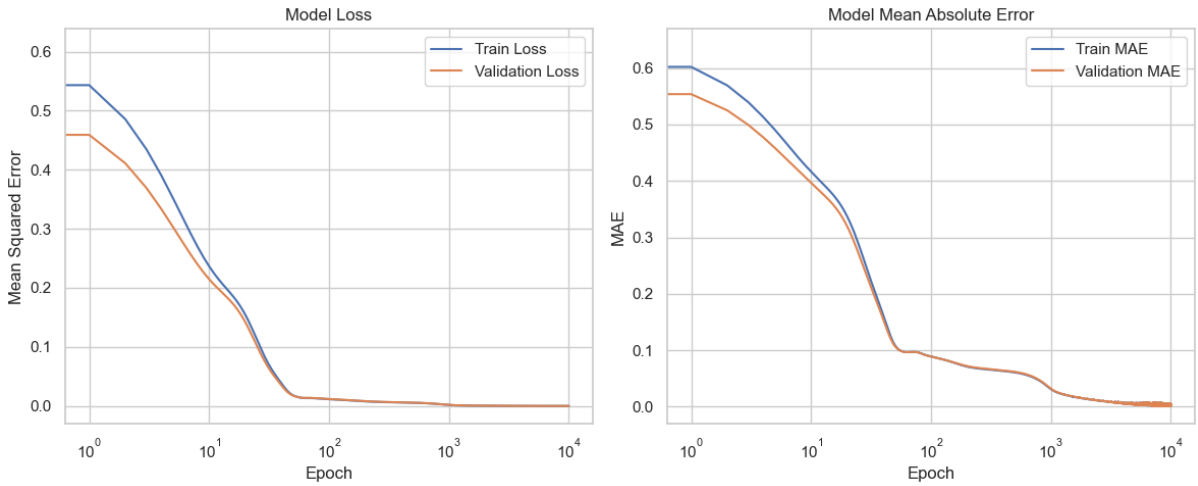


4.1.3. Hasil Testing



4.2. Sistem Identifikasi Controller (2 Hidden Layer, 64 dan 32 Neuron)

4.2.1. Hasil Tahap 1

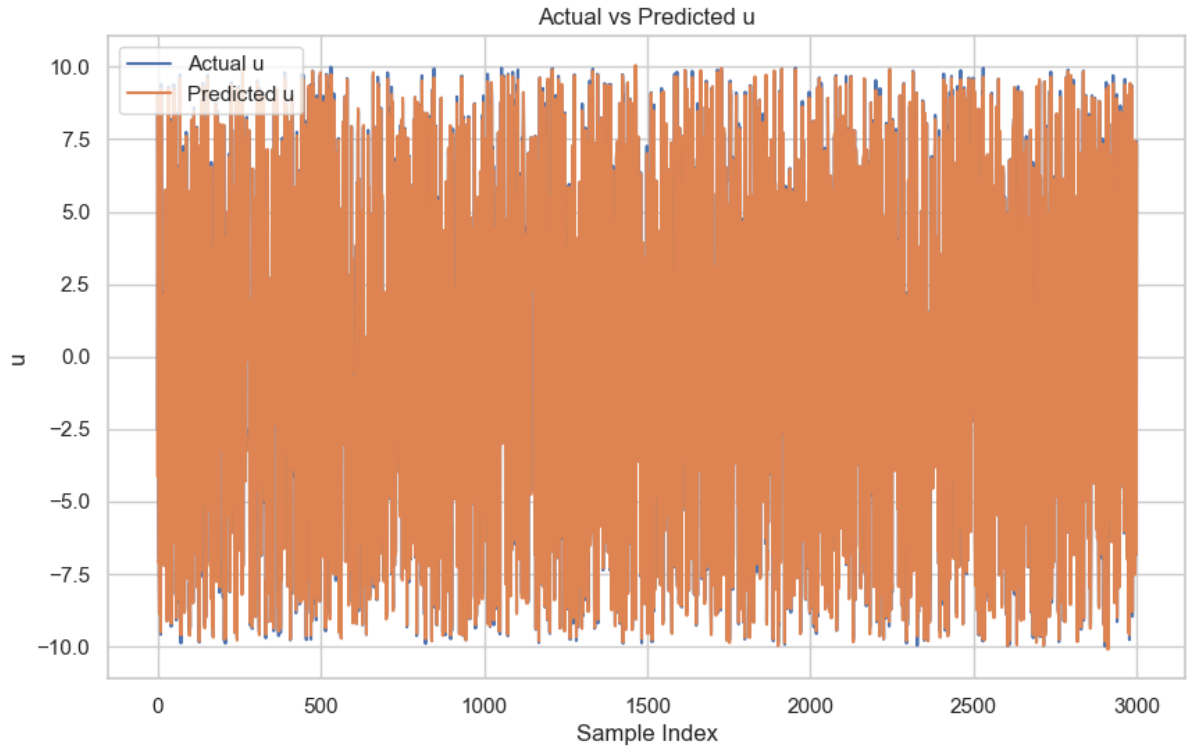


loss: 7.9067e-06

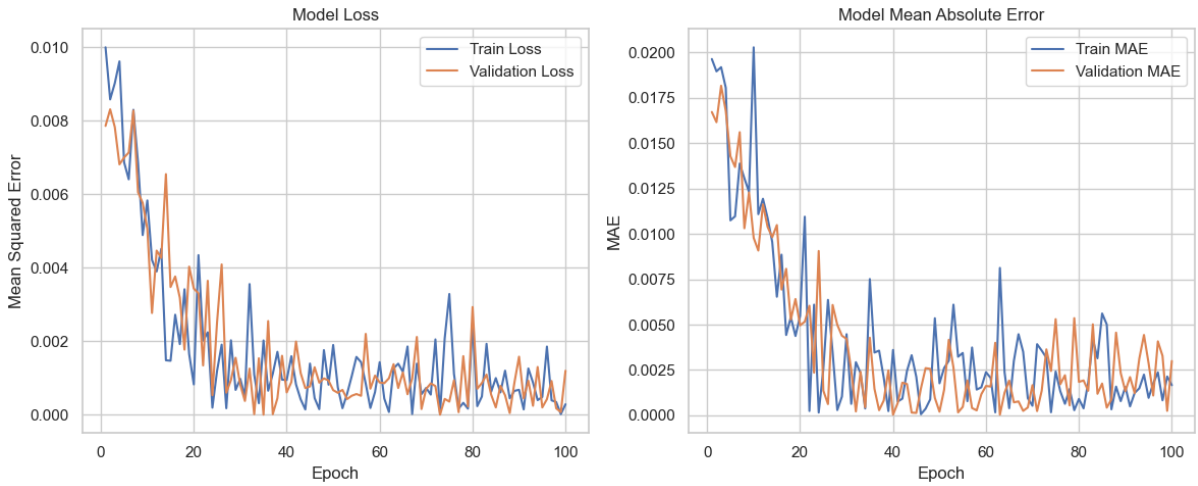
mean_absolute_error: 0.0018

val_loss: 4.5450e-06

val_mean_absolute_error: 0.0017



4.2.2. Hasil Tahap 2

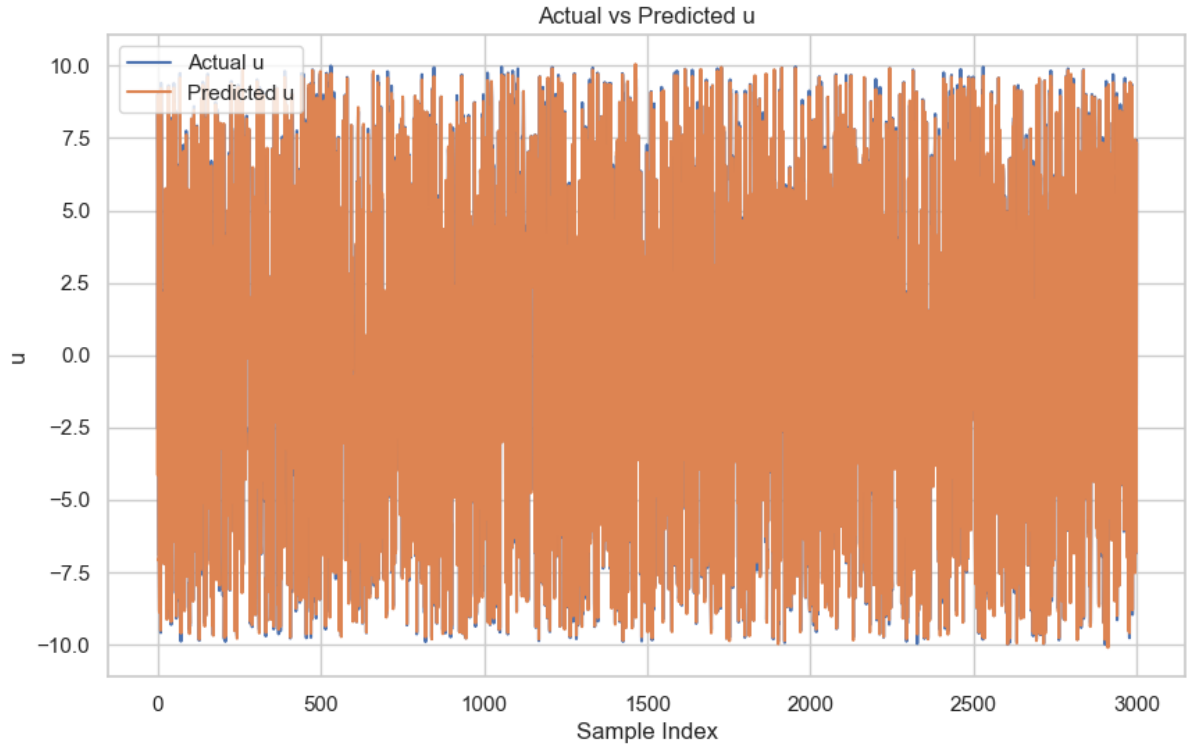


loss: 6.4239e-04

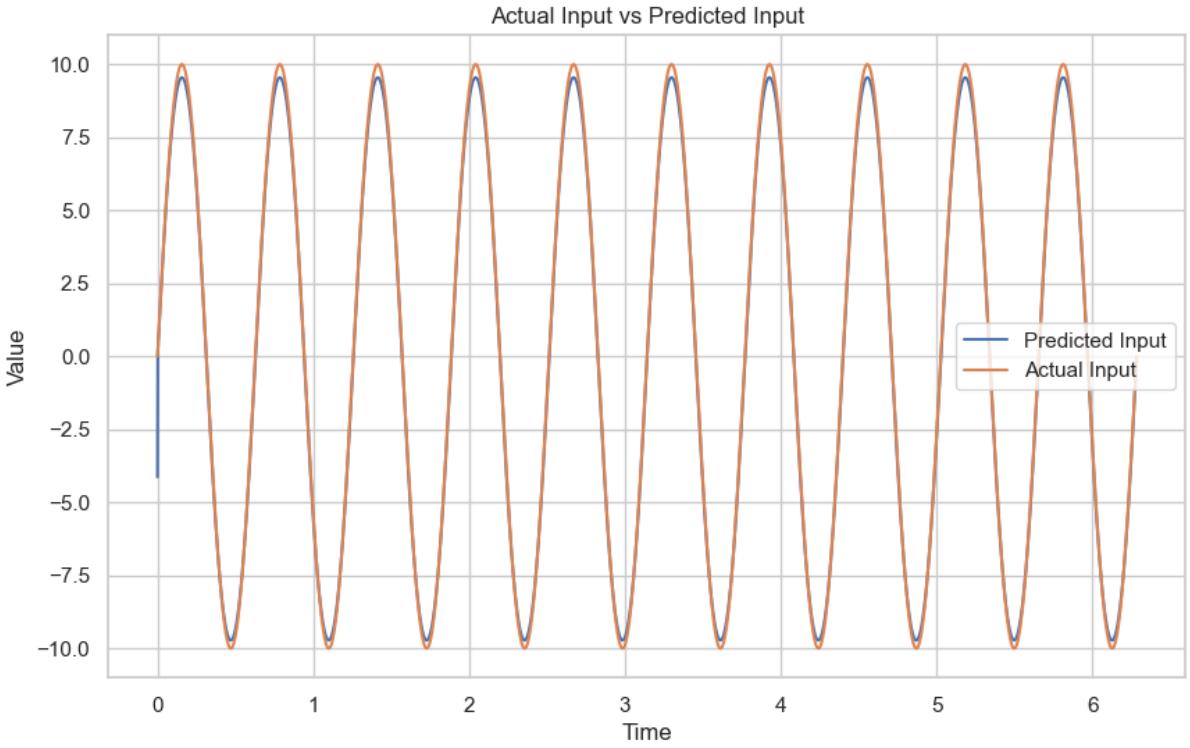
mean_absolute_error: 0.025

val_loss: 5.3367e-04

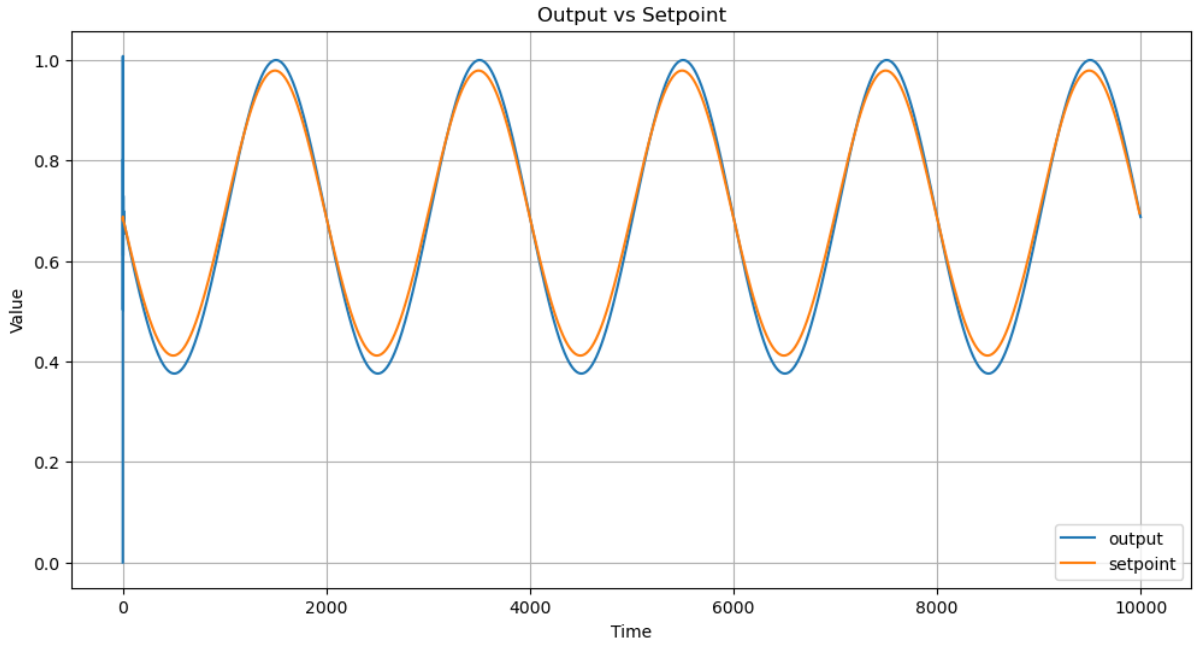
val_mean_absolute_error: 0.021

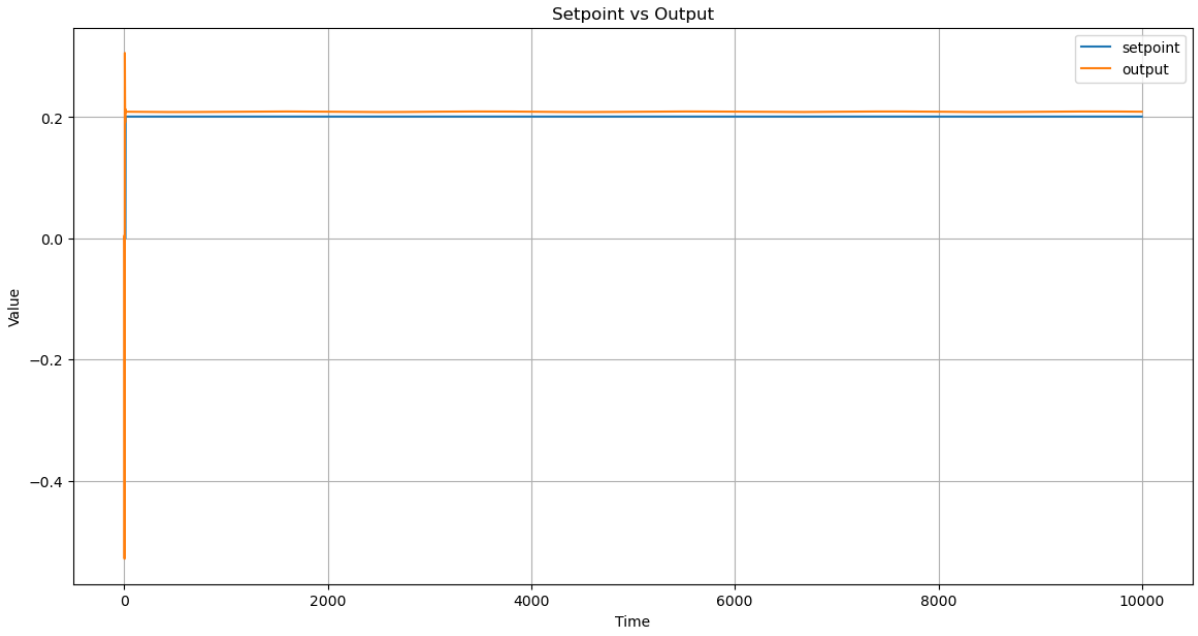


4.2.3. Hasil Testing



4.3. Direct Inverse Closed Loop Control (Gabungan Controller dan Plant)





4.4. Analisis Hasil Eksperimen

Pada sistem identifikasi plant dengan neural network menggunakan dua hidden layer (64 dan 32 neuron), dilakukan dua tahap pelatihan, validasi, dan pengujian. Pada Tahap 1, model neural network dilatih untuk mempelajari hubungan input-output tanpa menggunakan mekanisme *feedback*. Hasil pelatihan menunjukkan nilai kerugian (loss) sebesar 6.9188e-6 dan *mean absolute error* (MAE) sebesar 0.002 untuk data pelatihan, serta nilai kerugian validasi sebesar 6.5215e-6 dengan MAE validasi 0.0020. Grafik *Model Loss* dan *Model MAE* menunjukkan tren konvergensi yang baik antara data

pelatihan dan validasi. Penurunan kerugian yang cepat pada epoch awal hingga mencapai kestabilan menunjukkan bahwa model mampu mempelajari pola hubungan input-output dengan baik tanpa mengalami overfitting.

Pada Tahap 2, model dilatih dengan mekanisme *feedback*, di mana keluaran prediksi sebelumnya digunakan sebagai input untuk mempelajari pola temporal dalam sistem plant. Hasil pelatihan menunjukkan nilai kerugian sebesar 4.2132×10^{-4} dan MAE sebesar 0.013 untuk data pelatihan, serta nilai kerugian validasi sebesar 4.0216e-4 dengan MAE validasi 0.012. Grafik *Model Loss* dan *Model MAE* menunjukkan fluktuasi pada awal pelatihan, namun tren menurun secara keseluruhan hingga mencapai kestabilan. Meskipun nilai kerugian lebih tinggi dibandingkan Tahap 1, mekanisme *feedback* memberikan kemampuan model untuk menangkap pola temporal yang lebih kompleks, sesuai dengan tujuan identifikasi sistem dinamik.

Hasil pengujian dilakukan dengan menggunakan berbagai jenis sinyal masukan, seperti gelombang sinusoidal, kotak, dan segitiga, untuk mengevaluasi kemampuan generalisasi model. Grafik *Actual vs Predicted Output* menunjukkan bahwa model mampu mereplikasi keluaran plant dengan sangat baik, di mana garis prediksi hampir sepenuhnya menutupi garis keluaran aktual. Pada grafik *Actual vs Predicted y* untuk data kompleks, model menunjukkan akurasi tinggi dengan kesesuaian prediksi terhadap nilai aktual, bahkan pada kondisi dinamik yang berubah-ubah.

Model neural network identifikasi sistem plant pada Tahap 1 berhasil mempelajari hubungan input-output langsung dengan akurasi tinggi. Sementara itu, Tahap 2 dengan mekanisme *feedback* meningkatkan kemampuan model untuk menangkap pola temporal dalam sistem dinamik, meskipun terdapat sedikit peningkatan nilai kerugian akibat kompleksitas yang lebih tinggi. Hasil pengujian menunjukkan bahwa model memiliki kemampuan generalisasi yang baik pada berbagai sinyal masukan, membuktikan efektivitas metode identifikasi sistem berbasis neural network ini. Dengan demikian, sistem identifikasi plant berbasis neural network ini mampu merepresentasikan perilaku sistem secara akurat, baik pada data sederhana maupun kompleks.

Pada sistem identifikasi controller menggunakan dua hidden layer dengan masing-masing 64 dan 32 neuron, dilakukan dua tahap pelatihan, validasi, dan pengujian. Pada Tahap 1, model neural network dilatih tanpa mekanisme feedback untuk mempelajari

hubungan input-output langsung. Hasil pelatihan menunjukkan nilai kerugian (loss) sebesar 7.9067e-06 dan mean absolute error (MAE) sebesar 0.0018 untuk data pelatihan, serta nilai kerugian validasi sebesar 4.5450e-06 dengan MAE validasi 0.0017. Grafik Model Loss dan Model MAE menunjukkan tren konvergensi yang baik antara data pelatihan dan validasi, mengindikasikan bahwa model berhasil mempelajari hubungan input-output dengan akurasi tinggi.

Pada Tahap 2, model dilatih dengan menggunakan mekanisme feedback, di mana keluaran prediksi sebelumnya digunakan sebagai input untuk menangkap pola temporal dalam sistem controller. Hasil pelatihan menunjukkan nilai kerugian sebesar 6.4239e-04 dan MAE sebesar 0.025 untuk data pelatihan, sementara nilai kerugian validasi sebesar 5.3367e-04 dengan MAE validasi 0.021. Grafik Model Loss dan Model MAE menunjukkan fluktuasi pada awal pelatihan, tetapi tren menurun secara keseluruhan hingga mencapai kestabilan. Mekanisme feedback memberikan kemampuan model untuk menangkap pola temporal, meskipun menghasilkan nilai kerugian yang lebih besar dibandingkan Tahap 1 akibat peningkatan kompleksitas.

Hasil pengujian menunjukkan bahwa model mampu menghasilkan keluaran kendali yang sesuai dengan nilai aktual pada berbagai sinyal masukan. Grafik Actual vs Predicted u memperlihatkan kesesuaian yang sangat baik, di mana prediksi model hampir sepenuhnya menutupi data aktual. Selain itu, grafik Actual Input vs Predicted Input untuk masukan sinusoidal menunjukkan keselarasan yang sangat baik antara prediksi dan nilai aktual, mencerminkan akurasi tinggi dalam pengujian model.

Sistem identifikasi controller berhasil memodelkan dinamika sistem kendali dengan baik. Pada Tahap 1, model neural network memberikan hasil yang sangat akurat untuk hubungan input-output langsung, sedangkan pada Tahap 2, mekanisme feedback memungkinkan model menangkap pola temporal yang lebih kompleks. Hasil ini menunjukkan bahwa neural network efektif untuk digunakan dalam sistem identifikasi controller, memberikan akurasi tinggi baik pada pelatihan maupun pengujian. Jika diperlukan, analisis tambahan dapat dilakukan untuk memperdalam evaluasi kinerja model.

Pada pengujian *Direct Inverse Closed Loop Control* yang menggabungkan model controller dan plant, dilakukan analisis terhadap respons sistem tertutup untuk membandingkan keluaran sistem (*output response*) dengan nilai referensi (*setpoint*).

Grafik menunjukkan bahwa sistem mampu mengikuti setpoint baik untuk input berupa *step* maupun *sinusoidal*. Namun, terdapat beberapa karakteristik yang perlu dianalisis lebih lanjut, seperti *overshoot*, respons transien, dan kesalahan steady-state.

Pada respons terhadap *step input*, sistem menunjukkan adanya *overshoot* yang terlihat pada fase awal transien, di mana keluaran sistem melebihi nilai setpoint sebelum akhirnya kembali ke nilai referensi. *Overshoot* ini mencerminkan adanya akumulasi energi yang berlebih dalam proses adaptasi awal, tetapi sistem dengan cepat meredam osilasi dan mencapai kestabilan. Respons transien cukup cepat, dengan waktu adaptasi yang singkat sebelum sistem memasuki kondisi steady-state. Setelah mencapai steady-state, perbedaan antara keluaran dan setpoint menjadi sangat kecil, menunjukkan kesalahan steady-state yang hampir nol.

Untuk respons terhadap *sinusoidal input*, sistem mampu mengikuti perubahan setpoint dengan cukup baik. Namun, terdapat *overshoot* yang muncul pada saat perubahan trajektori di ujung-ujung pola sinusoidal, yaitu ketika setpoint berubah secara signifikan. Hal ini dapat terjadi karena sistem membutuhkan waktu untuk menyesuaikan diri terhadap percepatan atau perlambatan pada perubahan sudut sinusoidal. Meskipun demikian, *overshoot* ini relatif kecil dan segera teredam, sehingga keluaran tetap berada di sekitar nilai setpoint secara keseluruhan.

Response sistem kendali menunjukkan performa yang baik dalam skenario *step* maupun *sinusoidal input*. *Overshoot* yang terjadi di awal transien pada *step input* dan pada perubahan trajektori *sinusoidal input* merupakan bagian alami dari proses adaptasi sistem kendali. Respons steady-state menunjukkan akurasi tinggi, dengan kesalahan steady-state yang hampir nol. Sistem juga tidak menunjukkan osilasi berlebihan atau ketidakstabilan, mencerminkan desain kontrol yang efektif. Kombinasi controller dan plant dalam sistem tertutup ini mampu menghasilkan performa yang stabil, akurat, dan adaptif terhadap berbagai jenis input, menjadikannya sesuai untuk aplikasi kendali dinamis yang kompleks.

BAB 5
KESIMPULAN

5.1. Kesimpulan

5.1.1. Neural network berhasil mengidentifikasi sistem dinamik diskrit secara efektif. Model dapat mempelajari pola input-output dengan baik, terutama pada sinyal yang sederhana dan kontinu seperti gelombang sinus.

5.1.2. Neural network mampu memahami hubungan input-output pada sistem dengan umpan balik. Eksperimen di Tahap 2 menunjukkan bahwa mekanisme umpan balik membantu meningkatkan akurasi prediksi, terutama untuk pola dinamis yang kompleks.

5.1.3. Gabungan model controller dan plant dalam sistem tertutup menunjukkan performa kendali yang stabil dan akurat. Sistem mampu mengikuti setpoint dengan baik, meskipun terdapat overshoot pada fase transien untuk sinyal *step* dan saat perubahan trajektori pada ujung setpoint sinusoidal. Kesalahan steady-state sangat kecil, mencerminkan akurasi kendali yang tinggi dalam berbagai skenario input.

5.1.4. Neural network efektif dalam memodelkan sistem non-linear dan kompleks, terbukti dari kemampuannya memprediksi sinyal baik plant ataupun controller.

5.2. Saran

5.2.1. Optimalisasi Arsitektur Model: Perlu dilakukan eksplorasi lebih lanjut terhadap konfigurasi arsitektur neural network, seperti jumlah lapisan dan neuron, untuk menemukan keseimbangan optimal antara kompleksitas dan kemampuan generalisasi. Penggunaan teknik seperti dropout atau batch normalization dapat membantu mengurangi overfitting.

5.2.2. Pengaturan Hiperparameter yang Lebih Baik: Disarankan untuk mencoba teknik pengaturan laju pembelajaran dinamis (learning rate schedules) atau algoritma optimasi lainnya, seperti RMSProp atau SGD dengan momentum, untuk meningkatkan efisiensi pelatihan dan stabilitas model. Penggunaan early stopping juga dapat mencegah pelatihan berlebihan (overfitting).

5.2.3. Eksplorasi Teknik Regularisasi: Untuk mengatasi overfitting, penggunaan regularisasi yang lebih kuat, seperti L1 regularisasi atau dropout, dapat dieksplorasi. Hal ini akan membantu model untuk lebih fokus pada pola yang relevan dan menghindari belajar dari noise dalam data.

5.2.4. Pengujian dengan Data yang Lebih Beragam: Perluasan jenis sinyal uji, termasuk sinyal yang lebih kompleks atau non-periodik, akan memberikan pemahaman lebih mendalam tentang kemampuan model untuk melakukan generalisasi. Hal ini juga dapat membantu mengidentifikasi keterbatasan model dalam memodelkan pola dinamis yang lebih rumit.

5.2.5. Penggunaan Teknik Feedback yang Lebih Canggih: Mempertimbangkan penggunaan teknik feedback yang lebih canggih, seperti Long Short-Term Memory (LSTM) atau Recurrent Neural Networks (RNNs), dapat meningkatkan kemampuan model dalam menangani pola dinamis yang bergantung pada nilai sebelumnya. Ini akan sangat berguna untuk aplikasi di mana ketergantungan temporal sangat kuat.

5.2.6. Pengembangan Sistem Kendali Adaptif: Disarankan untuk mengembangkan pengendali berbasis neural network yang adaptif. Dengan memanfaatkan hasil identifikasi sistem, neural network dapat digunakan untuk mengatur parameter kontrol secara otomatis sesuai kondisi sistem. Pendekatan ini berpotensi meningkatkan performa sistem secara keseluruhan, terutama pada aplikasi dinamis yang membutuhkan respons prediktif dan adaptif yang cepat.

DAFTAR PUSTAKA

S. Haykin, "Neural Networks: A Comprehensive Foundation," 2nd ed., Prentice Hall, 1999.

T. M. Mitchell, "Machine Learning," McGraw Hill, 1997.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," Nature, vol. 323, no. 6088, pp. 533-536, Oct. 1986.

Y. LeCun, L. Bottou, G. B. Orr, and K. R. Müller, "Efficient BackProp," in Neural Networks: Tricks of the Trade, Springer, 2012, pp. 9-48.

I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," MIT Press, 2016.

D. Nguyen and B. Widrow, "Improving the Learning Speed of 2-Layer Neural Networks by Choosing Initial Values of the Adaptive Weights," in Proceedings of the IJCNN, 1990, vol. 3, pp. 21-26.

S. Ruder, "An overview of gradient descent optimization algorithms," arXiv preprint arXiv:1609.04747, 2016.

A. Ng, "Machine Learning Yearning," Deeplearning.ai, 2019.

K. S. Narendra and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks," IEEE Transactions on Neural Networks, vol. 1, no. 1, pp. 4-27, 1990.