ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – TM184835

# OPTIMASI PARAMETER PENGENDALI PID PADA SISTEM SUSPENSI AKTIF MODEL SEPEREMPAT KENDARAAN DENGAN METODE BACK PROPAGATION NEURAL NETWORK DAN GENETIC ALGORITHM

**DANIEL MICHAEL RADJA PANDE**
NRP. 02111840000098

Dosen Pembimbing
**Mohammad Khoirul Effendi, S.T., M.Sc.Eng., Ph.D.**
NIP. 198204142010121001

**Program Studi Teknik Mesin**
Departemen Teknik Mesin
Fakultas Teknologi Industri dan Rekayasa Sistem
Institut Teknologi Sepuluh Nopember
Surabaya
2022

(*Halaman ini sengaja dikosongkan*)

**TUGAS AKHIR – TM184835**

# OPTIMASI PARAMETER PENGENDALI PID PADA SISTEM SUSPENSI AKTIF MODEL SEPEREMPAT KENDARAAN DENGAN METODE BACK PROPAGATION NEURAL NETWORK DAN GENETIC ALGORITHM

**DANIEL MICHAEL RADJA PANDE**
**NRP 02111840000098**

Dosen Pembimbing
**Mohammad Khoirul Effendi, S.T., M.Sc.Eng., Ph.D.**
NIP. 198204142010121001

**Program Studi Teknik Mesin**
Departemen Teknik Mesin
Fakultas Teknologi Industri dan Rekayasa Sistem
Institut Teknologi Sepuluh Nopember
Surabaya
2022

(*Halaman ini sengaja dikosongkan*)

**FINAL PROJECT – TM184835**

# PID CONTROLLER PARAMETERS OPTIMIZATION IN ACTIVE SUSPENSION SYSTEM QUARTER VEHICLE MODEL WITH BACK PROPAGATION NEURAL NETWORK AND GENETIC ALGORITHM METHODS

**DANIEL MICHAEL RADJA PANDE**
NRP 02111840000098

Advisor
**Mohammad Khoirul Effendi, S.T., M.Sc.Eng., Ph.D.**
NIP. 198204142010121001

**Study Program of Mechanical Engineeering**
Department of Mechanical engineering
Faculty of Industrial Technology and Systems Engineering
Institut Teknologi Sepuluh Nopember
Surabaya
2022

(*This page is intentionally blank*)

## LEMBAR PENGESAHAN

**OPTIMASI PARAMETER PENGENDALI PID PADA SISTEM SUSPENSI AKTIF MODEL SEPEREMPAT KENDARAAN DENGAN METODE *BACK PROPAGATION NEURAL NETWORK* DAN *GENETIC ALGORITHM***

**TUGAS AKHIR**

Diajukan untuk memenuhi salah satu syarat
memperoleh gelar Sarjana Teknik pada
Program Studi S-1 Teknik Mesin
Departemen Teknik Mesin
Fakultas Teknologi Industri dan Rekayasa Sistem
Institut Teknologi Sepuluh Nopember

Oleh : **DANIEL MICHAEL RADJA PANDE**
NRP. 02111840000098

Disetujui oleh Tim Penguji Tugas Akhir :

1. Mohammad Khoirul Effendi, S.T., M.Sc.Eng., Ph.D. — Pembimbing
   NIP. 198204142010121001
2. Dinny Harnany, S.T., M.Sc. — Penguji I
   NIP. 198905132019032013
3. Ari Kurniawan Saputra, S.T., M.T. — Penguji II
   NIP. 198604012015041001

**SURABAYA**
**Juli, 2022**

(*Halaman ini sengaja dikosongkan*)

## APPROVAL SHEET

**PID CONTROLLER PARAMETERS OPTIMIZATION IN ACTIVE SUSPENSION SYSTEM QUARTER VEHICLE MODEL WITH BACK PROPAGATION NEURAL NETWORK AND GENETIC ALGORITHM METHODS**

**FINAL PROJECT**
Submitted to fulfill one of the requirements
for obtaining a degree of bachelor of engineering at
Undergraduate Study Program of Mechanical Engineering
Departement of Mechanical Engineering
Faculty of Industrial Technology and Systems Engineering
Institut Teknologi Sepuluh Nopember

By : **DANIEL MICHAEL RADJA PANDE**
NRP. 02111840000098

Approved by Final Project Examiner Team :

1. Mohammad Khoirul Effendi, S.T., M.Sc.Eng., Ph.D. — Advisor
   NIP. 198204142010121001
2. Dinny Harnany, S.T., M.Sc. — Examiner I
   NIP. 198905132019032013
3. Ari Kurniawan Saputra, S.T., M.T. — Examiner II
   NIP. 198604012015041001

**SURABAYA**
**July, 2022**

(*This page is intentionally blank*)

## PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini:

| | |
|---|---|
| Nama mahasiswa/NRP | : Daniel Michael Radja Pande /02111840000098 |
| Departemen | : Teknik Mesin |
| Dosen Pembimbing/NIP | : Mohammad Khoirul Effendi, S.T., M.Sc.Eng., Ph.D. /198204142010121001 |

Dengan ini menyatakan bahwa Tugas Akhir dengan judul "OPTIMASI PARAMETER PENGENDALI PID PADA SISTEM SUSPENSI AKTIF MODEL SEPEREMPAT KENDARAAN DENGAN METODE *BACK PROPAGATION NEURAL NETWORK* DAN *GENETIC ALGORITHM*" adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, 1 Agustus 2022

| Mengetahui<br>Dosen Pembimbing | Mahasiswa |
|---|---|
|  |  |
| (Mohammad Khoirul Effendi, S.T., M.Sc.Eng., Ph.D.) | (Daniel Michael Radja Pande) |
| NIP. 198204142010121001 | NRP. 02111840000098 |

(*Halaman ini sengaja dikosongkan*)

## STATEMENT OF ORIGINALITY

The undersigned below:

| | |
|---|---|
| Name of student/NRP | : Daniel Michael Radja Pande /02111840000098 |
| Departement | : Mechanical Engineering |
| Advisor/NIP | : Mohammad Khoirul Effendi, S.T., M.Sc.Eng., Ph.D. /198204142010121001 |

Hereby declare that the Final Project with the title of "PID CONTROLLER PARAMETERS OPTIMIZATION IN ACTIVE SUSPENSION SYSTEM QUARTER VEHICLE MODEL WITH BACK PROPAGATION NEURAL NETWORK AND GENETIC ALGORITHM METHODS" is the result of my own work, is original, and is written by following the rules of scientific writing.

If in the future there is a discrepancy with this statement, then I am willing to accept sanctions in accordance with the provisions that apply at Institut Teknologi Sepuluh Nopember.

Surabaya, August 1st 2022

Acknowledged
Advisor

(Mohammad Khoirul Effendi, S.T., M.Sc.Eng., Ph.D.)
NIP. 198204142010121001

Student

(Daniel Michael Radja Pande)
NRP. 02111840000098

(*This page is intentionally blank*)

# OPTIMASI PARAMETER $K_P$, $K_I$, DAN $K_D$ PADA KONTROLER PID SISTEM SUSPENSI AKTIF MODEL SEPEREMPAT KENDARAAN DENGAN METODE BACK PROPAGATION NEURAL NETWORK-GENETIC ALGORITHM

**Nama Mahasiswa : Daniel Michael Radja Pande**
**NRP : 02111840000098**
**Departemen : Teknik Mesin FTI-RS**
**Dosen Pembimbing : Mohammad Khoirul Effendi, S.T., M.Sc.Eng., Ph.D.**

## ABSTRAK

Sistem suspensi merupakan sistem yang berfungsi untuk meredam getaran yang mempengaruhi tingkat kenyamanan penumpang saat dihadapkan pada medan jalanan yang memiliki kontur yang tidak konstan. *Active Suspension System* mampu menghasilkan kualitas berkendara yang lebih baik dibandingkan dengan sistem suspensi pasif. Dalam suspensi aktif diperlukan suatu *controller* yang berfungsi untuk mengontrol gaya yang diberikan sehingga dapat meredam getaran yang dihasilkan oleh kontur jalan yang tidak konstan. Pengendali PID adalah salah satu jenis pengendali yang digunakan untuk mengontrol gaya tersebut. Dalam penelitian ini akan dilakukan *tuning* Pengendali PID menggunakan metode metaheuristik *Back Propagation Neural Network-Genetic Algorithm* (BPNN-GA).

Penelitian ini akan mengacu dan dibandingkan dengan parameter dan hasil pada *paper "Stochastic Algorithm for PID Tuning Of Bus Suspension System"* oleh A.Karthikraja, dkk. (2009). Pada penelitian ini akan dilakukan optimasi arsitektur BPNN yang paling baik untuk melakukan prediksi nilai ITAE yang akan digunakan pada proses optimasi parameter pengendali PID yaitu nilai konstanta proporsional, konstanta integral, dan konstanta derivatif ($K_P$, $K_I$, dan $K_D$). Optimasi parameter pengendali PID dilakukan menggunakan perangkat lunak MATLAB. Parameter *input* dari BPNN adalah nilai $K_P$, $K_I$, dan $K_D$, dan parameter *output* adalah nilai *Integral Times Absolute Error* (ITAE). Optimasi dengan *Genetic Algorithm* akan menggunakan *objective function* berupa *net* BPNN yang mengeluarkan prediksi nilai ITAE, yang akan dicari nilai minimumnya *(smaller is better).* Data yang digunakan dalam pembentukan *net* BPNN didapatkan melalui simulasi MATLAB. Pada penelitian ini akan dievaluasi performa optimasi berdasarkan nilai $K_P$, $K_I$, dan $K_D$ terbaik yang akan disimulasikan dan dilihat nilai *Settling Time* dan *Peak Overshoot*-nya.

Dalam penelitian ini pemodelan model seperempat kendaraan dapat dilakukan di *software* MATLAB. Hasil optimasi model arsitektur BPNN dengan GA untuk mendapatkan nilai MSE *(Mean Squared Error)* terkecil adalah arsitektur dengan 5 (lima) *hidden layer*, 10 (sepuluh) *neuron* di setiap *hidden layer*-nya, dan fungsi aktivasi satlin dengan nilai MSE *training* $1.6477 \times 10^{-8}$.

Didapatkan nilai $K_P$, $K_I$, dan $K_D$ paling optimum dengan metode BPNN-GA yaitu 809193, 621978, dan 243984, dengan nilai *Settling Time* ($T_S$) 1,33 s, dan *Peak Overshoot* -0,00834 m. Dapat disimpulkan performa dari BPNN-GA mendekati metode *Real Coded GA* dan *Particle Swarm Optimization* (PSO) untuk parameter *settling time* dan lebih baik untuk parameter *peak overshoot*, apabila dibandingkan dengan metode yang digunakan pada penelitian yang berjudul *"Stochastic Algorithm for PID Tuning Of Bus Suspension System*" oleh A. Karthikraja, dkk. (2009).

***Kata kunci: sistem suspensi aktif (active suspension system), quarter car model, Pengendali PID, neural network, genetic algorithm, backpropagation neural network-genetic algorithm***

# PID CONTROLLER PARAMETERS OPTIMIZATION IN ACTIVE SUSPENSION SYSTEM QUARTER VEHICLE MODEL WITH BACK PROPAGATION NEURAL NETWORK AND GENETIC ALGORITHM METHODS


**Student Name** : **Daniel Michael Radja Pande**
**Student ID** : **02111840000098**
**Departement** : **Mechanical Engineering FTI-RS**
**Advisor** : **Mohammad Khoirul Effendi, S.T., M.Sc.Eng., Ph.D.**


## ABSTRACT


The suspension system is a system that functions to dampen vibrations that affect the comfort level of passengers when faced with inconsistent road contours. The Active Suspension System can produce a better driving quality compared to the passive suspension system. In active suspension, a controller functions to control the applied force. PID controller is one of controller used to control the force. In this study, PID controller tuning will be carried out using the Back Propagation Neural Network-Genetic Algorithm (BPNN-GA) metaheuristic method.

This study will refer to paper titled "Stochastic Algorithm for PID Tuning of Bus Suspension System" by A. Karthikraja, et al. (2009). In this study, the best BPNN architecture optimization will be used to predict the ITAE value that will be used in the PID controller parameter optimization process, namely the value of proportional constants, integral constants, and derivative constants ($K_P$, $K_I$, and $K_D$). PID controller parameter optimization is done using MATLAB software. The input parameters from BPNN are the values of $K_P$, $K_I$, and $K_D$, and the output parameters are the values of Integral Times Absolute Error (ITAE). Optimization with Genetic Algorithm will use an objective function in the form of net BPNN which issues a prediction of the ITAE value, which will be sought for the minimum value (smaller is better). The data used for BPNN network were obtained through MATLAB simulations. In this study, the optimization performance will be evaluated based on the best $K_P$, $K_I$, and $K_D$ values that will be simulated and see the settling time and peak overshoot values.

In this study the modeling of a quarter vehicle model can be done in MATLAB software. Best architecture optimization result is with 5 (five) hidden layers, 10 (ten) neurons in each hidden layer, and a satlin activation function with an MSE value of $1{,}6477\times10^{8}$.

The best $K_P$, $K_I$, and $K_D$ parameters were obtained using the BPNN-GA method, namely 809193, 621978, and 243984, with a settling time of 1.33s, and Peak Overshoot -0.00834m. The performance of BPNN-GA approaches the Real Coded GA and Particle Swarm Optimization (PSO) methods for the settling time parameter and is better for the peak overshoot parameter, when compared to the method used in the study entitled “Stochastic Algorithm for PID Tuning Of Bus Suspension. System” by A. Karthikraja, et al. (2009).



***Keywords: active suspension system, quarter car model, PID controller, neural network, genetic algorithm, backpropagation neural network-genetic algorithm***

(*This page is intentionally blank*)

## KATA PENGANTAR

Segala puji dan syukur kepada Tuhan Yang Maha Esa, atas berkah dan izin-Nya tugas akhir ini dapat terselesaikan. Penulis sangat menyadari bahwa keberhasilan dalam penyelesaian tugas akhir ini tidak terlepas dari dukungan dan bantuan berbagai pihak. Melalui kesempatan yang baik ini, penulis ingin menyampaikan ucapan terima kasih dan penghargaan yang setinggi-tingginya kepada pihak-pihak yang telah banyak membantu dan mendukung baik secara moril maupun materiil dalam proses penyelesaian Tugas Akhir ini, antara lain:

1. Bapak dan Mama serta kakak dan adikku tersayang yang senantiasa memberi dukungan dan doa hingga penulis dapat menyelesaikan Tugas Akhir ini.
2. Bapak M. Khoirul Effendi, S.T., M.Sc.Eng., Ph.D selaku dosen pembimbing Tugas Akhir penulis yang selalu memberikan saran, arahan, dorongan, motivasi, dan ilmunya yang sangat bermanfaat kepada penulis. Terima kasih atas dedikasi dan waktu yang telah diberikan.
3. Ibu Dinny Harnany, S.T., M.Sc. dan Bapak Ari Kurniawan Saputra, S.T., M.T. selaku dosen penguji Tugas Akhir penulis. Terima kasih atas waktu dan saran-saran yang telah diberikan.
4. Seluruh Bapak dan Ibu Dosen Teknik Mesin ITS yang telah mendidik dan menempa penulis dalam mempelajari ilmu teknik mesin dan juga ilmu kehidupan yang sangat luas ini.
5. Tommy Fu sebagai partner mengerjakan Tugas Akhir yang saling mengingatkan deadline, teman begadang, serta memberikan saran dan masukan didalam perjuangan menyelesaikan tugas akhir ini.
6. Gede, Jimmy, Omar, Frans, Japar, Tommy dan Agung sahabat-sahabat penulis yang sangat supportif menemani di kala suka dan duka
7. Keluarga besar Manajemen dan Volunteer ITS Global Engagement yang telah menempa diri saya menjadi lebih disiplin dan menjadi manusia yang lebih baik.
8. Keluarga M61 dan SMRM yang telah menempa mental penulis, memberi pengalaman dan pembelajaran berharga kepada penulis selama berkuliah di Teknik Mesin ITS yang sama-sama kita cintai.
9. Seluruh civitas akademika ITS yang baik dan ramah.
10. Seluruh pihak yang tidak dapat penulis sebutkan satu persatu.

Penulis menyadari masih banyak kekurangan dalam penyusunan tugas akhir ini, oleh karena itu saran dan masukan dari semua pihak sangat penulis harapkan. Penulis berharap semoga tugas akhir ini dapat memberikan manfaat dan sumbangsih bagi perkembangan ilmu pengetahuan

Surabaya, Agustus 2022

Penulis

(*Halaman ini sengaja dikosongkan*)

# DAFTAR ISI

(*Halaman ini sengaja dikosongkan*)

## DAFTAR GAMBAR

# DAFTAR TABEL

(*Halaman ini sengaja dikosongkan*)

# BAB I
# PENDAHULUAN

## 1.1. Latar Belakang Masalah

Suspensi kendaraan memiliki fungsi vital dalam kenyamanan berkendara. Vibrasi yang berlebihan pada kendaraan dapat menyebabkan gangguan kenyamanan pada penumpang dan pengemudi pada kendaraan (Sezgin & Arslan, 2012) sehingga dapat mengganggu performa dari pengemudi (Azizan et al., 2017). Hal ini tentu saja tidak diinginkan oleh para konsumen kendaraan bermotor. Dalam kenyataannya kendaraan mengalami pola vibrasi yang acak dan tidak dapat diprediksi saat bergerak di permukaan jalan. Maka dari itu dibutuhkan suatu sistem yang berfungsi sebagai peredam getaran kendaraan yang diakibatkan oleh profil jalan. Sistem peredam getaran kendaraan atau suspensi pada umumnya menggunakan sistem pegas dan *damper*. Perkembangan dari industri otomotif mendorong lebih jauh lagi riset pada sistem suspensi otomotif.

Dalam perkembangannya, sistem suspensi kendaraan terbagi menjadi tiga yaitu sistem suspensi pasif, semi aktif, dan aktif. Sistem suspensi pasif menggunakan sistem pegas dan *damper* konvensional tanpa adanya elemen aktif. Dengan berkembangnya teknologi otomotif dikembangkanlah sistem suspensi semi aktif dan aktif. Sistem suspensi semi aktif adalah suspensi yang menggunakan *damper* dengan konstanta *damping* yang bisa divariasikan. Sedangkan sistem suspensi aktif adalah suspensi yang menggunakan komponen suspensi pasif, namun pada sistemnya ditambahkan aktuator yang memberikan gaya ke sistem dengan harapan dapat meredam getaran dengan lebih baik dan cepat.

Dalam perkembangan sistem suspensi aktif terdapat permasalahan untuk merancang sistem kendali yang mengatur gaya aktuator yang diberikan ke sistem, terutama dalam kasus ini adalah pengendali PID. Dalam mendesain suatu sistem kendali PID, secara umum dilakukan oleh seorang ahli yang familier dengan sistem. Proses *tuning* akan dilakukan dengan mengatur parameter PID melalui proses *trial-and-error* dan melihat apakah respons dari sistem sudah sesuai dengan kriteria yang diinginkan. Secara umum proses *tuning* pengendali PID menggunakan metode analitis seperti *Ziegler-Nichols*, metode ini adalah penurunan model matematika berdasarkan dari sistem yang ingin di kendalikan, kemudian dari representasi model matematika tersebut akan dilakukan proses *tuning*. Tetapi pada kenyataannya dikarenakan tingginya tingkat kompleksitas dari sistem, proses *tuning* menggunakan metode klasik menjadi sangat sulit, lama dan membutuhkan banyak proses *trial and error*. Metode klasik juga seringkali menunjukkan hasil yang tidak optimal, sehingga masih menimbulkan *error* yang cukup signifikan. Hal ini dikarenakan tidak ada rumusan pasti yang menghubungkan antara nilai $K_P$, $K_I$ dan $K_D$ dengan parameter performa dari pengendali yang berlaku universal untuk seluruh sistem. Apabila terjadi perubahan dari sistem dikarenakan kerusakan dari sistem ataupun perubahan dari lingkungan, parameter PID juga harus diubah secara dinamis untuk tetap mendapatkan nilai yang baik. Oleh sebab itu proses *tuning* pengendali PID dengan menggunakan *Artificial Intelligence* menjadi salah satu metode untuk melakukan proses *tuning* untuk mendapatkan parameter sistem kendali yang baik. Dikarenakan tidak adanya hubungan dan pengaruh pasti antara nilai $K_P$, $K_I$ dan $K_D$ dengan performa dari sistem, maka *Artificial Intelligence* dapat menjadi suatu solusi untuk mencari hubungan antar parameter ini. *Artificial Intelligence* sangat baik untuk menyelesaikan suatu kasus dengan mempelajari hubungan antara parameter *input* dan *output* dari sistem.

Pada penelitian ini akan dilakukan optimasi sistem kendali PID untuk sistem suspensi aktif dengan menggunakan metode metahueristik *Back Propagation Neural Network-Genetic Algorithm*. Algoritma akan mencari nilai *gain* dari komponen $K_P$, $K_I$, dan $K_D$ yang paling optimal dengan meminimalisasi parameter ITAE sehingga mendapatkan performa pengendali yang terbaik yaitu nilai *Settling Time* ($T_s$), dan *Peak Overshoot* yang kecil. Nilai ITAE akan di prediksi oleh *net* BPNN. Penelitian performa dari sistem kendali akan disimulasikan dengan perangkat lunak MATLAB R2021.

## 1.2. Perumusan Masalah

Perumusan masalah pada penelitian ini adalah sebagai berikut :

1. Bagaimana merancang dan mensimulasikan proses kendali PID pada sistem suspensi aktif menggunakan *software* MATLAB?
2. Bagaimana menentukan parameter arsitektur BPNN yang paling baik sehingga BPNN mampu memodelkan data yang di-*generate* oleh MATLAB dengan MSE sekecil mungkin.
3. Bagaimana tingkat kenyamanan dari kendaraan dengan sistem suspensi aktif menggunakan pengendali PID yang telah dioptimasi dengan metode BPNN-GA dibandingkan dengan parameter kenyamanan standar?
4. Bagaimana perbandingan performa proses *tuning* menggunakan metode BPNN-GA dengan metode lain dari *paper* "*Stochastic Algorithm for PID Tuning Of Bus Suspension System*" oleh A.Karthikraja, dkk. (2009).

## 1.3. Batasan Masalah

Batasan masalah yang digunakan pada penelitian ini adalah sebagai berikut :

1. Pemodelan suspensi kendaraan menggunakan model seperempat kendaraan dengan dua *Degree of Freedom*, persamaan diasumsikan sebagai persamaan dinamis yang dilinearkan dengan acuan pada titik ekuilibrium dan kecepatan kendaraan konstan.
2. Parameter penelitian diambil dari paper *"Stochastic Algorithm for PID Tuning Of Bus Suspension System"* oleh A.Karthikraja, dkk. (2009).
3. Data *training* dan *validating* BPNN didapatkan melalui simulasi pada *software* MATLAB R2021b dengan batasan yang ditentukan.
4. Variasi arsitektur BPNN yang ingin dioptimasi oleh *Genetic Algorithm* adalah sebagai berikut, *hidden layer* yang divariasikan pada jaringan BPNN adalah 1 sampai 5 *hidden layer* dan tiap *hidden layer* terdiri dari 2 sampai 10 *neuron*, dan jenis fungsi aktivasi yang divariasikan adalah *hardlim, hardlims, purelin, satlin, logsig,* dan *tansig* yang sama pada setiap *node* di tiap *hidden layer*. Dengan jumlah *Generation* 1000.
5. Metode *learning* yang digunakan adalah *Levenberg–Marquardt*.
6. Pada *Genetic Algorithm* pencarian nilai optimum ditentukan pada batasan tertentu, dilakukan *selection* dengan metode *roulette-wheel*, *crossover* dengan metode *uniform crossover*, dan terdapat mutasi dengan probabilitas 0,08.
7. Parameter optimasi parameter $K_P$, $K_I$ , dan $K_D$ yang digunakan adalah nilai *Integral Timed Absolute Error* (ITAE).
8. Performa pengendali yang akan dibandingkan adalah nilai $T_S$ *(settling time)* dan *Peak Overshoot*.
9. Performa kenyamanan dari kendaraan akan dievaluasi berdasarkan standar ISO 2631

10. Penelitian yang dilakukan terbatas pada tahap simulasi dan belum sampai pada tahap validasi secara eksperimental.

## 1.4. Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut :

1. Merancang dan mensimulasikan proses kendali PID pada sistem suspensi aktif menggunakan *software* MATLAB
2. Menentukan parameter BPNN yang paling baik sehingga BPNN mampu memodelkan data yang di-*generate* oleh *software* MATLAB dengan MSE *(Mean Squared Error)* sekecil mungkin.
3. Mengetahui tingkat kenyamanan dari kendaraan dengan sistem suspensi aktif dengan menggunakan pengendali PID yang telah dioptimasi dengan metode BPNN-GA dibandingkan dengan parameter kenyamanan standar.
4. Mengetahui perbandingan performa proses *tuning* menggunakan metode BPNN-GA dengan metode lain dari paper *"Stochastic Algorithm for PID Tuning Of Bus Suspension System"* oleh A.Karthikraja, dkk. (2009)".

## 1.5. Manfaat Penelitian

Manfaat yang dapat diperoleh dari tugas akhir ini adalah sebagai berikut :

1. Mendapatkan arsitektur *Back Propagation Neural Network* yang paling optimal untuk mendapatkan nilai MSE yang paling kecil.
2. Mendapatkan nilai prediksi $K_P$, $K_I$, dan $K_D$ yang optimal pada sistem suspensi aktif.
3. Mendapatkan program *tuner* PID yang berbasis BPNN-GA.
4. Sebagai referensi untuk penelitian yang sejenis dalam rangka pengembangan pengetahuan metode *tuning* pengendali PID.

(*Halaman ini sengaja dikosongkan*)

# BAB II
# DASAR TEORI

## 2.1. Kajian Pustaka

*Artificial Intelligence* atau Kecerdasan Buatan adalah teknologi yang dapat digunakan untuk proses *tuning* Pengendali PID. Penelitian mengenai Pengendali PID yang menggunakan teknologi AI terbagi menjadi dua. Pertama mengganti sepenuhnya Pengendali PID dengan menggunakan AI, dan kedua melakukan otomasi proses *tuning* dari Pengendali PID (Lee & Jang, 2021). Beberapa Industri masih memilih untuk menggunakan pendekatan *tuning* Pengendali PID, terutama pada sistem konservatif yang mengutamakan keamanan. Terdapat banyak metode *tuning* PID, salah satu metode klasik adalah metode Ziegler-Nichols, namun sering kali metode sulit untuk mendapatkan parameter PID yang paling optimal (Chiha et al., 2012). Oleh karena itu telah dikembangkan banyak metode lain yang lebih baik dan cepat untuk melakukan *tuning* parameter Pengendali PID, diantaranya menggunakan metode seperti *fuzzy logic, neural network, neural-fuzzy logic, immune algorithm, simulated annealing,* dan *pattern recognition*. Metode *tuning* PID yang berdasarkan pada metode *Genetic Algorithm*, seperti yang telah dilakukan oleh (Yusoff et al., 2015), pada penelitiannya digunakan optimasi nilai $K_P$, $K_I$ dan $K_D$ menggunakan *transfer function* dari kolom distilasi dengan menggunakan GA dengan jumlah populasi 20, 40, 60, dan 80, hasilnya kemudian dibandingkan dengan metode Ziegler-Nichols. Metode ini sukses mendapatkan nilai *rise time* dan *settling time* yang lebih baik, dan didapatkan jumlah populasi 40 adalah jumlah populasi yang paling optimal.

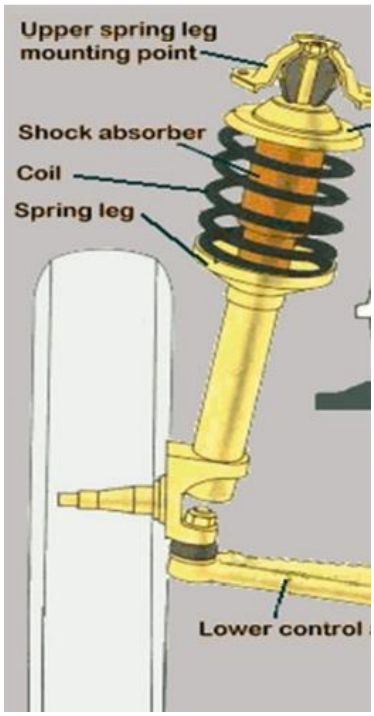
Metode GA juga telah dilakukan oleh (Yusuf et al., 2017) Pengendali PID-GA mampu meredam gangguan pada sistem pengaturan level air *steam drum*, level air dijaga pada titik tengah ketinggian drum atau *Normally Water Level*, agar uap yang dihasilkan memenuhi spesifikasi serta tidak merusak peralatan pada pembangkit. PID-Ga sukses meredam gangguan berupa beban minimal nominal, dan maksimal, yaitu dengan *perturbation peak*. Penelitian lain juga telah dilakukan pada aplikasi pada kontrol tempratur pada *electrothermal boiler* (Gu & Wang, 2012), dimana performa dari kontrol PID yang berbasis BPNN adalah solusi yang efektif, selain melakukan pengaturan terhadap parameter secara *online* tergantung pada *error function*, tetapi juga meningkatkan performa dan adaptabilitas terhadap model matematika berdasarkan hasil simulasi. Pada penelitian lain(Marino & Neri, 2019), telah dilakukan investigasi awal bagaimana arsitektur *Neural Network* dapat menjadi teknologi yang efisien untuk memodelkan Pengendali PID, Neural Network merupakan estimator fungsi yang sangat baik, dalam penelitian ini kemudian diinvestigasi apakah NN dapat membantu menentukan parameter Pengendali PID, didapatkan bahwa PID-NN melakukan proses *tuning* dengan lebih cepat dibandingkan dengan metode klasik yaitu Ziegler Nichols. Dalam penelitian lain Pengendali PID yang berbasis Neural Network diaplikasikan pada ROV *(Remotely Underwater Vehicle)* bawah laut, dibandingkan dengan Pengendali PID dapat disimpulkan bahwa Pengendali PID yang berbasis Neural Network memiliku performa terbaik dengan penggunaan energi yang lebih sedikit. (Hernández-Alvarado et al., 2016). Dalam penelitian kali ini akan digunakan pengaplikasian BPNN bersamaan dengan GA, dimana fungsi yang dikeluarkan oleh BPNN akan dioptimasi oleh GA, penggunaan *Neural Network* bersamaan dengan *Genetic Algorithm* telah dilakukan oleh (Pongfai & Assawinchaichote, 2017) untuk mengontrol *Brush DC* motor, dengan mengevaluasi *maximum overshoot, SSE, rise time, dan settling time*, NN-GA memiliki performa yang lebih baik dibandingkan dengan penggunaan GA murni dan NN murni.

Perkembangan dari sistem suspensi kendaraan memiliki tujuan untuk meningkatkan kenyamanan berkendara penumpang. Getaran pada kendaraan akan diusahakan pada level yang seminimal mungkin. Dalam sistem konvensional digunakan sistem suspensi pasif yaitu menggunakan sistem pegas dan *damper*. Seiring berkembangnya dunia otomotif dan keilmuan teknik mesin, dikembangkanlah sistem suspensi semi-aktif dan aktif. Sistem suspensi semi aktif telah didiskusikan dan telah dapat disimulasikan pada penelitian sebelumnya, suspensi semi-aktif adalah sistem suspensi yang prinsipnya sama dengan sistem suspensi pasif, namun pada komponen *damper-nya* dapat dimodulasi tergantung pada kebutuhan (Vinayak S. Dixit*, 2017). Sedangkan sistem suspensi aktif, adalah sistem suspensi pasif yang dilengkapi dengan aktuator yang memberikan gaya tertentu pada sistem suspensi, besar dan arah gaya di kontrol oleh berbagai jenis pengendali, beberapa jenis pengendali yang diteliti pada penelitian sebelumnya diantaranya, H∞, fuzzy, dan LQR telah dibandingkan masing-masing pengendali performanya, yang secara keseluruhan apabila dibandingkan dengan pengendali pasif memiliki keunggulan di setiap parameter (Kaleemullah et al., 2011). Salah satu pengendali yang juga digunakan pada sistem suspensi aktif adalah PID yang performanya didapatkan lebih baik dari suspensi pasif (Fayyad, 2012; Sharkawy et al., 2015).

Pengendali PID dalam praktiknya dapat diaplikasikan ke berbagai jenis sistem, namun pengendali PID harus melalui proses *tuning*, yaitu menentukan nilai $K_P$, $K_I$ dan $K_D$. Parameter-parameter tersebut bergantung pada model dan gangguan atau *disruption* pada sistem, sehingga mendapatkan performa terbaik pengendali. .Aplikasi pengendali PID dengan optimasi metahueristik pada sistem suspensi aktif kendaraan telah dilakukan pada penelitian-penelitian terdahulu, telah dilakukan penelitian mengenai aplikasi pengendali PID yang telah di optimasi dengan Neural Network untuk memprediksi nilai menentukan *gain parameter* dari pengendali PID (Müderrisoğlu et al., 2016). Input yang di masukkan adalah kecepatan, persentase *overshoot*, *settling time*, dan *steady state error* dari respons sistem suspensi, output yang di keluarkan adalah *gain parameter* dari PID yaitu nilai $K_P$, $K_I$ dan $K_D$, kemudian performanya, yaitu nilai, *sprung mass displacement, body acceleration, suspension deflection, tire deflection dan Power Spectral Density (PSD)*, pengendali dibandingkan dengan pengendali PID tanpa optimasi Neural Network dan suspensi pasif, didapatkan pengendali PID yang telah dioptimasi memiliki performa yang jauh lebih baik dari semua variasi (Kalaivani & Lakshmi, 2016a). Proses *tuning* otomatis juga telah dilakukan penelitiannya dengan menggunakan Neural Network, Neural Network akan melakukan identifikasi pada sistem berdasarkan dari data respons, dari parameter PID sebelumnya, kemudian Neural Network kedua akan memprediksi parameter PID berdasarkan Neural Network pertama, penelitian menunjukkan *tuner* dapat merekomendasikan parameter PID dalam 2 detik, efisiensi 92,9% dan proses *tuning* selesai dengan rata-rata pada 2,94 percobaan (Lee & Jang, 2021). Pada penelitian lainnya dilakukan perbandingan terhadap beberapa fungsi aktivasi pada Neural Network dan didapatkan bahwa fungsi terakurat dengan nilai regresi 99% dan MSE 4,2% pada fungsi *newelm* dan *training method* tercepat adalah metode *training* Levenberg-Marquardt apabila dibandingkan dengan metode *training* yang lain, dengan jumlah *node* dan *transfer function* tertentu (Heidari & Homaei, 2013). Penelitian lain yang dilakukan adalah membandingkan performa suspensi pasif dengan suspensi yang menggunakan kontrol BPNN-GA langsung, penelitian ini juga menunjukkan bahwa performa pengendali BPNN-GA memiliki performa yang lebih baik dibanding suspensi pasif (Tang et al., 2009). Dalam melakukan metode penelitian dilakukan dengan melakukan simulasi pada aplikasi Simulink pada MATLAB, dapat dimodelkan model suspensi dari kendaraan menggunakan model seperempat kendaraan.(Tandel et al., 2014).

Penelitian kali ini mengacu pada penelitian *"Stochastic Algorithm for PID Tuning Of Bus Suspension System"* oleh A.Karthikraja, dkk. (2009) pada penelitian dalam proses *tuning* Pengendali suspensi aktif pada bus (Arumugam et al., 2009) dalam metode ini digunakan model seperempat kendaraan, *input ramp* sebesar 0,1 m, diambil sebagai *input* pengendali dan parameter tingkat kenyamanan dan *handling* dari kendaraan adalah *output* dari pengendali, penelitian ini juga melakukan penelitian pada metode *Particle Swarm Optimization* (PSO). Penelitian ini memiliki tujuan untuk memperkecil osilasi dan meningkatkan kenyamanan kepada penumpang. Hasil dari penelitian ini dibandingkan dengan pengendali PID yang di-*tuning* dengan metode *Zeigler-Nichols*, didapatkan metode GA dan PSO memiliki performa dalam mencapai *settling time* lebih cepat dan *peak overshoot* yang lebih kecil. Pada penelitian dilakukan *tuning* pengendali PID menggunakan metode-metode seperti dijelaskan sebelumnya, penelitian ini bertujuan untuk mencari nilai parameter pengendali PID terbaik untuk meminimalkan *settling time* dan *peak overshoot*, metode ini berhasil mendapatkan nilai optimum dari pengendali PID, berdasarkan hasil simulasi, metode tuning ini memili performa yang lebih baik dibandingkan metode klasik, GA, dan *ant system* (Chiha et al., 2012). Metode ini akan dibandingkan dengan metode BPNN-GA pada penelitian ini dikarenakan metode BPNN memiliku mampu memprediksi persamaan polinomial orde tinggi dengan cara mengatur jumlah neuron dan jumlah layernya. Ketika BPNN sudah mendapatkan fungsi yang merepresentasikan hubungan antara parameter *input* dan responsnya, GA akan mencari nilai paling optimal dari seluruh *range* nilai yang ada pada BPNN untuk mendapatkan nilai yang terbaik. Hipotesa awal dari penelitian ini adalah metode BPNN-GA mampu memberikan parameter Pengendali PID yang lebih optimum, sehingga memberikan respons yang lebih baik dan memberikan nilai parameter lebih cepat dibandingkan metode terbaik pada *paper*.

## 2.2. Suspensi

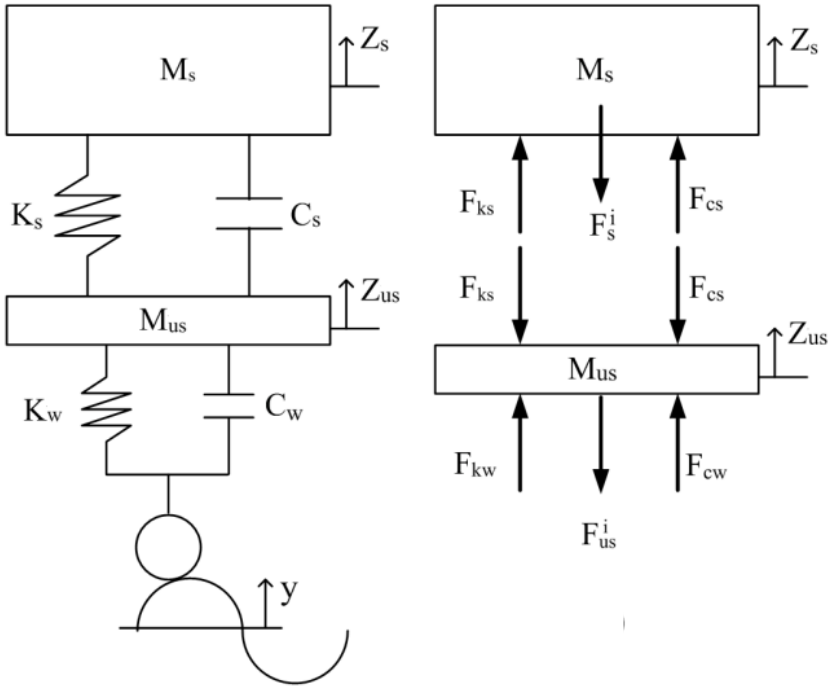



Gambar 2.1 Sistem Suspensi Pada Kendaraan (Kashem et al., 2018)

Sistem suspensi adalah elemen esensial dalam kendaraan untuk mengisolasi *frame* kendaraan dari gangguan yang disebabkan oleh pola jalan yang acak. Pada Gambar 2.1. Dapat dilihat contoh dari suspensi kendaraan pada umumnya. Komponen paling penting dari sistem suspensi adalah *damper*. *Damper* mengurangi efek dari gangguan yang tidak dapat diprediksi pada jalan, dengan cara melembutkan *shock* atau kejutan yang disebabkan oleh pola tersebut. *Damper* disebut juga sebagai *shock breaker* atau *shock absorber*. Pada *shock absorber* pada umumnya, energi getaran akan dikonversikan menjadi panas melalui suatu cairan *viscous*. Dalam *hydraulic cylinder*, fluida di dalam silinder memanas. Dalam *air cylinder*, udara panas di lepaskan ke atmosfer. Pada *damper* elektromagnetik, energi getaran dikonversi menjadi listrik melalui suatu motor listrik dan disimpan dalam baterai untuk penggunaan lanjut.(Kashem et al., 2018)

Suspensi kendaraan dikategorikan menjadi pasif, aktif dan semi-aktif yang dikelompokkan berdasarkan tingkat *controllability* masing-masing suspensi. Walaupun semua tipe dari sistem suspensi memiliki keuntungan dan kerugian masing-masing, semua tipe suspensi memanfaatkan komponen pegas dan *damper*. Performa dari sistem suspensi dievaluasi berdasarkan kualitas berkendara. Dua variabel yang pada umumnya dijadikan parameter untuk mendesain dan mengevaluasi sistem suspensi adalah *vehicle body acceleration*, yang menentukan kenyamanan berkendara, dan defleksi suspensi, yang mengindikasikan gerakan maksimum dari kendaraan saat diberi gangguan. Dalam penelitian sebelumnya *mean square* dari akselerasi vertikal dari kendaraan diambil sebagai kriteria performa (fungsi objektif) yang akan di optimasi.(Pekgökgöz et al., 2010)

### 2.2.1. Suspensi Pasif




Gambar 2.2 *Free Body Diagram* Sistem Suspensi Pasif (Hendrowati, dkk., 2012)

Sistem suspensi pasif adalah tersusun dari komponen pegas dan *damper* dengan properti komponen yang konstan. Dalam model ini $m_1$ dan $m_2$ mewakili *unsprung mass* dan *sprung mass*, sedangkan $k_s$ adalah konstanta kekakuan dari roda, sedangkan $k_w$ adalah konstanta kekakuan dari pegas suspensi. $c_s$ dan $c_w$ adalah konstanta damping dari roda dan suspensi. y, $z_s$ dan $z_{us}$ merepresentasikan

profil *input* dari jalan, perpindahan dari *unsprung mass* dan perpindahan dari *sprung mass*.

Suspensi pasif pada umumnya lebih sederhana, lebih andal, dan lebih murah dibandingkan tiga jenis suspensi lainnya (Kashem et al., 2018). Karakteristik konstanta pasif yang paling terlihat adalah konstanta *damping* suspensi yang konstan. Untuk konstanta pasif, biasanya digunakan pegas yang lembut dan *damper* dengan konstanta damping yang rendah untuk aplikasi pada kendaraan yang mengutamakan kenyamanan, dan biasanya pada kendaraan mewah. *Free body diagram* dari sistem suspensi pasif dapat dilihat pada gambar 2.2.

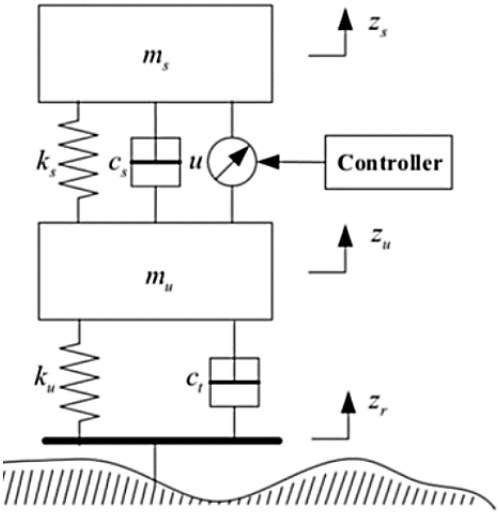
Pada kasus kendaraan mobil sport, akan menggunakan pegas yang keras dan *damper* dengan konstanta *damping* yang besar, untuk meningkatkan stabilitas dan kontrol. Hal inilah yang menjadi pertimbangan dalam mendesain suatu sistem suspensi pasif dengan masing-masing *trade-off* antara dua faktor yaitu kenyamanan berkendara dan stabilitas atau *controllability* dari kendaraan. Hal ini menjadi suatu yang harus disesuaikan dan dipertimbangkan dalam mendesain suatu sistem suspensi pasif, dikarenakan nilai properti dari suspensi pasif yang konstan.

### 2.2.2. Suspensi Semi-Aktif

Sistem suspensi semi aktif, adalah sistem suspensi pasif, namun pada properti dari *damper* bisa dikontrol sesuai dengan kebutuhan(Kalaivani & Lakshmi, 2016b). Suspensi ini menggunakan beberapa teknologi, diantaranya cairan *electrorheological* (ER) dan *magnetorheogical* (MR), katup solenoid dan aktuator *piezoelectric*.(Kashem et al., 2018). Suspensi semi-aktif memiliki performa yang lebih baik dibandingkan suspensi pasif. Dikarenakan suspensi ini aman, ekonomis dan tidak membutuhkan daya yang besar.

Tetapi, masih banyak tantangan yang haris dicapai agar teknologi ini mencapai potensi tertingginya. Pada MR, teknologi ini akan terdegradasi seiring dengan waktu, terdapatnya masalah pada *sealing*, dan sensitivitas pada temperatur menjadi hal yang krusial dalam *damper* dengan teknologi MR sehingga masih membutuhkan perkembangan yang lebih lanjut.

### 2.2.3. Suspensi Aktif



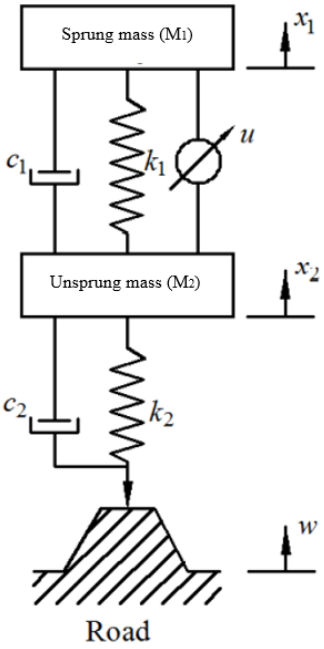

Gambar 2.3 *Free Body Diagram* Sistem Suspensi Aktif (Gaikwat,Ugale, 2020)

Sistem suspensi aktif adalah sistem suspensi pasif dengan tambahan komponen aktuator yang menarik dan mendorong sistem sambungan dari sistem. Secara umum suspensi aktif adalah jawaban dari tiga faktor yang ingin diseimbangkan dalam suspensi pasif, yaitu *road handling, suspension travel,* dan kenyamanan dari penumpang. Keuntungan dari sistem ini adalah walaupun sistem aktuator sistem suspensi aktif mengalami kegagalan, komponen pasif masih ada untuk membantu menjalankan fungsi dari suspensi (Mouleeswaran, 2012). *Free body diagram* dari sistem suspensi aktif dapat dilihat pada gambar 2.3.

## 2.3. Model satu-per-empat kendaraan (*quarter car model)*

Model satu per-empat-kendaraan telah banyak digunakan untuk mengevaluasi performa dari pengendali pada suspensi aktif (Kashem et al., 2018). Model seperempat kendaraan menggambarkan gerakan vertikal dari kendaraan. *Sprung mass*, komponen suspensi, dan *unsprung mass* dan roda adalah komponen dasar dari model satu-per-empat kendaraan. *Sprung Mass* adalah *body* dari kendaraan, model ini merepresentasikan hampir ¼ berat dari seluruh kendaraan. Sistem suspensi menghubungkan antara roda dan kendaraan dari kendaraan dan terdiri dari banyak komponen yang tergantung dari sistem suspensi yang digunakan, seperti sistem semi aktif, aktif, maupun pasif. *Unsprung mass* adalah segala sesuatu yang berada dibawah sistem suspensi, seperti *shaft*, roda, dan *velg*. Roda memberikan perwakilan dari karakteristik pegas dan *damper* dari roda.

Model satu-per-empat kendaraan memiliki 2 derajat kebebasan, sistem ini ditunjukkan oleh Gambar 2.4.




Gambar 2.4 Model Satu Per Empat Kendaraan (Pekgökgöz et al., 2010)

Model satu-per-empat kendaraan pada gambar 2.4. digunakan untuk mensimulasikan model untuk sistem pengendali nanti. Persamaan dinamis dari sistem ini dapat dilihat pada persamaan 2.1 dan 2.2

$$m_b\ddot{x}_1 = -c_1(\dot{x}_1 - \dot{x}_2) - k_1(x_1 - x_2) + u \quad (2.1)$$
$$m_w\ddot{x}_2 = c_1(\dot{x}_1 - \dot{x}_2) + k_1(x_1 - x_2) + c_2(\dot{w} - \dot{x}_2) + k_2(w - x_2) - u \quad (2.2)$$

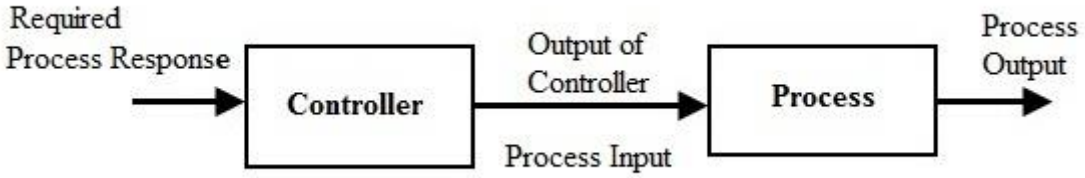
dimana $m_b$ , $m_w$, $k_1$, $k_2$, $c_1$dan $c_2$adalah unsprung dan sprung mass, koefisien pegas dan koefiesien damping dari elemen sprung dan unsprung . $x_1$, $x_2$, dan $w$ adalah perpindahan dari badan kendaraan, roda dan jalan.

## 2.4. Sistem Kontrol

Sistem kontrol adalah suatu sistem yang berfungsi untuk melakukan aksi kendali terhadap kondisi lingkungan agar sesuai dengan kondisi yang diharapkan, sistem kontrol akan mengukur nilai variabel yang dan menerapkan variabel dimanipulasi ke sistem untuk memperbaiki atau membatasi penyimpangan dari nilai yang terukur dari nilai yang diinginkan. Sistem kontrol terdiri dari input atau *setpoint* yang menyatakan kondisi terkini dari lingkungan atau respons sistem, output yang merupakan keluaran atau respons sistem aktual, dan *plant* yaitu objek yang dikendalikan atau dikontrol. Sistem kontrol pada umumnya terbagi menjadi dua yaitu sistem kontrol *Open Loop* dan sistem kontrol *Closed Loop*.

### 2.4.1. Sistem Kendali *Open Loop*

Sistem kendali *open loop* adalah sistem kontrol yang responsnya tidak mempengaruhi aksi dari pengontrolan. Tindakan pengendalian tidak dipengaruhi oleh keluaran sistem sehingga sistem tidak dapat melakukan koreksi apabila terjadi gangguan pada sistem. Kendali *open loop* digunakan apabila hubungan antara masukan dan keluaran diketahui dan tidak terdapat gangguan internal maupun eksternal. Pada sistem ini, perubahan keluaran hanya bisa di perbaiki dengan mengubah masukan secara manual. Contoh sistem kontrol *open loop* adalah mesin cuci otomatis, lampu lalu lintas, sistem penghangat rumah (tanpa thermostat, *feedback*, dan kontrol). Sistem kontrol *open loop* memiliki komponen yang simpel dalam desain, dan ekonomis. Diagram blok sistem kontrol *open loop* dapat dilihat pada gambar 2.5.

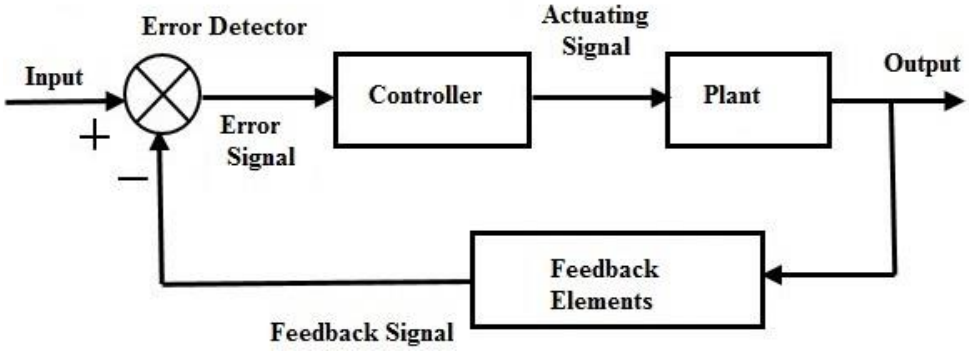



Gambar 2.5 Blok Diagram *Open Loop Control System* (electronicshub.com)

### 2.4.2. *Closed Loop*

Sistem kontrol *closed loop*, adalah sistem kontrol yang keluaran atau respons sistem mempengaruhi aksi kendali. Pada sistem kendali ini aksi pengendali sangat tergantung dari *feedback* yang diambil dari hasil keluaran sistem. Dengan ini sistem dapat mengoreksi kesalahan atau kelemahan yang ada pada keluaran sistem. Dalam sistem *closed loop*, variabel yang dikontrol (*output*) akan dilihat setiap waktu, dan dibandingkan dengan masukan yang diinginkan, yang akan menghasilkan *signal error*. Signal ini yang akan mengatur sistem untuk melakukan aksi yang dibutuhkan untuk mengoreksi sistem, sehingga mengeluarkan hasil keluaran yang sesuai dengan keinginan. Sistem ini juga memperhitungkan gangguan atau *disturbance*. Sistem kontrol ini akurat, stabil, dan lebih

tahan terhadap *noise*, tetapi sistem ini mahal dan rumit. Diagram blok sistem closed loop dapat dilihat pada gambar 2.6.




Gambar 2.6 Blok Diagram *Closed Loop Control System* (Control Systems – Introduction tutorialpoints.com)

## 2.5. Pengendali PID

Pengendali PID adalah salah satu pilihan standar dalam mendesain suatu sistem kontrol, Sistem ini terdiri dari tiga komponen, komponen proporsional, integral, dan derivatif seperti dimodelkan pada persamaan 2.3 berikut

$$u = K_p e_p + K_i e_i + K_d e_d \tag{2.3}$$

Dimana u adalah *output* dari kontrol dan $e_p, e_i, dan\ e_d$ adalah komponen *error* yang didefinisikan seperti persamaan 2.4 berikut

$$e_p = e,\ e_i = \int_0^t edt,\ ed = \frac{de}{dt} \tag{2.4}$$

Dimana e adalah input *error* (Marino & Neri, 2019). Pengendali PID merupakan bagian dari proses desain via *root locus* (Nise 2016). Pengendali PID merupakan gabungan dari beberapa komponen kendali yaitu *Proportional Integrative Controller* (PI) dan *Proporsional Derivative Controller* (PD), setiap komponen memiliki fungsi yang berbeda-beda dalam mengubah suatu karakteristik dari suatu sistem. Pada pengendali PD, data keluaran diolah dengan sinyal acuan secara *proportional derivative* dengan tujuan untuk memperbaiki *transient response* dari sebuah *close loop system*. Pengendali PI adalah sebuah pengendali yang mengolah data keluaran *proportional-integral* dengan tujuan untuk memperbaiki *steady state error* dari sebuah *close-loop system*. Pengendali PID adalah gabungan dari PI dan PD yang berfungsi untuk memperbaiki respons transien serta *steady state error* pada *close-loop system*, efek dari setiap komponen pengendali PID dapat dilihat pada tabel 2.1

Sebuah Pengendali PID biasanya digunakan dikarenakan harganya yang murah dan kemungkinan penggunaannya tanpa harus mengetahui pasti proses apa yang dikontrol. (Marino & Neri, 2019). Pengendali PID dapat memperbaiki keseluruhan sistem, baik untuk sistem dengan respons *transient* maupun *steady state* yang digunakan untuk mengontrol sistem suspensi aktif. Pengendali PID dapat dimodelkan dalam basis *error* sesuai dengan persamaan 2.5 berikut

$$G_c = k_p e(t) + k_i \int e(t)dt + k_d \frac{de}{dt} \tag{2.5}$$

Dimana, $G_c$ adalah output dari pengendali, $k_p$ adalah konstanta proportional $k_i$adalah konstanta integral dan $k_d$ adalah konstanta differential,

$e(t) = z_s - z_{desired}$ adalah *error* dari signal
$\int e(t)dt$ adalah waktu integral dari signal *error*
$\frac{de}{dt}$ adalah waktu derivative dari signal *error*

Tabel 2.1 Pengaruh Sistem Kontrol PID Terhadap Perubahan Parameter (Arumugam et al., 2009)

| Respon | Rise Time | (%OS) | (Ts) | Steady State *Error* |
|---|---|---|---|---|
| **$K_P$** | Turun | Naik | Sedikit Perubahan | Turun |
| **$K_I$** | Turun | Naik | Naik | Hilang |
| **$K_D$** | Sedikit Perubahan | Turun | Turun | Sedikit Perubahan |

Nilai *tuning* dari konstanta $k_p$, $k_i$, dan $k_d$ dari pengendali PID adalah proses yang menggunakan proses *trial and error*. Metode *tuning* PID dapat juga dilakukan dengan metode *Ziegler-Nichols* (Ziegler & Nichols, 1942).

### 2.5.1. Kendali Proporsional

Pada Pengendali proporsional jika G(s)=$K_p$, dengan $K_p$ adalah konstanta. Jika u=G(s)$e(t)$, maka u=$K_p$·e dengn $K_p$ adalah Konstanta Proporsional. $K_p$ adalah konstanta yang berfungsi sebagai *gain* atau penguat. Kendali proporsional memberikan penguatan tanpa memberikan efek dinamik kepada kinerja pengendali. Pada penggunaannya $K_p$ memiliki beberapa keterbatasan dikarenakan sifatnya yang tidak dinamis. Tetapi dalam pengaplikasian dasar kontrol P cukup mampu untuk memperbaiki *transient response*, khususnya pada property *rise time* dan *settling time*. Pengendali proporsional memiliki keluaran yang berbanding lurus atau proporsional dengan besarnya nilai sinyal *error*.

Sifat-sifat dari pengendali proporsional adalah :

1. Apabila nilai $K_p$ kecil, pengendali proporsional memberikan koreksi kesalahan yang kecil pula, sehingga akan menghasilkan respons yang lambat (menambah nilai *rise time*)
2. Jika nilai $K_p$ naik, maka respons sistem akan semakin cepat untuk mencapai *steady state* (mengurangi *rise time*).
3. Jika nilai $K_p$ dinaikkan lagi sehingga melewati nilai yang terlalu besar, akan mengakibatkan sistem bekerja tidak stabil sehingga sistem akan berosilasi.
4. Nilai $K_p$ dapat di-*set* pada nilai optimumnya sehingga mengurangi nilai *steady state error*, tetapi tidak membuat nilainya menjadi nol.

### 2.5.2. Kendali Integral

Pengendali integral berfungsi untuk menghasilkan respons dengan kesalahan dalam keadaan *steady* nya (SSE=0). Sebuah pengendali tanpa pengendali integral tidak mampu menjamin keluaran sistem dengan SSE nol. Keluaran kendali integral adalah hasil penjumlah dari nilai perubahan dari *error*, jika sinyal kesalahan tidak mengalami perubahan, maka keluaran akan tetap sama

seperti sebelumnya. Sinyal keluaran dari pengendali integral merupakan luas bidang yang dibentuk oleh kurva *error*.

Sifat-sifat dai pengendali integral :

1. Keluaran dari pengendali integral membutuhkan selang waktu tertentu, sehingga pengendali integral cenderung memperlambat respon.
2. Ketika *error* bernilai nol, keluaran pengendali akan bertahan pada nilai sebelumnya.
3. Jika sinyal kesalahan tidak bernilai nol, keluaran akan menunjukkan kenaikan dan penurunan dipengaruhi oleh besarnya nilai kesalahan dan nilai dari $K_i$.
4. Konstanta integral $K_i$ yang berharga besar akan mempercepat hilangnya *offset*. Tetapi semakin besar nilai konstanta $K_i$ akan mengakibatkan peningkatan osilasi dari sinyal keluaran pengendali.

### 2.5.3. Kendali Derivatif

Keluaran pengendali derivatif memiliki sifat seperti halnya suatu operasi derivatif atau turunan. Perubahan yang mendadak pada masukan pengendali akan mengakibatkan sinyal derivatif yang sangat besar dan cepat. Ketika masukannya tidak mengalami perubahan, keluaran pengendali juga tidak mengalami perubahan, sedangkan apabila sinyal masukan berubah mendadak dan menaik (berbentuk fungsi *step*), keluaran menghasilkan sinyal berbentuk impuls. Jika sinyal masukan berubah naik secara perlahan (fungsi *ramp*), keluarannya justru merupakan fungsi step yang besar magnitudonya sangat dipengaruhi oleh kecepatan naik dari fungsi *ramp* dan nilai konstanta $K_d$.

Dari persamaan sinyal kontrol u yang dihasilkan oleh kendali D dapat dinyatakan bahwa sifat kendali D tergantung pada rate perubahan dari error. Dengan sifat ini pengendali D dapat digunakan untuk memperbaiki transient response dengan memprediksi error yang akan terjadi. Kendali derivatif hanya berubah saat ada perubahan error sehingga saat error bernilai konstan, pengendali ini tidak akan bereaksi, hal ini pula yang menyebabkan kendali derivatif tidak dapat digunakan tanpa pengendali lain.

Adapun sifat-sifat dari pengendali derivatif :

1. Pengendali tidak dapat menghasilkan keluaran jika tidak ada perubahan pada nilai *error*.
2. Jika sinyal kesalahan berubah terhadap waktu, maka keluaran yang dihasilkan tergantung pada nilai $K_D$ dan laju perubahan sinyal kesalahan.
3. Pengendali derivatif mempunyai suatu karakter untuk mendahului, sehingga pengendali ini dapat menghasilkan koreksi yang signifikan sebelum terjadi kesalahan atau *error* menjadi sangat besar. Sehingga pengendali diferensial dapat mengantisipasi pembangkit kesalahan, memberikan aksi korektif dan cenderung meningkatkan stabilitas sistem,
4. Dengan menaikkan nilai $K_D$, dapat meningkat stabilitas sistem dan mengurangi *overshoot*

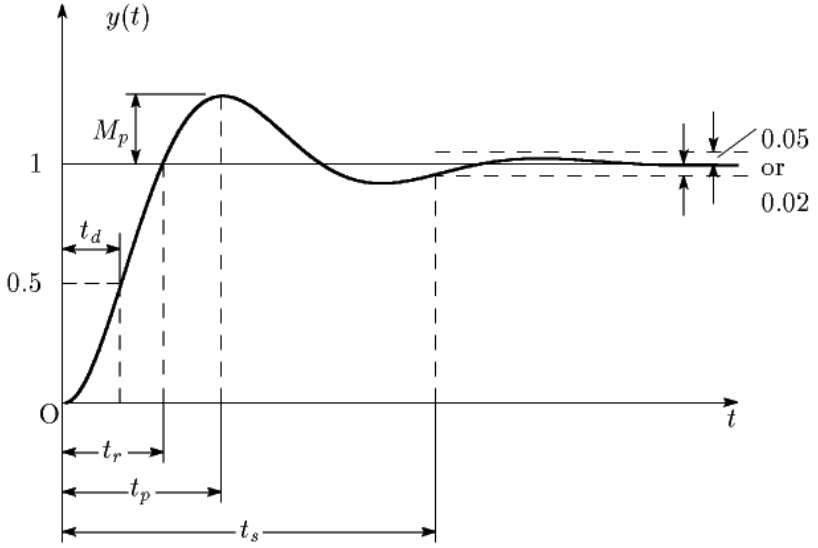
Pengoperasian pengendali diferensial hanya berlaku dalam waktu yang sempit, yaitu selama masa transisi. Oleh karena itu, pengontrol diferensial tidak akan digunakan tanpa pengontrol lain. Berdasarkan karakteristik pengendali ini, pengendali diferensial biasanya digunakan untuk mempercepat respons awal sistem, tetapi tidak untuk mengurangi kesalahan kondisi *steady state*.

## 2.6. Karakteristik Respons Waktu (*Time Response*)

Karakteristik respons waktu merupakan karakteristik respons yang melihat spesifikasi performa berdasarkan bentuk respons *output* terhadap perubahan waktu. Terdapat dua jenis pengamatan, yaitu :

### 2.6.1. Spesifikasi respons *transient*

Merupakan pengamatan spesifikasi respons sistem pada saat terjadi perubahan sinyal *input* maupun gangguan hingga respons memasuki keadaan *steady state*. Parameter yang digunakan untuk mengukur kualitas respons *transient* ini antara lain *rise time, peak time, settling time,* dan *Peak Overshoot* atau *Maximum Overshoot*. Parameter ini digunakan pada sistem orde dua. Grafik respons transient beserta karakteristik yang bisa dianalisa dari grafik tersebut dapat dilihat pada gambar 2.7 berikut




Gambar 2.7 Grafik respons dari suatu sistem (http://shiwasu.ee.ous.ac.jp)

Dari gambar 2.7 dapat dilihat beberapa karakteristik respons dari grafik respons transient, berikut penjelasan dari karakteristik-karakteristik berikut

a.*Peak Overshoot* ($M_p$)

*Peak Overshoot* adalah harga puncak maksimum dari kurva respon yang diukur dari nilai masukan.

b. *Rise Time ($t_r$)*

Waktu yang dibutuhkan respons untuk naik mulai naik dari 10 – 90%, 5 – 95% atau 0 – 100% dari nilai masukan yang diberikan. menuju ke 90% nilai akhir dari 10% nilai akhir respons.

c. *Settling time* ($t_s$)

Waktu yang dibutuhkan oleh respons untuk mencapai dan mempertahankan *(steady state)* untuk tetap pada nilai masukan yang diberikan didalam toleransi yang dapat diterima. Ukurannya ditentukan dengan persentase mutlak dari harga akhir yang biasanya 5% atau 2%.

d. *Peak Time ($t_p$)*

Waktu yang dibutuhkan respons untuk mencapai puncak overshoot untuk pertama kali.

### 2.6.2. Spesifikasi Respons *Steady State*

Merupakan pengamatan spesifikasi respons sistem pada saat respons memasuki keadaan tersebut hingga waktu tak terbatas $t = \infty$. Parameter untuk mengukur kualitas respon *steady-state* ini adalah %*steady-state error. Steady-state error* dalam domain waktu dapat dirumuskan sebagai rumus 2.13 berikut

$$e(t) = r(t) - c(t) \qquad (2.13)$$

dengan :

e(t) = fungsi *error*, berubah terhadap waktu

r(t) = fungsi input

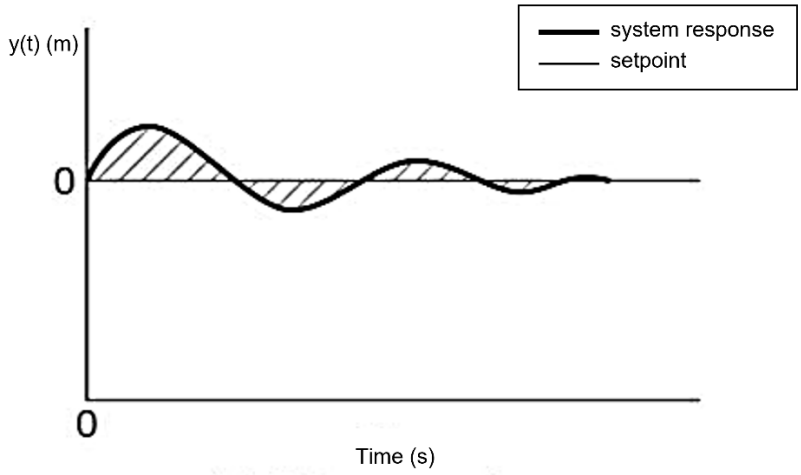
c(t) = fungsi output, berubah terhadap waktu

t = waktu

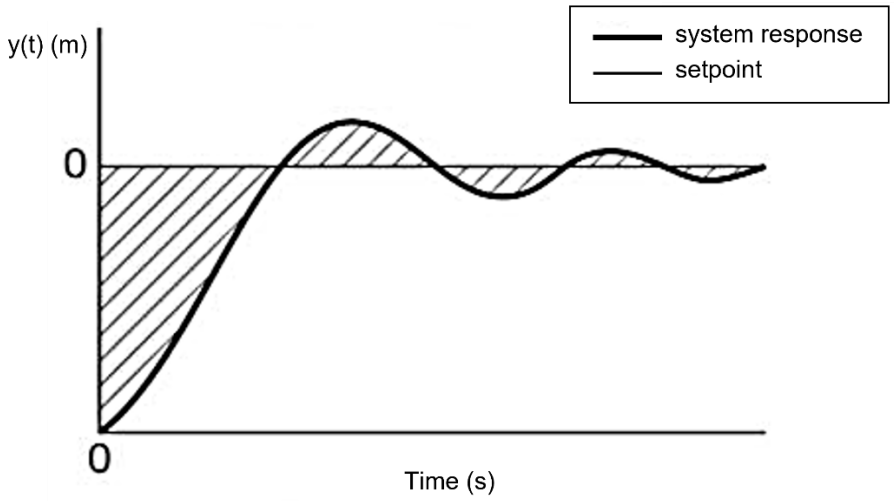
*Steady-state error* terjadi pada saat kondisi tunak (*steady-state*), sehingga *error* tersebut terjadi pada waktu yang menuju tak hingga dan dalam keadaan nilai output sudah tidak berubah terhadap waktu.

## 2.7. Kriteria Performa Sistem Kontrol (Rs, 2012)

Terdapat beberapa kriteria performa yang dapat digunakan dari sistem kontrol, salah satu metode yang sering digunakan adalah dengan cara menghitung total *error* dari sistem, yaitu menjumlahkan *error* dalam waktu tertentu. Hal ini sama juga dengan melakukan integral dari luas daerah yang dibentuk oleh garis respons sistem dengan garis setpoint, seperti ditunjukan pada gambar 2.8 (a) dan (b). Terdapat beberapa variasi dalam menghitung integral *error* yang memiliki kelebihan dan kekurangan masing-masing seperti ditunjukkan pada subbab 2.7.1-2.7.3




(a)




(b)

Gambar 2.8 (a) dan (b) Ilustrasi *Integral Error* untuk *disturbance change* dan *setpoint change* (Model-based PID tuning methods Two degree of freedom controllers, Shayna 2014)

### 2.7.1. *Integral of Squared Error (ISE)*

$$ISE = \int_0^\infty \{e(t)\}^2 dt \qquad (2.14)$$

ISE mengintegrasikan kuadrat kesalahan dari waktu ke waktu. Galat yang kecil akan menghasilkan ISE yang besar. Dengan meminimalkan ISE cenderung menghilangkan kesalahan besar dengan cepat, tetapi akan mentolerir kesalahan kecil yang bertahan untuk jangka waktu yang lama. Rumus ISE dapat ditunjukkan pada persamaan 2.14. Dengan menggunakan kriteria ini kita akan mendapatkan respons paling cepat, dengan amplitudo cukup rendah, tetapi berosilasi. Ilustrasi daerah *error* dapat dilihat pada Gambar 2.8, dimana setiap titik *error* akan dikuadratkan dan di jumlahkan.

### 2.7.2. *Integral Of Time Multiplied By Absolute Error* (ITAE)

$$ITAE = \int_0^\infty t|e(t)| \ dt \qquad (2.15)$$

ITAE memiliki pengali waktu tambahan dari galat fungsi yang menitikberatkan pada lamanya durasi galat, oleh karena itu kriteria ini paling sering diterapkan dalam sistem yang membutuhkan waktu penetapan yang cepat. Persamaan untuk menghitung nilai ITAE dapat dilihat pada persamaan 2.15.

Pada ITAE, nilai *error* di setiap titik akan dikalikan dengan waktu kemudian dijumlahkan, sehingga *error* yang masih terjadi setelah waktu yang lama akan mendapatkan bobot yang besar, ilustrasi dapat dilihat di Gambar 2.8. Pada penelitian ini ITAE akan digunakan sebagai parameter optimasi. ITAE akan dobahas lebih lanjut pada subbab 2.8.

### 2.7.3. *Integral Of Absolute Error (IAE)*

$$IAE = \int_0^\infty |e(t)| \ dt \qquad (2.16)$$

IAE memiliki prinsip yang sama dengan ITAE namun tidak dikalikan oleh waktu. IAE mengintegrasikan kesalahan mutlak dari waktu ke waktu seperti pada Gambar 2.8. Persamaan untuk menghitung nilai ITAE dapat dilihat pada persamaan 2.16.

Hal ini menghasilkan respons lebih lambat dari sistem ISE optimal, tetapi osilasinya lebih berkurang sehingga lebih cepat teredam.

## 2.8. Hubungan Parameter yang Di optimasi (Nilai $K_P$, $K_I$ dan $K_D$), Fungsi Objektif (Nilai ITAE), dan Parameter Performa Pengendali ($T_S$ dan *Peak Overshoot*)

Pada penelitian ini akan dilakukan optimasi parameter $K_P$, $K_I$, dan $K_D$ menggunakan parameter ITAE *(Integral Timed Absolute Error)*, diharapkan dengan mengoptimasi nilai ini, akan mendapatkan performa pengendali terbaik, yang dilihat dari parameter $T_S$ dan *Peak Overshoot*.

Hubungan antar parameter-parameter akan dijelaskan pada subbab ini

### 2.8.1. Hubungan Antara Parameter $K_P$, $K_I$, dan $K_D$ dengan ITAE

Nilai $K_P$, $K_I$, dan $K_D$ yang optimal akan menghasilkan nilai ITAE yang rendah dikarenakan kemampuannya yang baik untuk meredam amplitudo getaran dan dalam waktu yang cepat. Tidak ada rumusan yang pasti untuk menghubungkan antara nilai $K_P$, $K_I$, dan $K_D$. Namun, kembali lagi nilai $K_P$, $K_I$, dan $K_D$ yang optimal akan memperkecil nilai *error* dimana dalam penelitian ini menggunakan nilai ITAE.

Nilai ITAE ditentukan oleh respons dari sistem yang bergantung pada optimal atau tidaknya nilai $K_P$, $K_I$, dan $K_D$. Nilai $K_P$, $K_I$, dan $K_D$ yang optimum dicari melalui proses *tuning*. Ada banyak metode untuk melakukan *tuning*, dan pada setiap metode tidak memiliki hubungan rumus tertentu yang pasti dan universal, yang menghubungkan antara nilai $K_P$, $K_I$, dan $K_D$ dan *error* yang terjadi untuk semua sistem yang ada. Oleh sebab itu pada penelitian ini penulis berusaha untuk menggunakan metode metaheuristik untuk menemukan suatu hubungan antara parameter-parameter ini.

Untuk memberikan gambaran hubungan antar parameter-parameter ini, dimisalkan dua buah sistem suspensi aktif dengan pengendali PID yang dimodelkan dan diberi nama Sistem A dan Sistem B. Dua sistem ini memiliki parameter sistem sesuai dengan *paper* referensi yang dapat dilihat pada subbab 3.1.2. Pembeda antara dua sistem ini adalah parameter pengendali PID-nya yaitu nilai $K_P$, $K_I$, dan $K_D$ . Parameter kontroler kedua yang didefinisikan sebagai berikut

Sistem A

$K_P$ = 801838

$K_I$ = 649984

$K_D$ = 249681

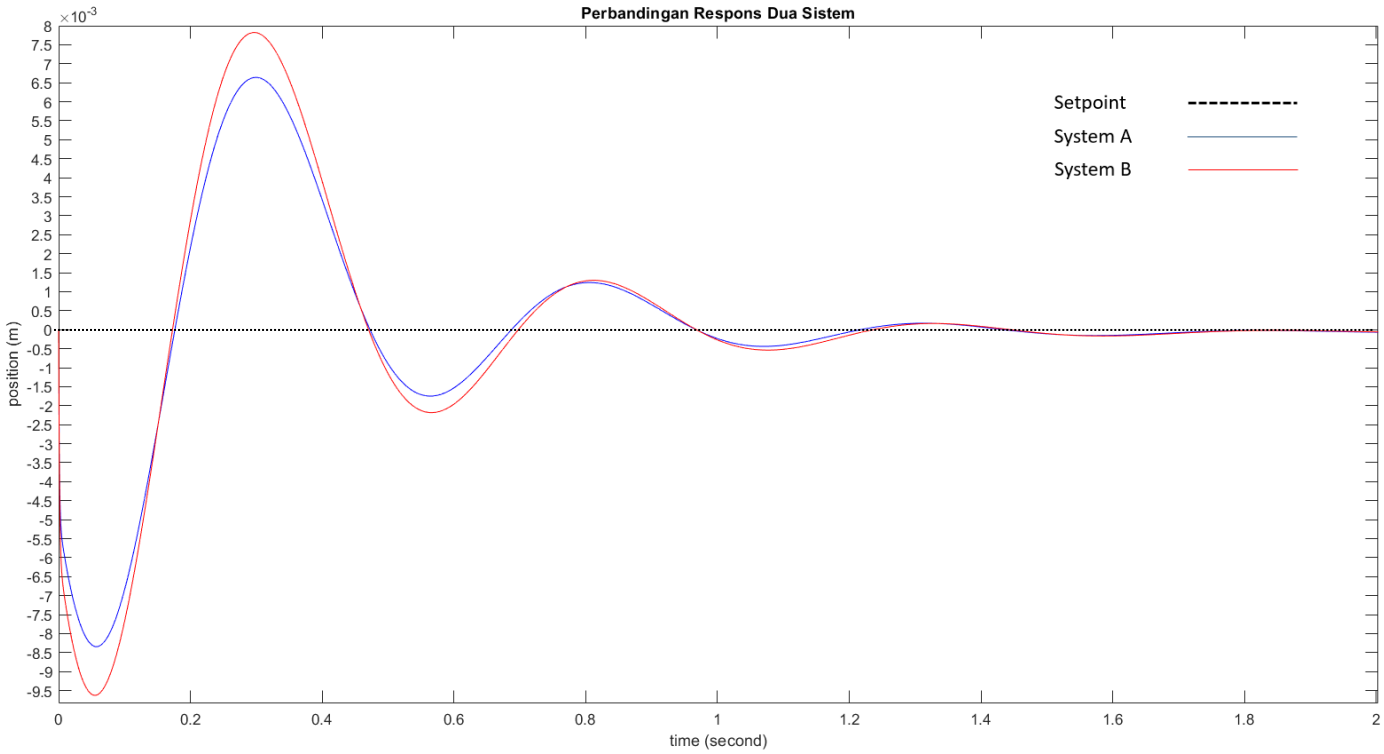
Sistem B

$K_P$ = 847879

$K_I$ = 649291

$K_D$ = 204070

Sistem A dan Sistem B kemudian disimulasikan dengan diberikan *disturbance* berupa input *step* sebesar 0,1 m, dan dianalisa grafik respons nya.



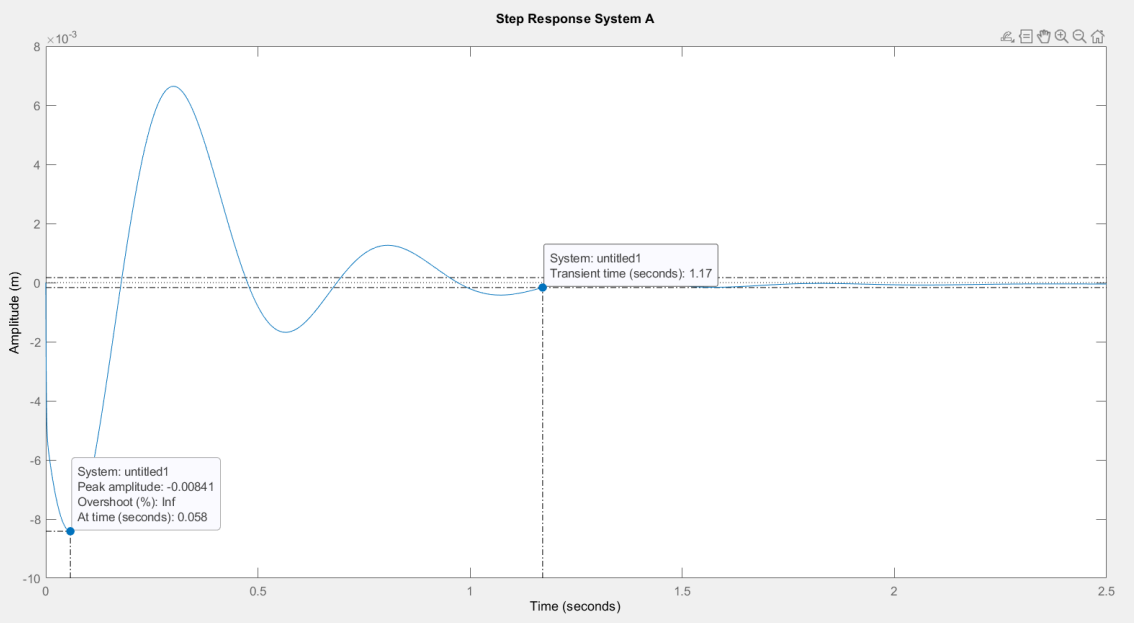

Gambar 2.9 Perbandingan Dua Grafik Respon

Pada gambar 2.9 di atas dapat diperhatikan perbandingan antara dua grafik respons dengan nilai $K_P$, $K_I$ dan $K_D$ yang berbeda. Dari perhitungan nilai ITAE sesuai dengan persamaan 2.15, nilai ITAE dari sistem A didapatkan sebesar 0,0010074 dan sistem B sebesar 0,001299. Hal ini menujukkan bahwa sistem A memiliki parameter pengendali PID yang lebih optimal dibandingkan dengan sistem B. Hal ini akan didukung dengan parameter $T_S$ dan *Peak Overshoot* yang akan dibahas pada subbab berikutnya.

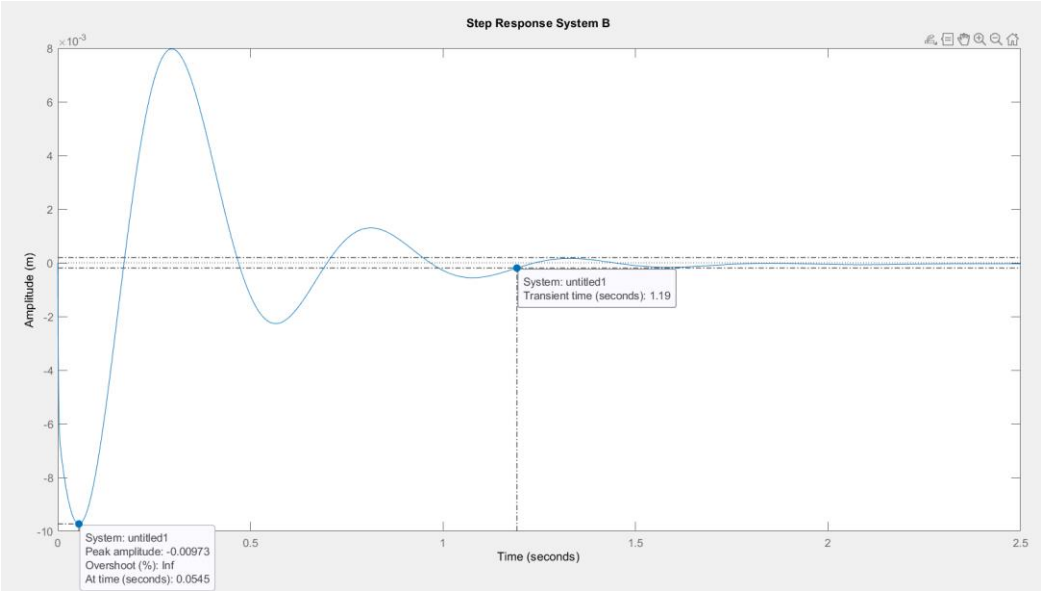
### 2.8.2. Hubungan Antara Parameter ITAE dengan $T_S$ dan *Peak Overshoot*

Setelah mengetahui hubungan antara parameter pengendali PID dengan nilai ITAE, pada subbab ini akan dihubungkan bagaimana nilai ITAE berpengaruh pada parameter $T_S$ dan *Peak Overshoot*.

Dua grafik respons dengan karakteristik respons dari Sistem A dan Sistem B dibandingkan yang ditunjukkan oleh gambar 2.10 dan 2.11




Gambar 2.10 Grafik Respons Dari Sistem A




Gambar 2.11 Grafik Respons Dari Sistem B

Nilai *Peak Overshoot* yang besar dapat menyebabkan nilai ITAE semakin besar, hal ini dikarenakan *Peak Overshoot* yang besar akan memperluas daerah *error* seperti diobservasi apabila dibandingkan karakteristik respons sistem A dan sistem B pada gambar 2.10 dan 2.11. Dapat diperhatikan bahwa nilai *Peak Overshoot* dari sistem A yaitu -0,00841 m sedangkan sistem B -0,00973. Hal ini selaras dengan nilai ITAE sistem A yang lebih kecil dibandingkan dengan sistem B.

Begitu juga dengan nilai $T_S$, nilai $T_S$ yang besar akan memperbesar nilai ITAE, hal ini dikarenakan terdapat variabel faktor pengali t dalam perhitungan ITAE, apabila suatu pengendali gagal menstabilkan suatu sistem dalam waktu yang singkat, maka variabel t akan menjadi semakin besar dan memperbesar faktor pengali dalam perhitungan nilai ITAE. Sebagai contoh nilai *error* pada sistem B yang masih ada sampai detik ke 1,5, nilai *error* pada titik itu akan

dikalikan dengan 1,5, begitu juga dengan nilai *error* yang berada didetik selanjutnya, nilai ITAE akan menjadi lebih tinggi secara signifikan dikarenakan nilai *error* akan dikalikan dengan faktor t.

Seperti diobservasi dan dibandingkan antara sistem A dan sistem B pada gambar 2.10 dan 2.11. Sistem A memiliki nilai *settling time* sebesar 1,17s dan sistem B 1,19 s. Hal ini selaras dengan nilai ITAE sistem A yang lebih kecil dari sistem B.

Sehingga dari penjelasan diatas dapat disimpulkan apabila sebuah sistem memiliki nilai $T_S$ dan *Peak Overshoot* yang besar maka sistem tersebut memiliki nilai ITAE yang besar. *Peak overshoot* dan *settling time* yang tinggi akan membuat nilai ITAE menjadi lebih besar dikarenakan akan memperluas luas daerah *error*.

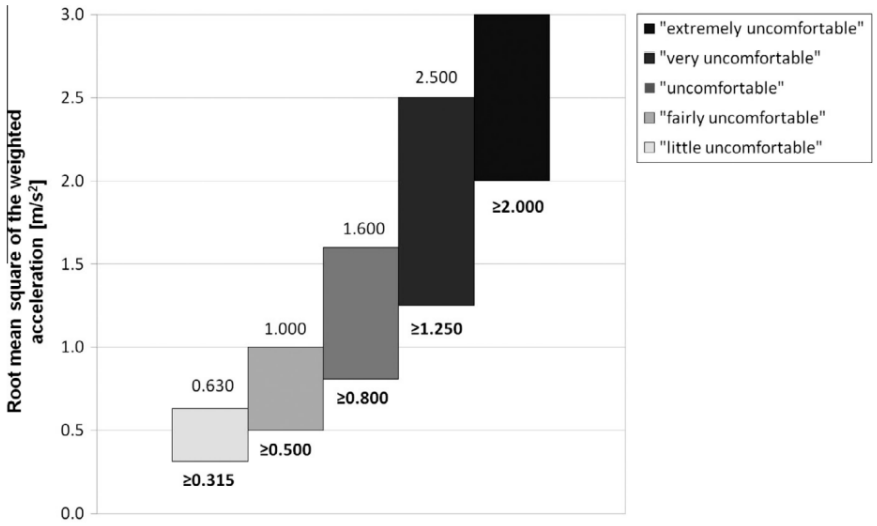

Gambar 2.12 Diagram Hubungan Tiap Parameter

Pada gambar 2.12 diatas menggambarkan hubungan antara, parameter yang di optimasi, fungsi objektif optimasi dan parameter performa hasil optimasi. Dari gambar diatas, nilai ITAE dapat pula menghasilkan nilai yang sama untuk dua sistem dengan karakteristik respons yang berbeda. Hal ini dikarenakan dua sistem tersebut memiliki nilai penjumlahan *error* yang sudah dikalikan dengan variabel t yang sama, pada hal ini bisa saja nilai keuntungan nilai *peak overshoot* yang rendah pada sistem pertama dirugikan oleh *settling time* yang lama, begitu juga sebaliknya, *peak overshoot* yang tinggi pada sistem kedua diuntungkan dengan *settling time* yang cepat.

## 2.9. Kriteria Kenyamanan Kendaraan

Kenyamanan adalah salah satu parameter untuk menentukan kualitas dari kendaraan. Salah satu kriteria kenyamanan bagi penumpang kendaraan yang terkena getaran vertikal adalah kriteria yang melihat parameter *Whole-Body Vibration* (WBV). WBV adalah jenis getaran yang dirasakan oleh penumpang dan pengendara saat berada didalam kendaraan. Kriteria ini diatur dalam suatu standar yang diatur oleh *International Standard Organization* yang dipublikasikan pada ISO2631. Paparan WBV dari gerakan kendaraan dapat mengakibatkan ketidaknyamanan pada pengendara.

Standar ini akan digunakan dalam penelitian ini sebagai kriteria kenyamanan dari pengendara. Gambar 2.13 menunjukkan kriteria reaksi kenyamanan pengendara terhadap getaran yang terjadi. Dalam standar ini parameter yang dilihat adalah nilai *Root Mean Square* (RMS) dari akseselerasi kendaraan. Kriteria dibagi menjadi enam yaitu *extremely uncomfortable, very uncomfortable, uncomfortable, fairly uncomfortable,* dan *little uncomfortable*.




Gambar 2.13 Reaksi Kenyamanan Penumpang Terhadap RMS getaran (ISO 2631)

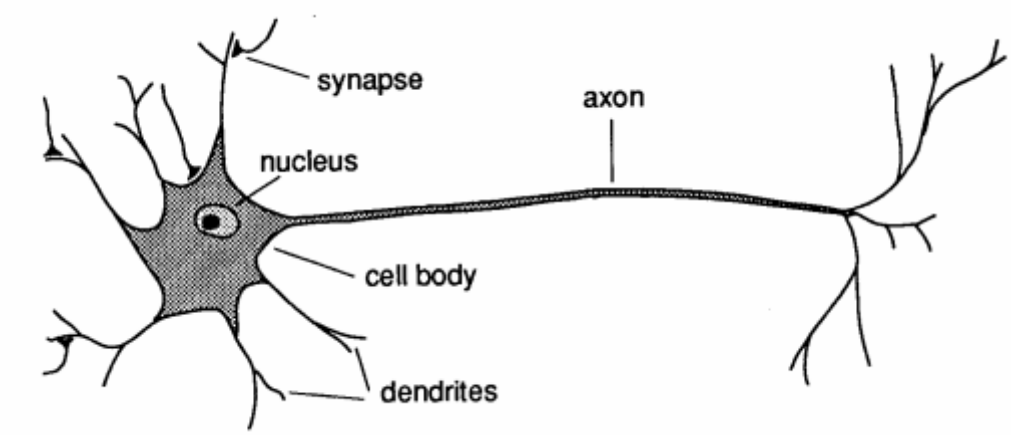
Batas-batas nilai dari masing-masing parameter dapat dilihat pada tabel 2.2 berikut
Tabel 2.2 Kriteria Kenyamanan Penumpang Kendaraan (ISO 2631)

| No. | Getaran | Keterangan |
|---|---|---|
| 1 | > 0,315 m/s$^2$ | Tidak ada eluhan |
| 2 | 0,315 – 0,63 m/s$^2$ | Sedikit tidak nyaman |
| 3 | 0,5 -1 m/s$^2$ | Agak tidak nyaman |
| 4 | 0,8 -1,6 m/s$^2$ | Tidak nyaman |
| 5 | 1,25 – 2,5 m/s$^2$ | Sangat tidak nyaman |
| 6 | > 2 m/s$^2$ | Amat sangat tidak nyaman |

Diharapkan getaran yang dialami oleh pengendara tidak berada pada kriteria tidak nyaman (*uncomfortable)* atau lebih buruk yaitu diatas 0,8 m/s$^2$

## 2.10. *Artificial Neural Network*

Neuron adalah satuan unit pemroses terkecil pada otak, bentuk sederhana ada sebuah neuron yang oleh para ahli dianggap sebagai satuan unit pemroses. Neuron digambarkan seperti Gambar 2.14.




Gambar 2.14 Neuron Pada Sistem Syaraf Manusia (Fausett & Fausett, 1994)

Gambar tersebut adalah satuan unit jaringan otak manusia yang telah disederhanakan. Jaringan otak manusia tersusun tidak kurang dari $10^{13}$ buah neuron yang masing-masing terhubung oleh sekitar $10^{15}$ buah dendrit. Fungsi dendrit adalah menyampaikan sinyal antar neuron yang terhubung. Bagian yang menerima sinyal disebut *sinaps*, dan yang mengeluarkan sinyal disebut sebagai *axon*. Penjelasan lebih rinci tentang hal ini dapat diperoleh pada disiplin ilmu *biology molecular*.

Jaringan syaraf secara umum terdiri dari jutaan, bahkan lebih, struktur dasar neuron yang terinterkoneksi dan terintegrasi antara satu dengan yang lain sehingga jaringan-jaringan tersebut dapat melaksanakan aktivitas secara teratur dan kontinu sesuai dengan kebutuhan

*Artificial Neural Network* atau Jaringan Syaraf Tiruan (JST) adalah sebuah perkembangan dari generalisasi model matematika dari kemampuan kognisi manusia atau sebuah syaraf biologis. Sistem ini melakukan proses pengolahan informasi berdasarkan asumsi bahwa:

1. Proses pengolahan informasi dilakukan dalam elemen neuron
2. Sinyal dialirkan diantara neuron melalui suatu konektor
3. Setiap sambungan neuron memiliki nilai *weight*, dimana sinyal akan dikalikan dengan nilai *weight* tersebut

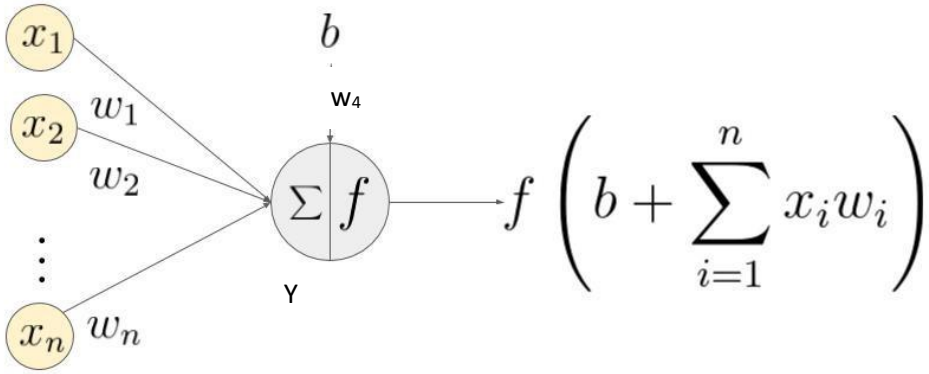
Setiap neuron memiliki fungsi aktivasi (biasanya non-linear) yang mengubah masukannya yang telah dikalikan oleh *weight* dan akan menghasilkan suatu nilai keluaran

Sebuah *neural network* memiliki dikarakteristikan sebagai berikut,

1. Pola dari koneksi diantara neuron (disebut sebagai arsitektur)
2. Metode dari menentukan nilai *weights* diantara sambungan (disebut sebagai *training* atau *learning algorithm)*
3. Fungsi aktivasi

Neural *network* dapat diaplikasikan pada *Signal Processing*, *Control*, *Pattern Recognition*, Obat-obatan, *Speech Production* and *Recognition*, dan Bisnis.(Fausett & Fausett, 1994)

### 2.10.1. Prinsip Dasar ANN




Gambar 2.15 Neuron dalam jaringan syaraf tiruan (towardsdatascience.com/power-of-a-single-neuron-perceptron-c418ba445095)

Pada neuron Y yang diilustrasikan oleh gambar 2.15, neuron Y menerima masukan dari neuron $X_1$, $X_2$, $X_n$, dan b (bias) ke Neuron Y dengan nilai *weight* $w_1$, $w_2$, $w_3$, dan $w_4$. Masukan neuron Y adalah penjumlahan dari *output* sinyal dari $X_1$,$X_2$, dan $X_3$ yang telah dikalikan oleh faktor *weight* $w_1$,$w_2$ dan $w_3$. Persamaan untuk menghitung masukan ke neuron Y dapat dilihat pada persamaan 2.17.

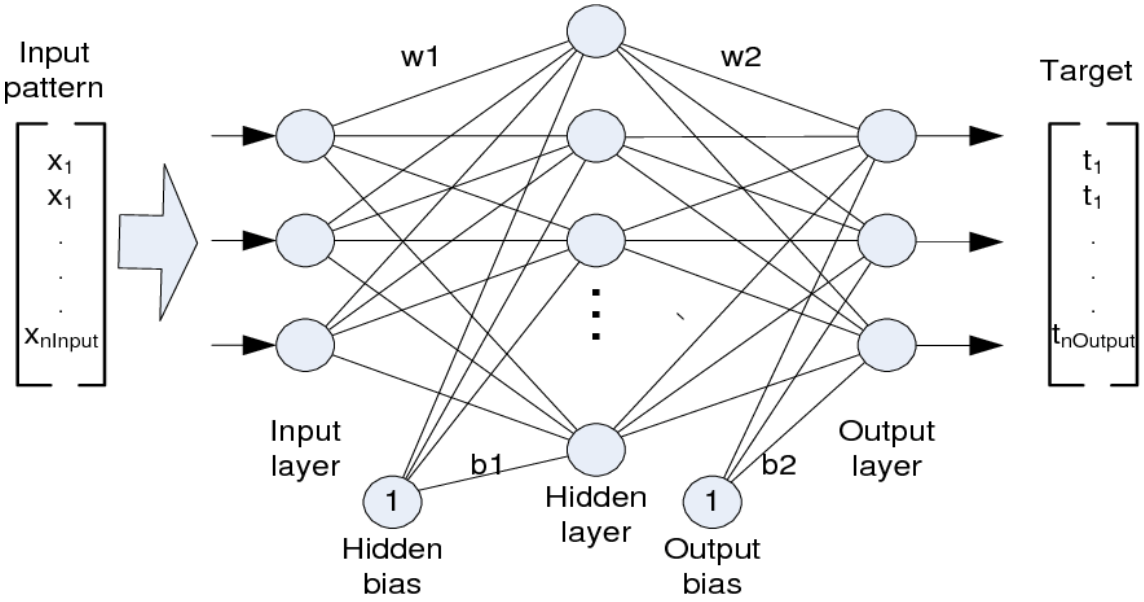
$$y_{in} = w_1X_1 + w_2X_2 + w_3X_3 + \cdots + w_nX_n + w_4b \qquad (2.17)$$

Fungsi aktivasi y pada neuron Y adalah suatu fungsi dari nilai masukannya, contohnya pada kasus ini adalah fungsi *logistic sigmoid*. Fungsi *logistic sigmoid* dapat dilihat pada persamaan 2.18.

$$f(x) = \frac{1}{1+\exp(-x)} \qquad (2.18)$$

Fungsi aktivasi dapat diubah dengan fungsi aktivasi lain, yang akan dibahas pada 2.8.3.

Apabila neuron disambungkan dengan neuron lain seperti gambar 2.16 dengan nilai *weight* masing-masing. Neuron Y mengirim sinyal pada setiap unit ini. Secara umum nilai yang diterima oleh setiap neuron akan berbeda, dikarenakan setiap sinyal akan dikalikan oleh *weight* masing-masing. Dalam jaringan pada umumnya, nilai aktivasi setiap neuron akan bergantung pada *input* dari banyak neuron, seperti di tunjukkan dalam contoh sederhana ini. Arsitektur dari jaringan dan metode *training* dari jaringan akan dibahas pada 2.8.2.




Gambar 2.16 Neuron dalam jaringan syaraf tiruan yang dihubungkan dengan *hidden layer* dan unit keluaran (Radivoje Radi, 2011)

Walaupun *neural network* pada gambar sangat sederhana, dengan adanya *hidden unit*, dan juga fungsi aktivasi yang non linear, memberikan kemampuan untuk menyelesaikan lebuh banyak lagi masalah, dibandingkan dengan jaringan dengan hanya unit *input* dan *output*. Di sisi lain jarin gan ini akan lebih sulit untuk di latih (mencari nilai optimal dari weights).

Salah satu karakteristik penting dalam Jaringan Syaraf Tiruan yang sama dengan Jaringan Syaraf Biologis adalah toleransi kesalahan. Sistem Jaringan Biologis memberikan toleransi kesalahan terhadap dua kasus. Pertama dapat mengenali banyak sinyal masukan yang bahkan berbeda dengan sinyal yang biasa diterima oleh jaringan tersebut. Contohnya adalah ketika kita mampu melihat seseorang yang belum pernah kita kenal sebelumnya dalam sebuah foto, atau mengenali seseorang yang sudah lama kita tidak temui.(Fausett & Fausett, 1994)

Kedua, jaringan ini mampu mentoleransi kerusakan pada sistemnya. Manusia lahir dengan 100 miliar neuron. Sebagian besar dari neuron ini akan mati dan tidak digantikan. Walaupun jumlah neuron kita kan terus mengalami pengurangan, kita masih dapat melakukan proses pembelajaran. Meskipun dalam kasus traumatis seperti kecelakaan, neuron lain bisa dilatih untuk mengambil alih fungsi dari sel neuron yang rusak. Begitu juga dengan Jaringan Syaraf Tiruan, jaringan ini dapat didesain untuk tidak sensitif terhadap kerusakan kecil, dan juga jaringan dapat dilatih untuk mengatasi kerusakan yang signifikan.

### 2.10.2. Arsitektur Jaringan

Arsitektur dari jaringan yang sering dipakai dalam jaringan syaraf antara lain :

*a.Single Layer Network*

*Single Layer Network* adalah jaringan yang hanya memiliki satu lapisan dari koneksi dengan *weights*. Jaringan hanya terdiri dari *input units*, yang menerima sinyal dan *output unit* yang merespons ke keluaran jaringan.

*Single Layer Network* dapat digambarkan pada gambar. Pada gambar 2.15 dapat terlihat bahwa *unit input* akan terkoneksi sepenuhnya ke *unit output* dan tidak terkoneksi oleh *unit input* lainnya dan *output unit* tidak terkoneksi dengan *output unit* lainnya

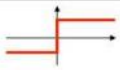
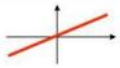
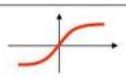
*b. Multilayer Net*

*Multilayer Net* adalah jaringan dengan satu atau lebih lapisan *nodes* (biasanya disebut *hidden units*) antara input units dan output *units*. Umumnya, terdapat lapisan *weights* diantara dua level yang bersebelahan (input, hidden, dan output), tetapi *training* dapat menjadi lebih sulit dengan arsitektur ini. *Multilayer Net* dapat dilihat pada gambar 2.16.

### 2.10.3. Fungsi aktivasi

Fungsi aktivasi adalah fungsi yang dipakai untuk menentukan *output* dari sebuah *neuron*. Masukan dari fungsi aktivasi adalah penjumlahan linear masukan dan nilai *weight*. Jika $a = \sum x_i w_i$ , maka $f(a) = f(\sum x_i w_i)$

Tabel 2.3 Fungsi-fungsi aktivasi dalam Jaringan Syaraf Tiruan (Primartha, 2018)

| Activation function | Equation | Example | 1D Graph |
|---|---|---|---|
| Unit step (Heaviside) | $\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$ | Perceptron variant |  |
| Sign (Signum) | $\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$ | Perceptron variant |  |
| Linear | $\phi(z) = z$ | Adaline, linear regression |  |
| Piece-wise linear | $\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$ | Support vector machine |  |
| Logistic (sigmoid) | $\phi(z) = \frac{1}{1 + e^{-z}}$ | Logistic regression, Multi-layer NN |  |
| Hyperbolic tangent | $\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$ | Multi-layer Neural Networks |  |

Pada tabel 2.3 ditunjukkan beberapa fungsi aktivasi yang biasa digunakan.

### 2.10.4. Bias

Bias adalah suatu unit masukan yang ditambahkan pada model agar model dapat memberikan lebih banyak fleksibilitas kepada model, hal ini juga membantu apabila masukan bernilai 0, dengan tidak adanya bias, maka hanya nilai w yang bisa berubah-ubah. Nilai bias juga dapat berdampak pada grafik fungsi aktivasi, yang berefek pada grafik yang dapat bergeser ke kiri atau ke kanan. Penjumlahan sebuah nilai keluaran apabila melibatkan bias, dapat dilihat dalam persamaan 2.19 (Fausett & Fausett, 1994)

$$a = b + \sum_{i=1}^{n} x_i w_i \qquad (2.19)$$

Dengan a adalah output, b adalah bias, n adalah jumlah input, x=input, w=bobot (*weights*)

### 2.10.5. Pelatihan jaringan syaraf tiruan (mengatur besarnya *weights*)

Ada 2 macam pelatihan untuk mengubah nilai *weight* yaitu pelatihan dengan pengawasan (*supervised training)* dan tanpa pengawasan *(unsupervised training)*

*a.Supervised Training*

*Supervised training* adalah metode pembelajaran yang paling umum digunakan dalam mengatur setting sebuah *neural network*. Proses pelatihan dilakukan dengan memberikan data pelatihan dengan pola yang akan yang akan diasosiasikan oleh jaringan dengan nilai vektor output. Nilai *weights* kemudian akan diatur berdasarkan dari *learning algorithm* (Fausett & Fausett, 1994).

*b. Unsupervised Training*

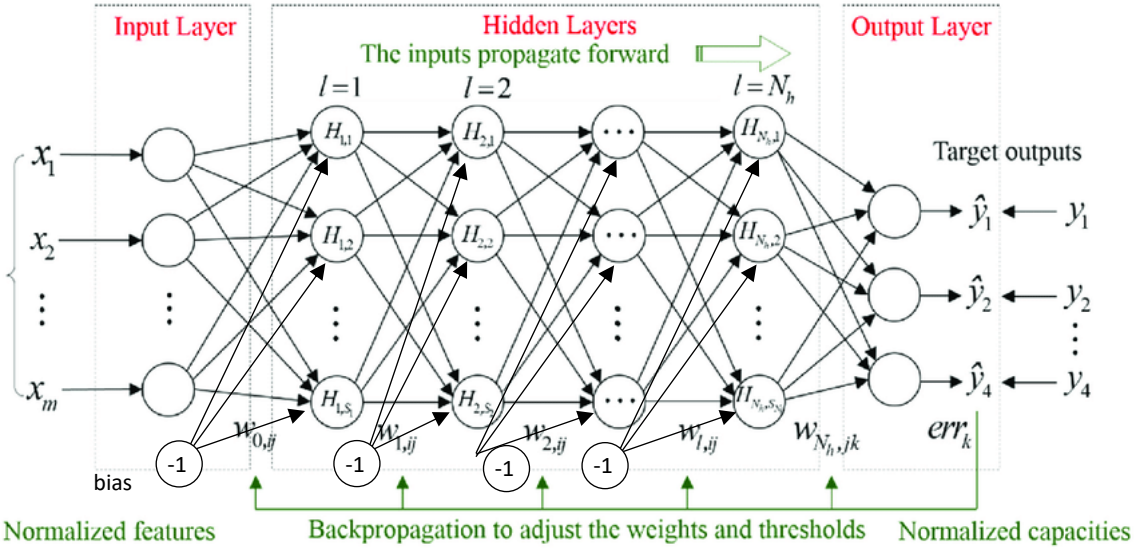
Pada *Unsupervised Training,* jaringan akan mengelompokkan vektor input tanpa menggunakan data *training* untuk menypesifikkan jenis dari setiap grup data. Pada metode ini vector input akan dipersiapkan namun tidak ada pendefenisian vektor target. Jaringan memodifikasi nilai *weights* sehingga input yang sama akan di dihubungkan dengan output yang sama atau klaster (unit). (Fausett & Fausett, 1994)

Pelatihan dengan metode supervisi lebih banyak dan digunakan dan terbukti cocok dipakai dalam berbagai aplikasi, akan tetapi kelemahan utama pelatihan dengan pengawasan adalah waktu komputasinya yang lama dikarenakan b*error*der eksponensial ini berarti data pelatihannya cukup banyak, sehungga prosesnya sangat lambat.(Fausett & Fausett, 1994)

## 2.11. Back Propagation Neural Network

*Back Propagation* adalah salah satu model JST yang mempunyai kemampuan mendapatkan keseimbangan antara kemampuan jaringan untuk mengenali pola yang digunakan selama pelatihan serta kemampuan jaringan untuk memberikan respons yang benar terhadap pola masukan yang serupa (tapi tidak sama) dengan pola yang dipakai selama pelatihan (Kusmaryanto, 2014).

*Back Propagation* memiliki beberapa unit neuron yang ada dalam satu atau lebih lapisan tersembunyi. Arsitektur *Back Propagation* dengan n buah masukan (ditambah dengan satu bias), dan sebuah layar tersembunyi yang terdiri dari p unit (ditambah satu bias), dan juga m buah unit keluaran. Arsitektur Jaringan *Back Propagation* seperti tampak pada gambar 2.17.




Gambar 2.17 Arsitektur Network Back Propagation Neural Network (Zhenliang Liu,2019)

Dalam *Back Propagation*, fungsi aktivasi yang dipakai harus memenuhi beberapa syarat yaitu: kontinyu, terdiferensial dengan mudah dan merupakan fungsi yang tidak turun. Fungsi yang dipakai adalah *sigmoid biner, tansig, dan purelin.* (Kusmaryanto, 2014).

Penentuan nilai *weights* awal pada BPNN ditentukan sendiri oleh pengguna, kemudian apabila nilai keluaran prediksi dari BPNN dan target terdapat perbedaan maka terdapat *error*, sehingga nilai *weights* harus diperbaiki, nilai *weights* diperbaiki menggunakan proses *Back Propagation*. Dalam proses *Back Propagation* akan dilakukan proses perbaikan nilai *weights* proses perbaikan itu dilakukan pada setiap layer, dengan tahapan sebagai berikut.

1. Menghitung *Error Signal* dari setiap nodes

   *Error signal* dihitung untuk digunakan untuk perhitungan selanjutnya, *error signal* dihitung dengan menggunakan rumus 2.20 sebagai berikut

   $$\delta_n = f'(tot_n)(t - o_n) = o_n(1 - o_n)(t - o_n) \qquad (2.20)$$

   dengan $\delta_n$ adalah *error signal* di *nodes* n, $t$ adalah target yang diinginkan, $o_n$ adalah output dari *nodes* n.
2. Menghitung nilai perubahan *weight* untuk setiap nodes

   $$\Delta w_{nj} = \eta \delta_n o_j \qquad (2.21)$$

   $$w_{nj}new = w_{nj}old + \Delta w_{nj} \qquad (2.22)$$

   Dengan $\Delta w_{nj}$ adalah nilai perubahan *weight* yang menghubungkan *node* n dan *node* j, $w_{nj}new$ adalah nilai *weight* baru yang menghubungkan *node* n dan *node* j, dan $w_{nj}old$ adalah nilai *weight* lama yang menghubungkan *node* n dan *node* j. Persamaan untuk menghitung $\Delta w_{nj}$ dan $w_{nj_{new}}$ dihitung berdasarkan persamaan 2.21 dan 2.22.
3. Lakukan dua tahapan di atas pada setiap layer dan *nodes* pada *Neural Network.*

   Apabila setiap *nodes* pada setiap layer telah di perbaiki nilai *weights*-nya maka tahapan *forward propagation* dan *backward propagation* ini disebut 1 *Epoch*, kemudian proses akan dilanjutkan ke *Epoch* ke-2. Kemudian akan dilakukan *forward propagation* kembali dan apabila masih terdapat *error* yang

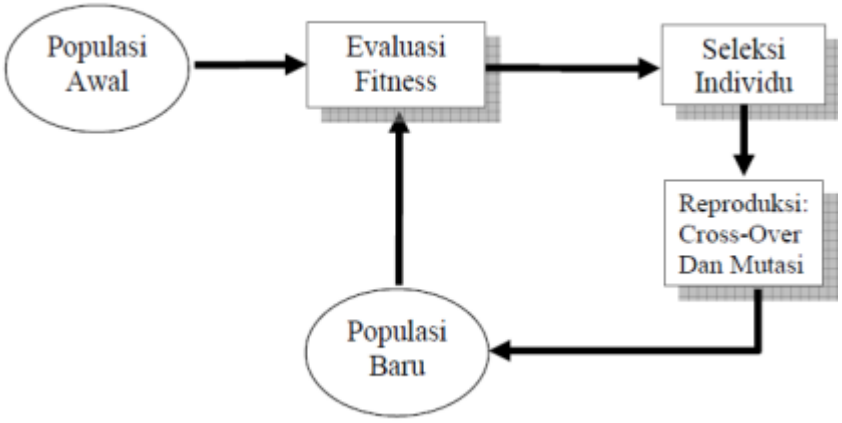
masih diluar nilai toleransi *error* atau *stopping criterion*, maka akan dilakukan kembali backpropagation.

Analisa performa dari Neural Network dilihat dari MSE *(Mean Squared Error)* yang dihitung dari perbedaan nilai output dan target. MSE dihitung dengan persamaan 2.23 berikut

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(t_n - o_n) \quad (2.23)$$

Dengan n adalah jumlah output *nodes*, $t_n$ adalah target di output *nodes* n dan $o_n$ adalah nilai prediksi di output *nodes* n.

## 2.12. Genetic Algorithm

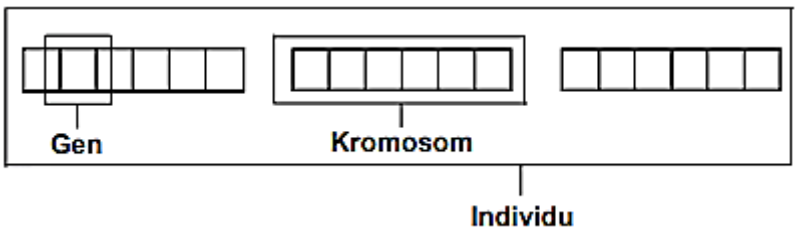



Gambar 2.18 Flowchart Algoritma *Genetic Algorithm*

Algoritma genetika ditemukan oleh John Holland dan dikembangkan oleh muridnya David Goldberg. Algoritma genetika adalah algoritma yang memanfaatkan proses seleksi alamiah yang dikenal dengan proses evolusi. Dalam proses evolusi, individu akan secara terus menerus mengalami perubahan gen untuk menyesuaikan dengan lingkungan hidupnya. Gambar 2.18 menunjukkan alur proses terjadinya *Genetic Algorithm*.

Proses seleksi alamiah ini melibatkan perubahan gen yang terjadi pada individu melalui proses perkembang-biakan. Dalam algoritma genetika ini, proses perkembang-biakan menjadi suatu proses dasar untuk mendapatkan hasil atau keturunan yang baru. Algoritma Genetika adalah bagian dari algoritma evolusi *(evolutionary algorithms, EAs*) merupakan sub-set dari komputasi evolusi (*evolutionary computation, EC*) yang merupakan bentuk generik dari algoritma optimasi *meta-heuristic* berbasis populasi. (Reeves, 2010)

### 2.12.1. Kromosom atau Individu




Gambar 2.19 Kromosom dan Gen dalam Genetic Algorithm

Kromosom atau Individu merupakan bagian dari *genetic algorithm* yang penting, karena pemodelan kromosom akan mempengaruhi kualitas solusi yang diberikan. Kromosom merupakan representasi dari masalah yang akan diselesaikan. Visualisasi dari suatu individu, kromosom dan gen dapat dilihat pada gambar 2.19.

Dalam *genetic algorithm* ada beberapa bentuk representasi dari kromosom mulai dari representasi biner, representasi integer, representasi *real code*, dan representasi permutasi. Setiap anggota kromosom disusun oleh gen-gen dimana masing-masing gen mewakili elemen dari solusi, dengan dibangkitkannya populasi ini, maka akan tersedia banyak pilihan solusi.(Jocom, 2017)

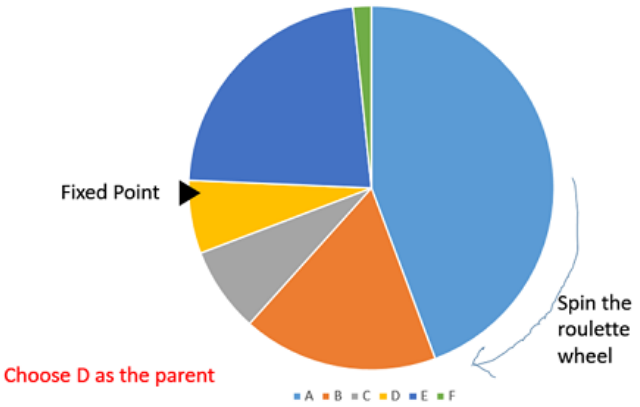
### 2.12.2. Fitness

*Fitness* merupakan suatu model bentuk untuk mengetahu seberapa besar nilai dari suatu kromosom. Fungsi *fitness* dapat berhubungan langsung dengan fungsi objektif. Solusi-solusi akan dievaluasi menggunakan fungsi *fitness*. Fungsi *fitness* yang biasa digunakan adalah fungsi $F(x) = \frac{1}{f(x)+1}$, dimana f(x) adalah fungsi objektif masalah yang akan diselesaikan. Untuk kasus maksimasi maka dicari F(x) yang bernilai besar, sebaliknya apabila membutuhkan kasus minimasi, dicari nilai F(x) yang kecil. Fungsi objektif dapat menggunakan nilai f(x) sendiri yaitu $F(x) = f(x)$

### 2.12.3. Elitisme

Konsep dasar dari elitisme dalam GA adalah mempertahankan individu yang baik dan membuang individu yang kurang baik. Dalam memilih kromosom elit digunakan beberapa metode diantaranya *Tournament* dan *Roulette Wheel*.

### 2.12.4. Seleksi Dengan Roulette Wheel




| Chromosome | Fitness Value |
|---|---|
| A | 8.2 |
| B | 3.2 |
| C | 1.4 |
| D | 1.2 |
| E | 4.2 |
| F | 0.3 |

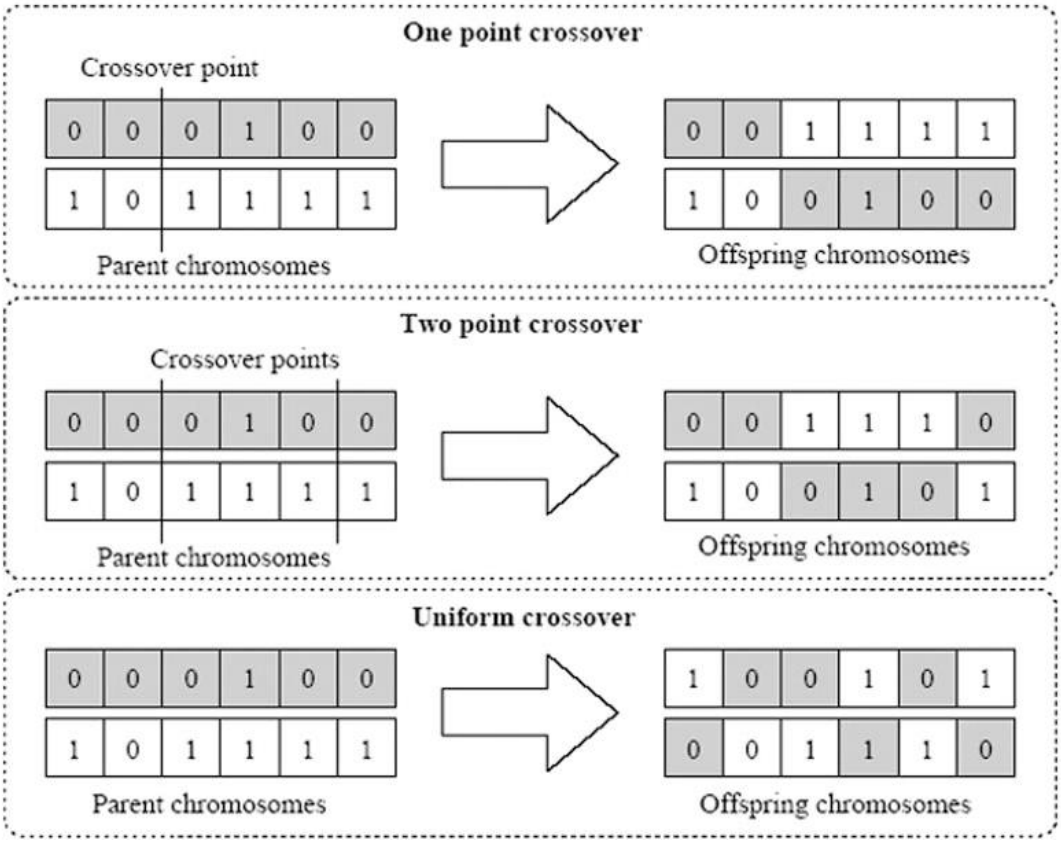
Gambar 2.20 Seleksi dengan *Roulette Wheele* (https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_parent_selection.htm)

Dalam seleksi dengan Roulette Wheel semua individu memiliki kesempatan yang sama, setiap individu akan dicari nilai probabilitasnya dengan menggunakan *fitness function*-nya. Untuk setiap individu akan direpresentasikan dengan luas daerah pada lingkaran *roulette wheele* seperti ditampilkan pada gambar 2.20.

Untuk setiap individu atau solusi akan dibangkitkan bilangan random, bilangan random ini akan di cocokkan dengan *range*, individu yang terpilih akan menjadi *elit.* Dengan nilai *fitness* yang besar akan memiliki luasan daerah yang

lebih luas (dalam konsep *smaller is better*) sehingga memiliki kesempatan untuk terpilih lebih besar.

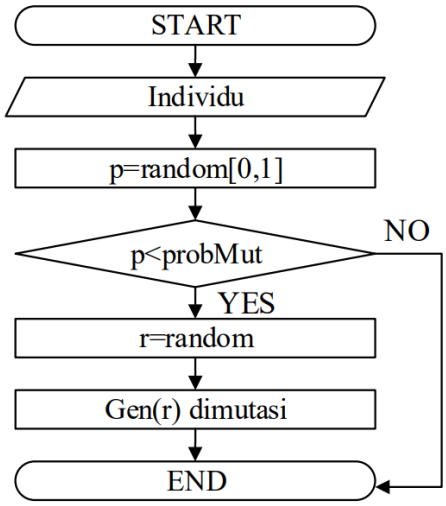
### 2.12.5. Cross-over




Gambar 2.21 Proses *Crossover* dalam *Genetic Algorithm* (Senaratna, 2005)

*Crossover* merupakan salah satu operasi dalam algoritma genetika yang melibatkan induk (elit) untuk menghasilkan keturunan yang baru. Diharapkan dengan melakukan *cross-over* maka keturunan akan mewarisi sifat dari induknya. Proses *crossover* dilakukan pada setiap individu dengan probabilitas *cross-over* ditentukan, *crossover* dilakukan dengan melakukan penyilangan antara gen dalam satu kromosom induk dengan kromosom induk lainnya. Terdapat beberapa metode untuk melakukan *cross-over*, yaitu *one point, two point, uniform,* dan lain-lain. Visualisasi proses *crossover* dapat dilihat pada gambar 2.21. (Senaratna, 2005)

### 2.12.6. Mutasi




Gambar 2.22 Flowchart proses mutasi dalam *Genetic Algorithm*.

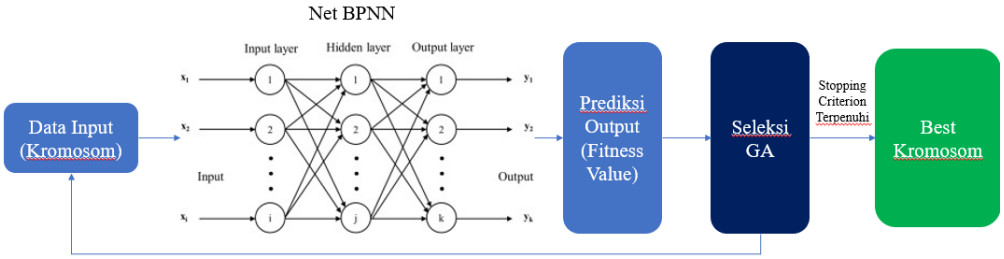
Mutasi dimaksudkan untuk memunculkan individu yang baru sama sekali belum ada dan benar-benar berbeda dengan individu yang sudah ada. Diagram alir proses mutasi dapat dilihat pada gambar 2.22.

Dalam konteks optimasi mutasi dilakukan untuk memunculkan solusi baru dan dapat keluar dari *local optimum*. Mutasi dilakukan dengan probabilitas mutasi yang ditentukan. Semakin besar probabilitas, semakin besar jumlah gen yang termutasi. Mutasi dilakukan dengan menggeser nilai gen pada gen yang terpilih untuk dimutasikan. Gen yang terpilih dan besar nilai mutasi dilakukan secara acak.

## 2.13. *Back Propagation Neural Network-Genetic Algorithm* (BPNN-GA)

BPNN-GA adalah metode yang menggabungkan dua metode metaheuristik BPNN dan GA. Metode ini bekerja dengan BPNN mengeluarkan suatu fungsi yang merepresentasikan hubungan *input-output*, dimana fungsi tersebut akan digunakan sebagai fungsi objektif pada proses GA. Metode BPNN-GA dapat dilihat pada gambar 2.23.

Proses ini akan mencari nilai maksimum atau minimum dari *output* yang dikeluarkan oleh BPNN, berdasarkan *input* populasi yang akan dicari nilai optimumnya. Diagram alir BPNN-GA dapat dilihat pada Bab 3.




Gambar 2.23 Visualisasi Metode BPNN-GA

Algoritma *Genetic Algorithm* juga dapat digunakan untuk mengoptimasi Jaringan *Neural Network* untuk minimalisasi nilai MSE pada jaringan *Neural Network*. Hal ini dilakukan agar proses pemilihan arsitektur atau topologi *Neural Network* tidak melalui

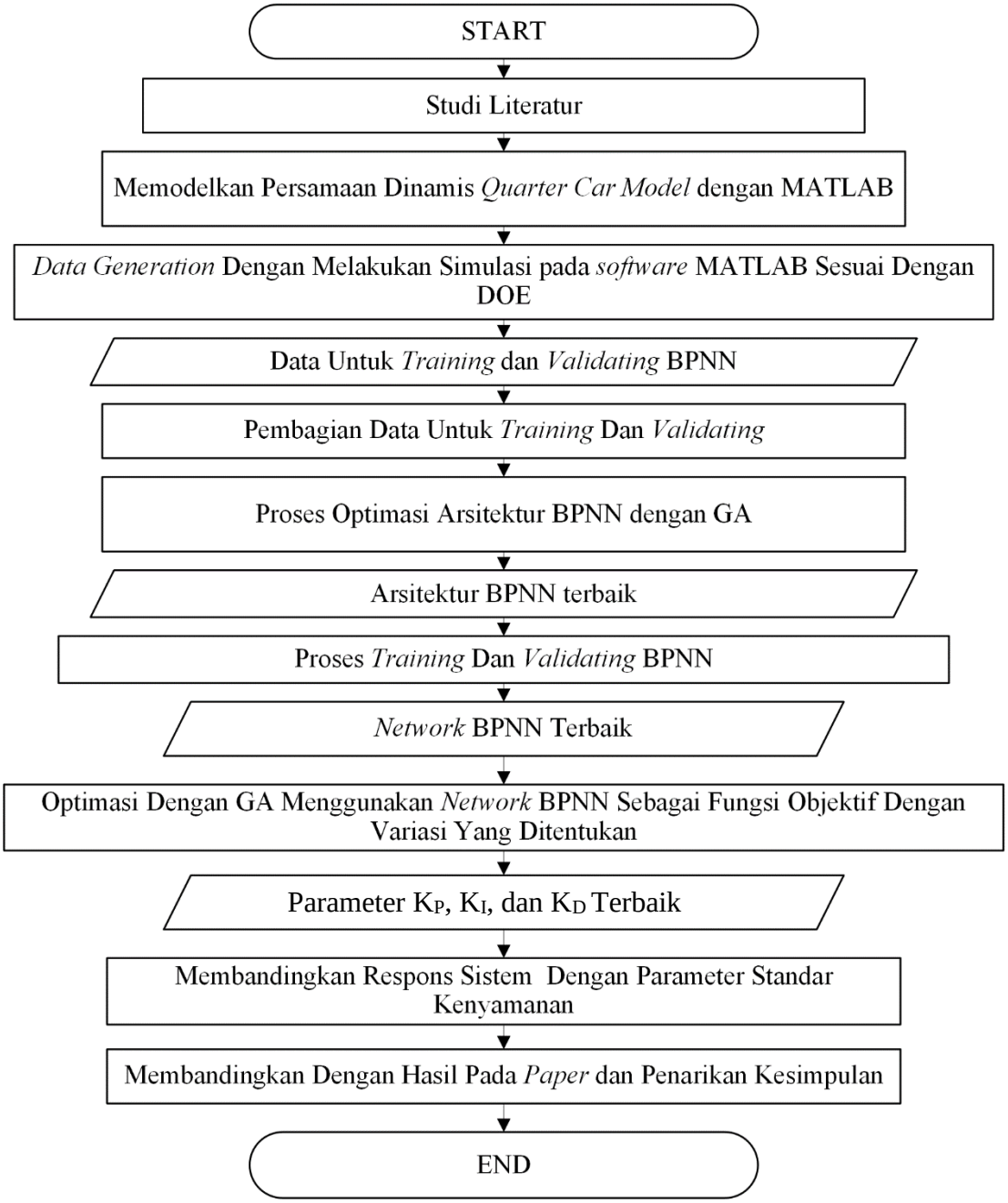
proses *trial and error* demi mendapatkan nilai MSE terkecil. Topologi yang dioptimasi adalah *activation function*, jumlah *node*, dan jumlah *hidden layer* dari *Neural Network*.

(*Halaman ini sengaja dikosongkan*)

# BAB III
# METODOLOGI PENELITIAN

## 3.1. Diagram Alir

Aktivitas yang akan dilakukan dalam penelitian ini digambarkan pada diagram alir penelitian pada Gambar 3.1.




Gambar 3.1 Diagram Alir Tugas Akhir

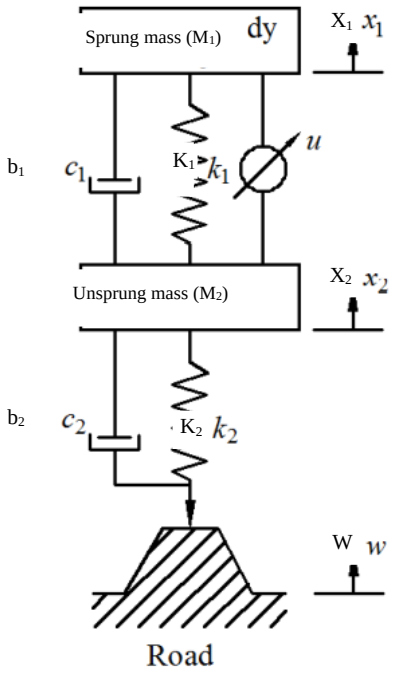
### 3.1.1. Studi Literatur

Langkah pertama pada penelitian ini adalah studi literatur. Studi literatur bertujuan untuk mendalami landasan teori yang akan digunakan pada penelitian

ini. Studi literatur dimulai dari mencari jurnal-jurnal penelitian sebelumnya yang meneliti tentang topik yang sejenis, yaitu mengenai, sistem suspensi aktif, pengendali PID, *Artificial Neural Network* (ANN), *Genetic Algorithm* (GA), dan penelitian mengenai optimasi Pengendali PID dengan menggunakan *Artificial Intelligence*.

### 3.1.2. Pemodelan Sistem Suspensi Aktif Dengan Pengendali PID

Pemodelan pada sistem suspensi aktif dengan pengendali PID, dilakukan pada *software* MATLAB, sistem suspensi aktif dengan model seperempat kendaraan digunakan untuk menyederhanakan sistem. Model ini adalah model suspensi aktif dengan aktuator dengan gaya kontrol U untuk mengendalikan gerakan dari badan kendaraan, dimodelkan berdasarkan persamaan sebagai berikut.




Gambar 3.2 Model ¼ kendaraan (Pekgökgöz et al., 2010)

Dari Gambar 3.2. berdasarkan hukum Newton, kita dapat mendapatkan persamaan gerak yang dirumuskan pada persamaan 3.1 dan 3.2.

$$M_1\ddot{X}_1 = -b_1(\dot{X}_1 - \dot{X}_2) - K_1(X_1 - X_2) + U \quad (3.1)$$

$$M_2\ddot{X}_2 = b_1(\dot{X}_1 - \dot{X}_2) + K_1(X_1 - X_2) + b_2(\dot{W} - \dot{X}_2) + K_2(W - X_2) - U \quad (3.2)$$

Dari persamaan 3.1 dan 3.2 kita dapat membuat model fungsi transfer dengan mengubah domain persamaan menjadi domain s, seperti ditunjukkan pada persamaan 3.3 dan 3.4.

$$(M_1s^2 + b_1s + K_1)X_1(s) - (b_1s + K_1)X_2(s) = U(s) \quad (3.3)$$

$$-(b_1s + K_1)X_1(s) + \left(M_2s^2 + (b_1 + b_2)s + (K_1 + K_2)\right)X_2(s) = (b_2s + K_2)W(s) - U(s) \quad (3.4)$$

Setelah mendapatkan persamaan dalam domain s kemudian persamaan dibentuk dalam model matriks dan dikelompokkan seperti persamaan 3.5

$$\begin{bmatrix} (M_1s^2 + b_1s + K_1) & -(b_1s + K_1) \\ -(b_1s + K_1) & \left(M_2s^2 + (b_1 + b_2)s + (K_1 + K_2)\right) \end{bmatrix} \begin{bmatrix} X_1(s) \\ X_2(s) \end{bmatrix} = \begin{bmatrix} U(s) \\ (b_2s + K_2)W(s) - U(s) \end{bmatrix} \quad (3.5)$$

$$A = \begin{bmatrix} (M_1s^2 + b_1s + K_1) & -(b_1s + K_1) \\ -(b_1s + K_1) & \left(M_2s^2 + (b_1 + b_2)s + (K_1 + K_2)\right) \end{bmatrix}$$

$$\Delta = \det \begin{bmatrix} (M_1s^2 + b_1s + K_1) & -(b_1s + K_1) \\ -(b_1s + K_1) & \left(M_2s^2 + (b_1 + b_2)s + (K_1 + K_2)\right) \end{bmatrix}$$

atau

$$\Delta = (M_1s^2 + b_1s + K_1) \cdot \left(M_2s^2 + (b_1 + b_2)s + (K_1 + K_2)\right) - (b_1s + K_1) \cdot (b_1s + K_1) \quad (3.6)$$

Dengan memindahkan matriks A ke sisi kanan seperti persamaan 3.7, dan mengelompokkan matriks U(s) dan W(s) maka dibentuk persamaan seperti persamaan 3.8.

$$\begin{bmatrix} X_1(s) \\ X_2(s) \end{bmatrix} = \frac{1}{\Delta} \begin{bmatrix} \left(M_2s^2 + (b_1 + b_2)s + (K_1 + K_2)\right) & (b_1s + K_1) \\ (b_1s + K_1) & (M_1s^2 + b_1s + K_1) \end{bmatrix} \begin{bmatrix} U(s) \\ (b_2s + K_2)W(s) - U(s) \end{bmatrix} \quad (3.7)$$

$$\begin{bmatrix} X_1(s) \\ X_2(s) \end{bmatrix} = \frac{1}{\Delta} \begin{bmatrix} (M_2s^2 + b_2s + K_2) & (b_1b_2s^2 + (b_1K_2 + b_2K_1)s + K_1K_2 \\ -M_1s^2 & (M_1b_2s^3 + (M_1K_2 + b_1b_2)s^2 + (b_1K_2 + b_2K_1)s + K_1K_2) \end{bmatrix} \begin{bmatrix} U(s) \\ W(s) \end{bmatrix} \quad (3.8)$$

Parameter sistem yang digunakan pada penelitian ini diambil dari *paper "Stochastic Algorithm for PID Tuning Of Bus Suspension System"* oleh A.Karthikraja, dkk. (2009). Berikut parameter yang digunakan

Berat ¼ kendaraan = 2500 kg

Berat suspensi = 320 kg

$k_1$ = 80000 N/m

$k_2$ = 500000 N/m

$b_1$ = 350 Ns/m

$b_2$ = 15020 Ns/m

u = gaya kontrol

Diasumsikan semua kondisi awal adalah 0, sehingga persamaan ini merepresentasikan situasi sesaat kendaraan menuju suatu gangguan di jalan. Persamaan dinamis dari persamaan dapat ditulis dalam bentuk *transfer function* dengan melakukan transformasi *Laplace*. Penurunan dari persamaan di atas adalah *transfer function* $G_1(s)$ dan $G_2(s)$ dengan *output* $X_1 - X_2$, dan input U dan W. $X_1$-W sangat sulit untuk diukur dan deformasi dari ban $X_2$-W dapat diabaikan, sehingga penelitian akan difokuskan dengan *output* $X_1$-$X_2$, yang akan

diteliti responsnya. *Road Disturbance* disimulasikan dengan *step input* sebesar 0,1 m yang dapat menggambarkan lubang di jalan. (Arumugam et al., 2009)

Ketika kita ingin melihat *input* pengendali U(s) saja, kita membuat W(s)=0. Sehingga kita mendapatkan *transfer function* $G_1(s)$ seperti persamaan 3.9 berikut

$$G_1(s) = \frac{X_1(s)-X_2(s)}{U(s)} = \frac{(M_1+M_2)s^2+b_2s+K_2}{\Delta} \quad (3.9)$$

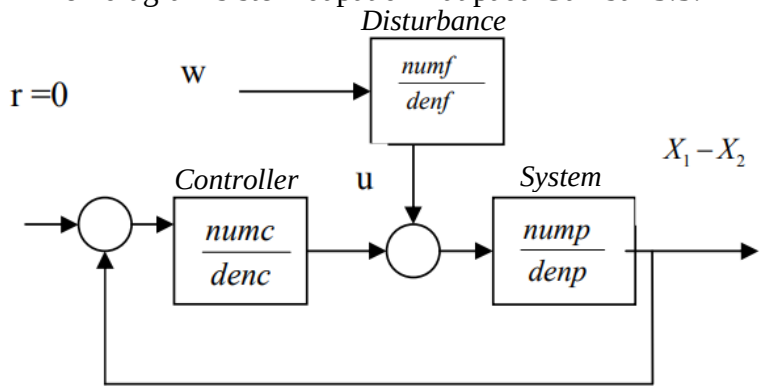
Disaat kita ingin melihat input dari *Road Disturbance* W(s), maka kita membuat U(s)=0. Sehingga kita mendapatkan *transfer function* $G_2(s)$ seperti persamaan 3.10 berikut.

$$G_2(s) = \frac{X_1(s)-X_2(s)}{W(s)} = \frac{-M_1b_2s^3-M_1K_2s^2}{\Delta} \quad (3.10)$$

Dengan memperhitungkan nilai gain proporsional, integral dan derivatif, $K_P$, $K_I$, $K_D$, dalam *transfer function*, persamaan dasar dari Pengendali PID disajikan pada persamaan 3.11.

$$K_P + \frac{K_I}{S} + K_D s = \frac{K_D s^2+K_P s+K_I}{s} \quad (3.11)$$

Blok diagram sistem dapat dilihat pada Gambar 3.3.




$$\frac{numc}{denc} = K_P + \frac{K_I}{S} + K_D s = \frac{K_D s^2+K_P s+K_I}{s} \quad (3.12)$$

$$\frac{nump}{denp} = G_1(s) = \frac{X_1(s)-X_2(s)}{U(s)} = \frac{(M_1+M_2)s^2+b_2s+K_2}{\Delta} \quad (3.13)$$

$$\frac{numf}{denf} = \frac{G_2(s)}{G_1(s)} = \frac{\mathrm{U(s)}}{W(s)} = \frac{(M_1+M_2)s^2+b_2s+K_2}{-M_1b_2s^3-M_1K_2s^2} \quad (3.14)$$

Gambar 3.3 Blok Diagram Sistem Kontrol (Arumugam et al., 2009)

Fungsi transfer untuk pengendalu, sistem, dan *disturbance*, dibentuk pada persamaan 3.12-3.14

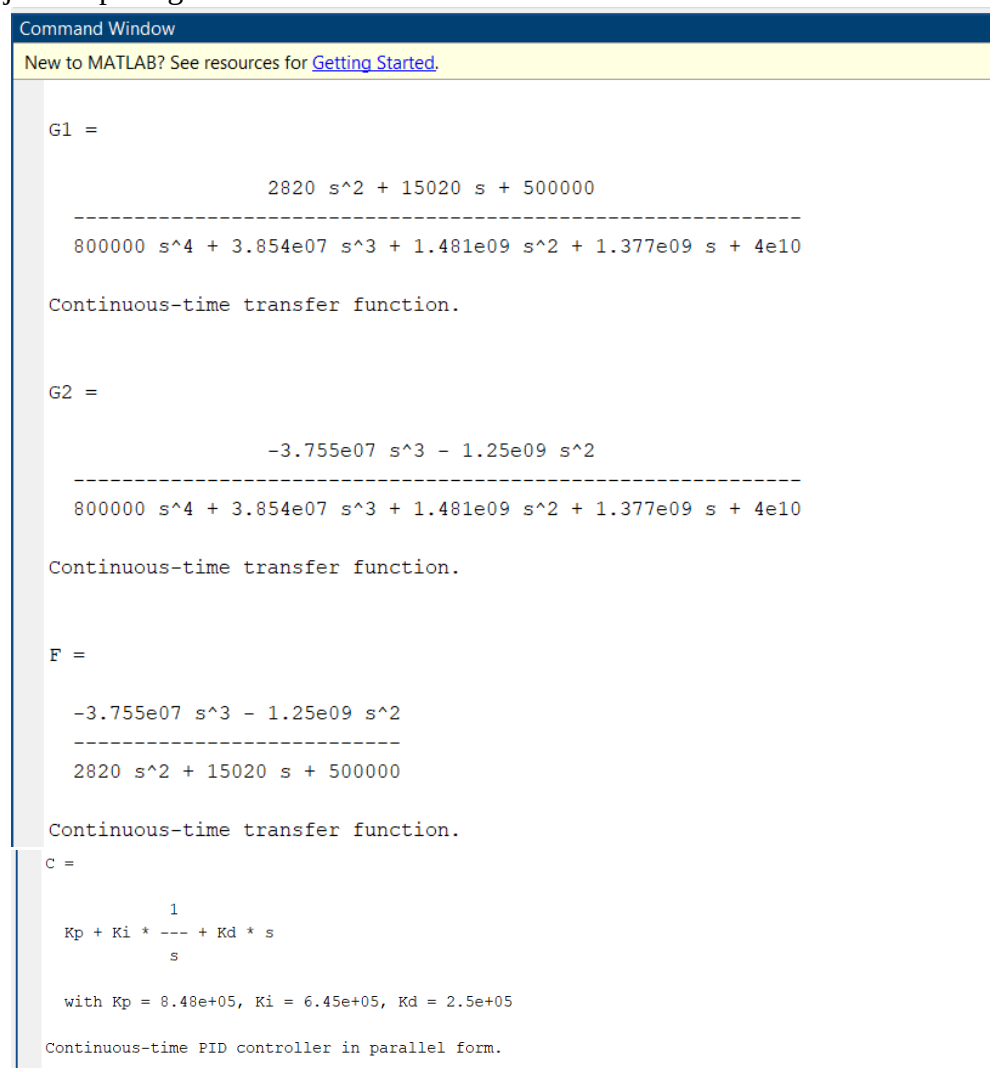
### 3.1.3. Pemodelan Sistem Pada *Software* MATLAB

Dalam pemodelan model persamaan dinamis *quarter car model* pada *software* MATLAB R2021b, *transfer function* yang telah diturunkan pada subbab 3.1.2. dimodelkan dalam sebuah sistem, yang direpresentasikan oleh kode berikut.

```
%system properties
m1 = 2500;
m2 = 320;
k1 = 80000;
k2 = 500000;
b1 = 350;
b2 = 15020;
```

```
%time step
t=0:0.005:10;

%pembentukan transfer function
nump=[(m1+m2) b2 k2];
denp=[(m1*m2) (m1*(b1+b2))+(m2*b1) (m1*(k1+k2))+(m2*k1)+(b1*b2)
(b1*k2)+(b2*k1) k1*k2];
G1=tf(nump,denp)

num1=[-(m1*b2) -(m1*k2) 0 0];
den1=[(m1*m2) (m1*(b1+b2))+(m2*b1) (m1*(k1+k2))+(m2*k1)+(b1*b2)
(b1*k2)+(b2*k1) k1*k2];
G2=tf(num1,den1)

numf=num1;
denf=nump;
F=tf(numf,denf)
```

Sistem G1, G2, F dan C akan terbentuk *transfer function*-nya dan memberikan keluaran seperti ditunjukkan pada *command window*, seperti di tunjukkan pada gambar 3.4.




Gambar 3.4 Keluaran *Transfer Function* G1, G2 dan C Pada *Workspace* di MATLAB

Kemudian *transfer function* yang sudah terbentuk akan digunakan untuk membentuk sistem (sys_c1), dengan *disturbance* F, sistem G1 dan di-*feedback* dengan pengendali C. Transfer function sys_c1 dapat dilihat keluarannya pada *command window* MATLAB seperti gambar 3.5.

```
%pembentukan sistem dengan input step dan Pengendali PID C
Kp= a
Ki= b
Kd= c
C=pid(Kp,Ki,Kd)
sys_cl=F*feedback(G1,C);
```

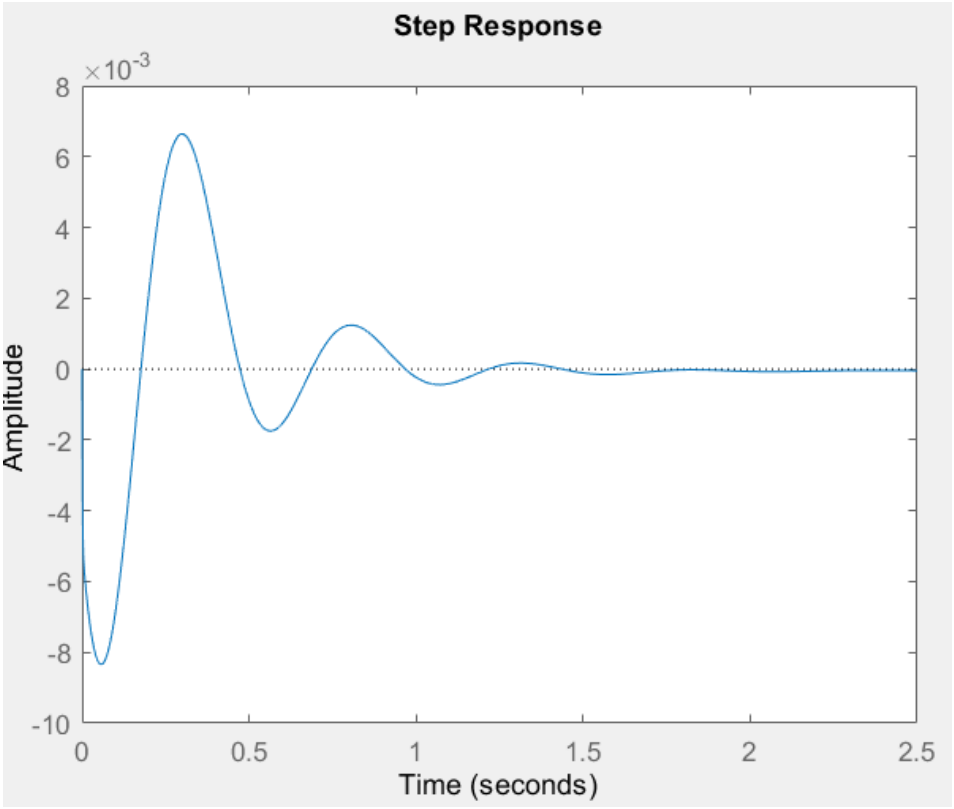
```
sys_cl =

                          -1.059e11 s^6 - 4.089e12 s^5 - 3.755e13 s^4 - 6.25e14 s^3

  ------------------------------------------------------------------------------------------------------------------

  2.256e09 s^7 + 2.108e12 s^6 + 3.308e13 s^5 + 8.836e14 s^4 + 7.267e15 s^3 + 7.848e16 s^2 + 2.418e17 s

                                                                                                + 1.613e17


Continuous-time transfer function.
```

Gambar 3.5 Keluaran *Transfer Function* Sistem (sys_c1) Pada *Workspace* di MATLAB

Setelah sistem sys_c1 telah terbentuk maka sistem akan diberikan *disturbance* berupa *input step* sebesar 0,1 m. Sistem kemudian akan dilihat grafik responsnya dan dapat dievaluasi nilai *Settling Time* dan *Peak Overshoot*-nya. Kode untuk mensimulasikan respons dari sistem dapat dimodelkan dengan kode berikut.

Keluaran dari simulasi akan menghasilkan grafik *step response* yang ditunjukkan oleh gambar 3.6.

```
%menampilkan step response dan mendefinisikan titik respon
[y,t] = step(0.1*sys_cl,t);
```



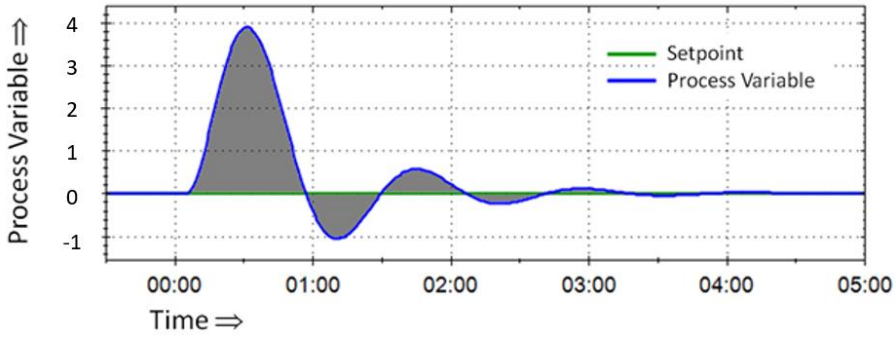

Gambar 3.6 Keluaran Grafik *Step Response* Sistem

### 3.1.4. Menghitung Nilai ITAE *(Integral Timed Absolute Error)*

Setelah mendapatkan pemodelan dan grafik respons dari sistem, kita dapat menghitung nilai ITAE-nya. Nilai ITAE tergantung dari respons dari suatu sistem dibanding dengan *setpoint*-nya.

$$ITAE = \int_0^t t|e(t)| \quad dt \tag{3.15}$$

Perhitungan ITAE *(Integral Timed Absolute Error)* dilakukan dengan melakukan penjumlahan nilai perkalian nilai waktu dan selisih mutlak antara *setpoint* dan posisi benda *(error)* di setiap waktu t. Perhitungan nilai ITAE dapat pula diartikan sebagai integral dari 0 sampai t dari nilai *error* dikali dengan variabel t, sesuai dengan persamaan 3.15.



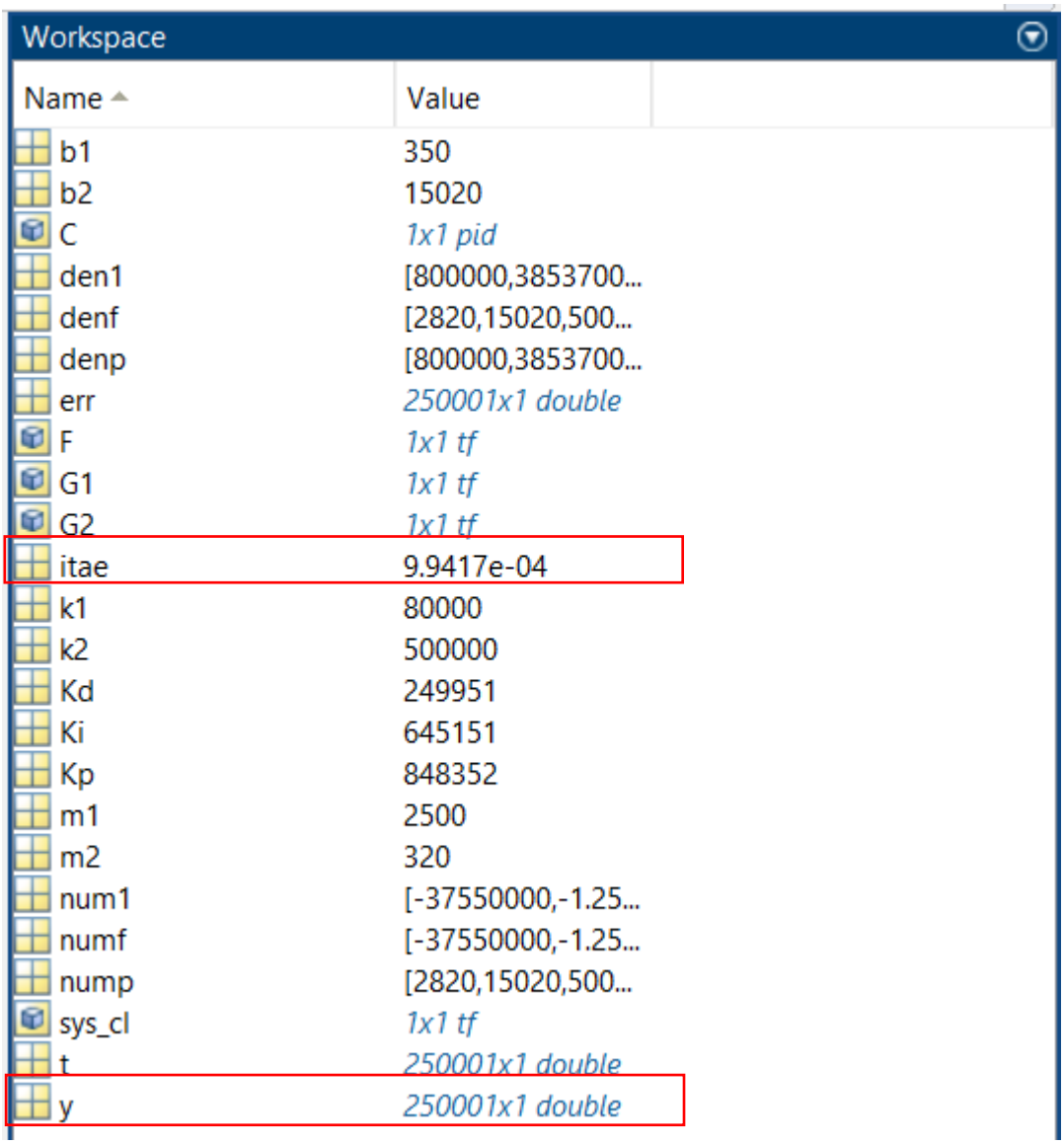

Gambar 3.7 Visualisasi Nilai Integral *Error* Pada Grafik Respons (blog.opticontrols.com)

Visualisasi dari integral *error* dapat dilihat di Gambar 3.7. Pada MATLAB, perhitungan ITAE dapat dimodelkan dengan kode berikut

```
t=0:0.0001:10;
```

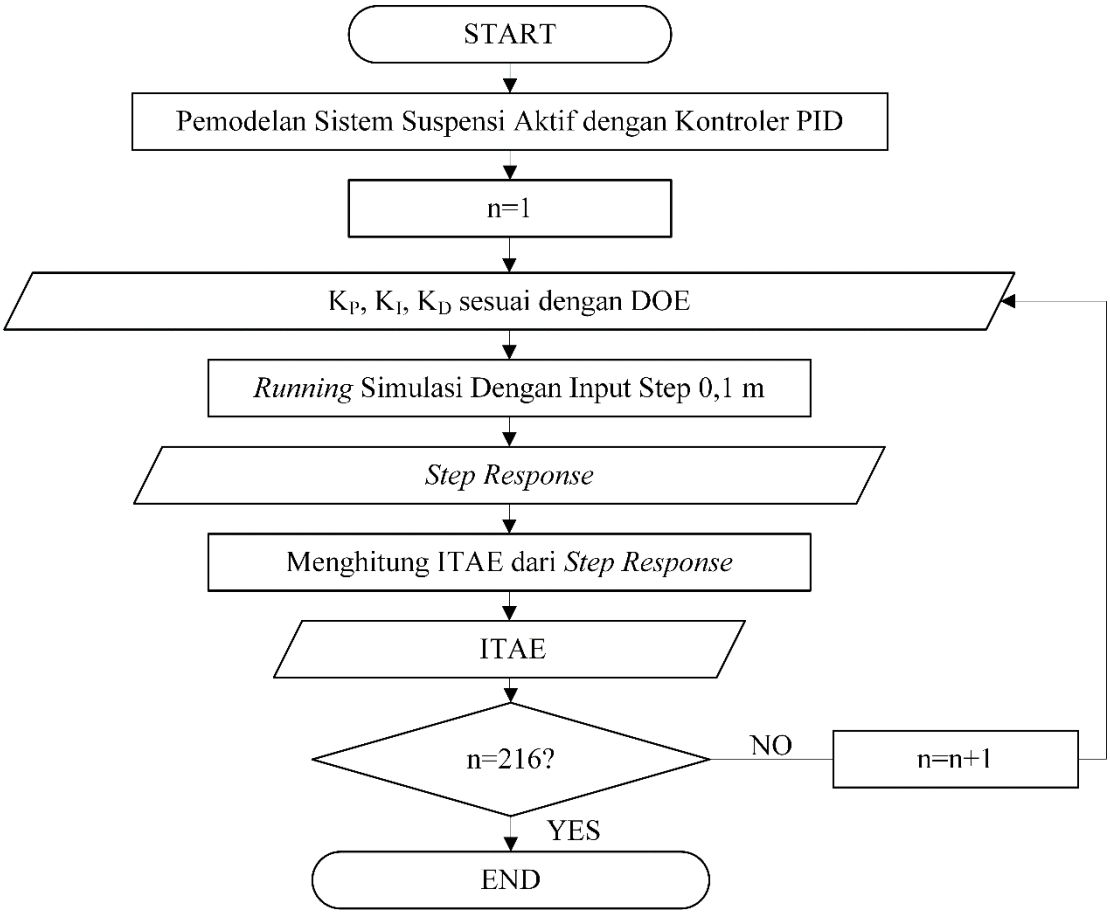
```
%mendefinisikan titik respon
[y,t] = step(0.1*sys_cl,t);

%mendefinisikan error
err = 0-y; %menghitung error
%mennghitung ITAE IAE ISE
itae = trapz(t,t.*abs(err));
```

ITAE dapat dihitung dari penjumlahan hasil pengurangan variabel y dengan *setpoint* disetiap titik waktu. Hasilnya dapat dilihat pada *workspace* yang dapat dilihat seperti gambar 3.8.




Gambar 3.8 Keluaran Variabel y dan Hasil Perhitungan ITAE dari *Workspace* MATLAB

### 3.1.5. *Data Generation*




Gambar 3.9 Diagram Alir Pengambilan Data di *Software* MATLAB

Setelah melakukan pemodelan, grafik respons, dan nilai ITAE, tahap selanjutnya adalah melakukan *data generation.* Proses *data generation* akan dilakukan sesuai dengan diagram alir pada gambar 3.9. Data akan digunakan untuk proses *training* dan *validating* pada *Back Propagation Neural Network.* Data diambil dari proses *looping* simulasi yang akan dilakukan di program MATLAB, simulasi akan dilakukan dengan kondisi sesuai dengan DOE pada tabel 3.1. Jumlah data total yang akan diambil adalah 216 data. Penulis telah melakukan simulasi yang ditabulasi pada tabel 3.2, dengan batasan nilai pelatihan dan optimasi, batas-batas nilai $K_P$, $K_I$, dan $K_D$ yang digunakan adalah *range* nilai $K_P$, $K_I$, dan $K_D$ yang berada disekitar *range* optimal pada *paper "Stochastic Algorithm for PID Tuning Of Bus Suspension System"* oleh A.Karthikraja, dkk. (2009)", penulis merasa diperlukan menggunakan batas agar *range* untuk proses *training* BPNN dan optimasi GA tidak terlalu besar. *Range* data dibatasi oleh persamaan 3.16-3.18

$$800000 \leq K_P \leq 850000 \quad (3.16)$$

$$600000 \leq K_I \leq 650000 \quad (3.17)$$

$$200000 \leq K_D \leq 250000 \quad (3.18)$$

Data yang diambil pada simulasi akan digunakan untuk proses *training* dan *validating* pada BPNN.

Tabel 3.1 *Design Of Experiment* Simulasi

| Level | | | | | | |
|---|---|---|---|---|---|---|
| | **Level 1** | **Level 2** | **Level 3** | **Level 4** | **Level 5** | **Level 6** |
| **Kp** | 800000 | 810000 | 820000 | 830000 | 840000 | 850000 |
| **Ki** | 200000 | 210000 | 220000 | 230000 | 240000 | 250000 |
| **Kd** | 600000 | 610000 | 620000 | 630000 | 640000 | 650000 |

Data hasil simulasi sesuai dengan *Design Of Experiment* yang telah didapatkan ditabulasi pada Lampiran 1**.**

### 3.1.6. Proses Optimasi Arsitektur BPNN dengan Metode GA

Setelah mendapatkan *dataset* tahap selanjutnya adalah menentukan arsitektur *neural network* yang terbaik untuk digunakan dalam proses optimasi nanti. Performa dari arsitektur *neural network* dipengaruhi oleh parameter *hidden layer*, jumlah *nodes* tiap layer, dan fungsi aktivasi. Oleh sebab itu dalam proses optimasi akan mencari parameter *hidden layer*, jumlah *nodes* tiap *layer*, dan fungsi aktivasi yang terbaik untuk mendapatkan nilai MSE paling minimum.

Dalam proses optimasi, pembentukan *net neural network* hanya dilakukan tahap *training* dan tahap *validating*, untuk proses testing akan dilakukan dengan melihat performa prediksi nilai ITAE dari 1000 nilai $K_P$, $K_I$ dan $K_D$ acak. Prediksi dari *net* BPNN akan dibandingkan dengan nilai ITAE riil. Data ITAE riil didapatkan dengan proses pengambilan data melalui simulasi sesuai *flowchart* pada gambar 3.9.

Hubungan antara tiap-tiap parameter yang di optimasi, yaitu *hidden layer*, jumlah *nodes* tiap layer, dan fungsi aktivasi dengan nilai MSE, tidak dapat diprediksi dan tidak ada ketentuan atau rumusan tertentu untuk memilih parameter yang tepat. Selama ini penentuan parameter-parameter tersebut dilakukan dengan *trial and error*, yaitu dengan mencoba setiap kombinasi parameter-parameter tersebut satu persatu. Dari percobaan tersebut kemudian akan dipilih arsitektur terbaik berdasarkan nilai dari MSE *(Mean Squared Error)*.

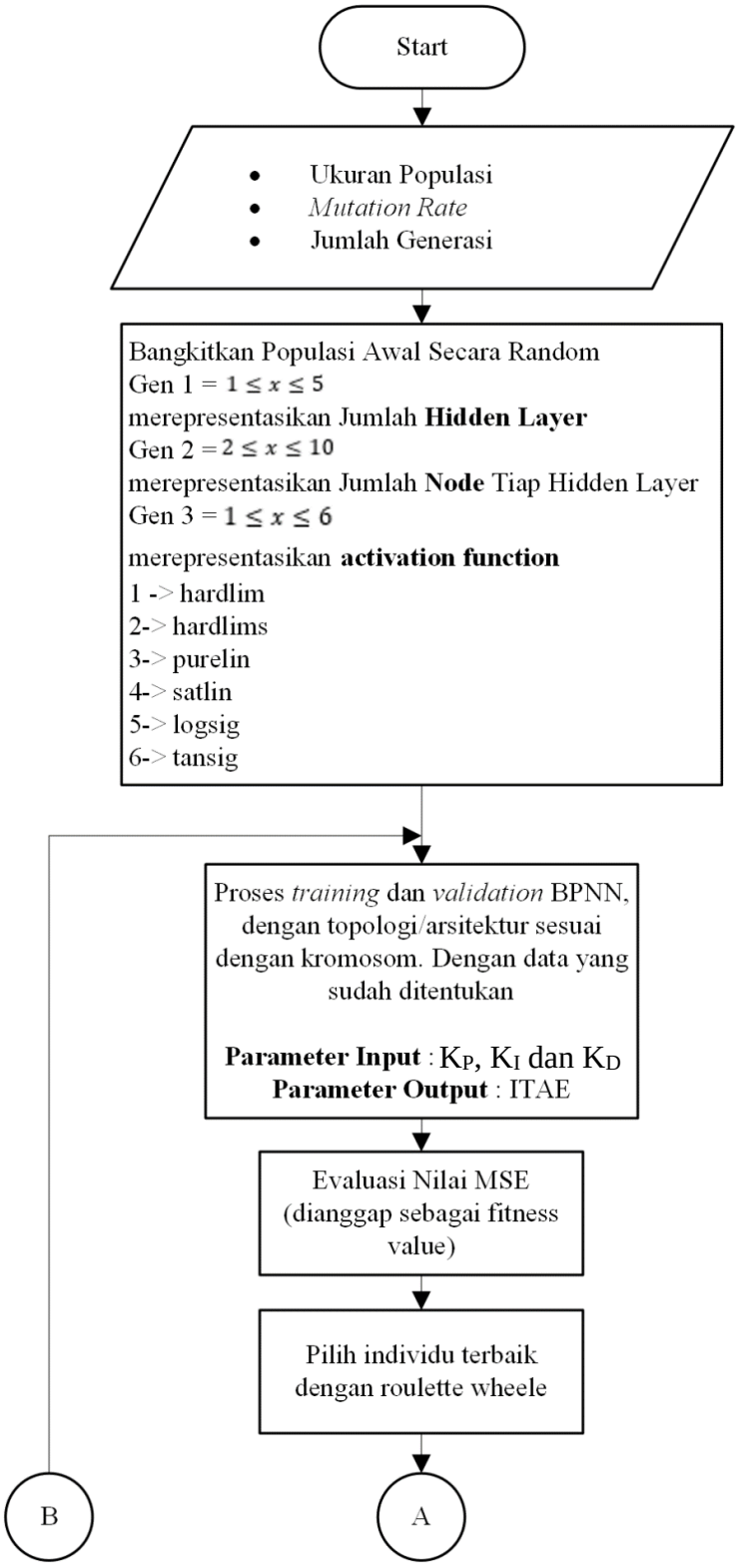
Oleh sebab itu proses optimasi menggunakan GA dirasa cocok digunakan untuk mengoptimasi parameter-parameter ini, dikarenakan setiap parameternya yang tidak diketahui hubungan pastinya. Algoritma dari GA akan mencoba mempelajari hubungan dari tiap parameter untuk mencari kombinasi terbaik.

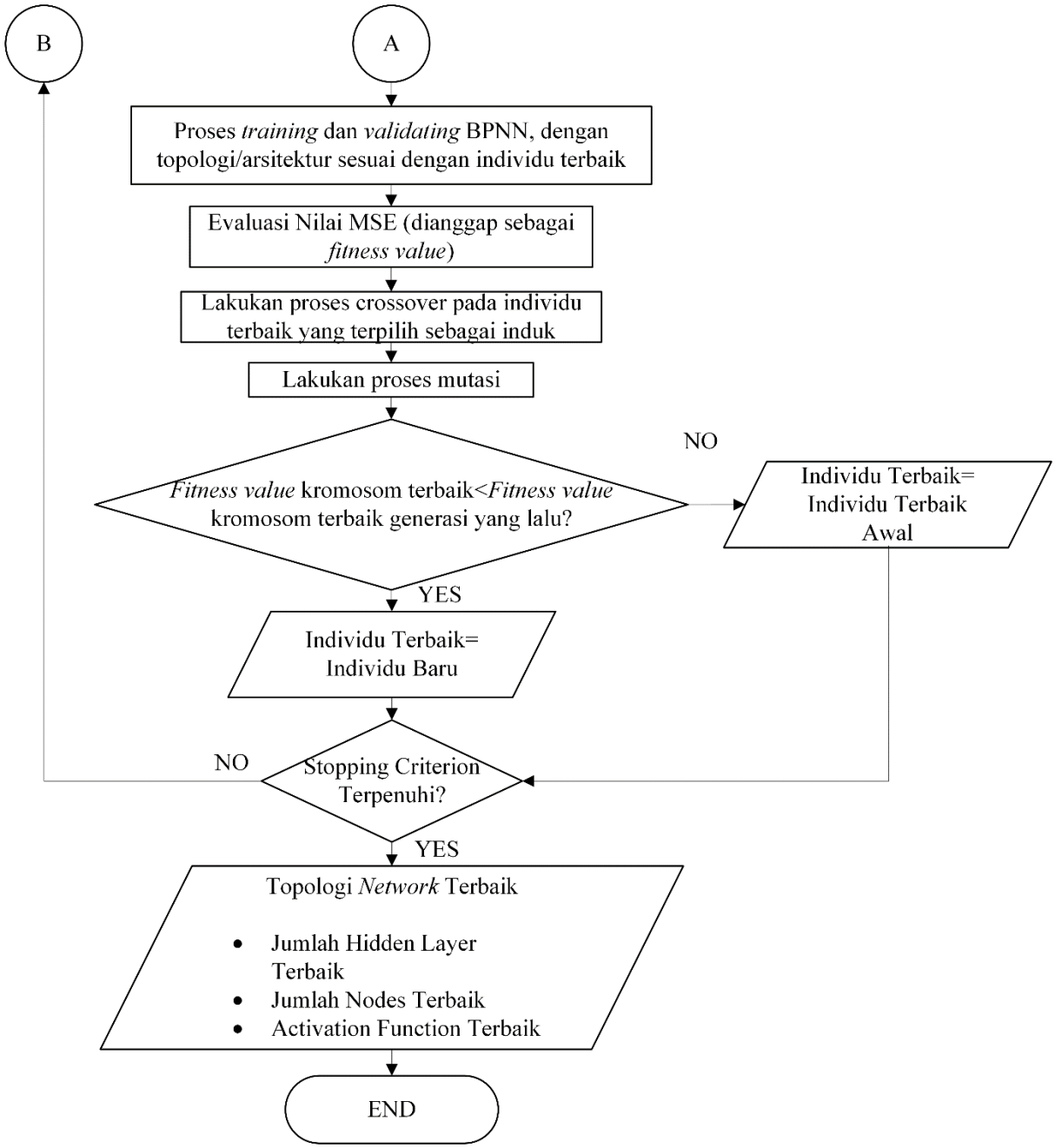
Dalam praktiknya selama ini, untuk mendapatkan parameter arsitektur BPNN yang terbaik, dilakukan dengan cara mencoba seluruh kombinasi parameter arsitektur BPNN yang ada, hal ini tentu saja memakan waktu dan tenaga yang cukup banyak. Oleh sebab itu metode ini diharapkan dapat menjadi suatu terobosan untuk mempersingkat waktu dan mengurangi usaha yang dibutuhkan untuk mendapatkan parameter arsitektur NN yang terbaik.

Proses optimasi arsitektur BPNN dengan menggunakan *Genetic Algorithm* dilakukan sebelum *net* dari BPNN di *training* dan digunakan sebagai *objective function* pada *Genetic Algorithm* pada proses optimasi parameter PID. Proses optimasi parameter BPNN dilakukan untuk mendapatkan topologi jaringan yang terbaik, sehingga memperkecil *error* nilai prediksi yang dikeluarkan oleh BPNN. Dengan nilai *error* BPNN yang kecil dapat memberikan hasil optimasi

yang valid. Proses optimasi akan dilakukan untuk mendapatkan nilai MSE terkecil *(smaller is better)*.

Dalam proses optimasi ini data untuk *training* dan *validating* sudah ditentukan porsinya yaitu 80% dan 20%. Proses *selection* dilakukan dengan metode, *roulette-wheel, crossover* dengan metode *uniform crossover*, terdapat mutasi dengan probabilitas 0,08, dan jumlah populasi 20. *Stopping criterion* pada proses optimasi ini adalah 1000 Generasi. Proses optimasi dilakukan sesuai dengan diagram alir pada Gambar 3.10.


Start
Ukuran Populasi
Mutation Rate
Jumlah Generasi
Bangkitkan Populasi Awal Secara Random
Gen 1 = 1 ≤ x ≤ 5
merepresentasikan Jumlah Hidden Layer
Gen 2 = 2 ≤ x ≤ 10
merepresentasikan Jumlah Node Tiap Hidden Layer
Gen 3 = 1 ≤ x ≤ 6
merepresentasikan activation function
1 -> hardlim
2-> hardlims
3-> purelin
4-> satlin
5-> logsig
6-> tansig
Proses training dan validation BPNN, dengan topologi/arsitektur sesuai dengan kromosom. Dengan data yang sudah ditentukan
Parameter Input : KP, KI dan KD
Parameter Output : ITAE
Evaluasi Nilai MSE (dianggap sebagai fitness value)
Pilih individu terbaik dengan roulette wheele
B
A



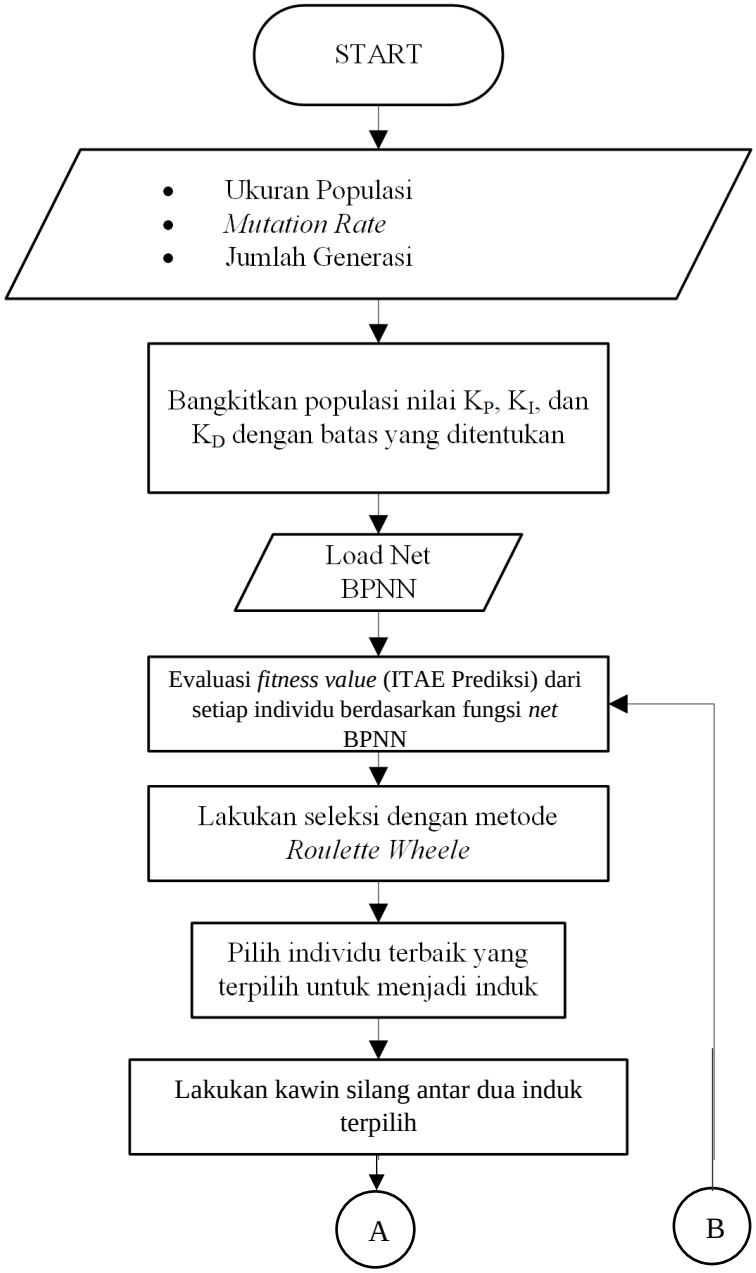

Gambar 3.10 Diagram Alir Optimasi Optimasi Arsitektur *Back Propagation Neural Network* Dengan Metode *Genetic Algorithm*

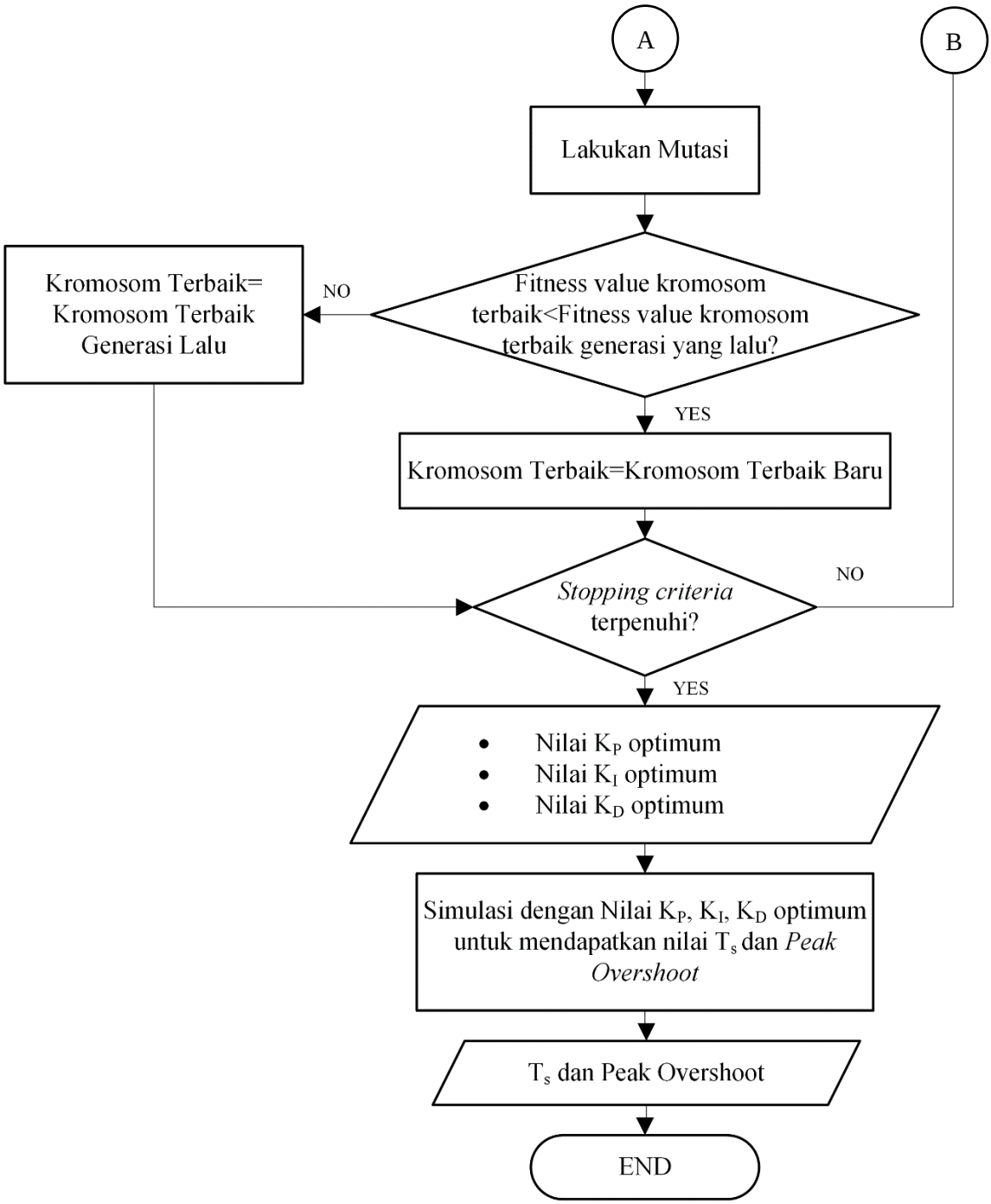
### 3.1.7. Proses Optimasi Nilai $K_P$, $K_I$, $K_D$ dengan BPNN-GA

Setelah mendapatkan arsitektur *neural network* terbaik, *net* tersebut akan digunakan dalam proses optimasi nilai $K_P$, $K_I$, dan $K_D$ akan dilakukan berdasarkan diagram alir pada Gambar 3.11. *Fitness value* pada proses optimasi ini adalah nilai ITAE yang diprediksi oleh BPNN. Setelah mendapat nilai yang optimal, nilai $K_P$, $K_I$, dan $K_D$ yang didapat akan disimulasikan untuk dilihat *step response*-nya dan dievaluasi nilai $T_S$ dan *Peak Overshootnya*.

Jumlah generasi yang digunakan untuk proses optimasi ini adalah 220 generasi dengan jumlah populasi, kemudian jumlah generasi dan populasi akan

dinaikkan masing-masing menjadi 1000 dan 6000, dan dilihat performanya. Pada optimasi ini range nilai $K_P$, $K_I$, dan $K_D$ dibatasi oleh persamaan 3.16-3.18.






Gambar 3.11 Diagram Alir Proses Optimasi Nilai $K_P$, $K_I$, dan $K_D$ dengan BPNN-GA

### 3.1.8. Membandingkan Respons Sistem Dengan Parameter Standar Kenyamanan

Setelah mendapatkan nilai $K_P$, $K_I$ dan $K_D$ terbaik, grafik respons sistem dengan parameter tersebut akan dibandingkan nilai *Root Mean Square* (RMS) akselerasinya dengan parameter standar kenyamanan yang dibahas pada subbab 2.9.

### 3.1.9. Komparasi Hasil

Hasil yang didapat dari hasil optimasi menggunakan metode BPNN-GA akan dibandingkan dengan hasil dari penelitian sebelumnya pada *paper* dengan

judul *"Stochastic Algorithm for PID Tuning Of Bus Suspension System"* oleh A.Karthikraja, dkk. (2009), yang ditunjukkan oleh Tabel 3.2.

Tabel 3.2 Hasil Penelitian Pada Paper (Arumugam et al., 2009)

| **Metode** | *Zeigler-Nichols (Paper)* | PSO *(Paper)* | *Real Coded GA (Paper)* | *Real Coded GA (Check)* |
|---|---|---|---|---|
| **Kp** | 832100 | 817790,085 | 809133,20 | 809133,20 |
| **Ki** | 624075 | 608269.494 | 621093.62 | 621093.62 |
| **Kd** | 208025 | 243616.576 | 243338.32 | 243338.32 |
| **Ts** | 1,19 | 1,17 | 1,17 | 1,17 |
| ***Peak Overshoot*** | -0,0095 | -0,0085 | -0,00825 | -0,00857 |

Pada Tabel 3.3 didapatkan performa terbaik yaitu pada metode *Real Coded Genetic Algortihm* dengan nilai $T_S$ 1,17s dan *Peak Overshoot* -0,00857 m. Diharapkan performa metode optimasi BPNN-GA memberikan hasil yang lebih baik.

### 3.1.10. Penarikan Kesimpulan dan Saran

Pada tahap ini akan membahas mengenai hasil dari penelitian serta pemberian saran agar dapat membantu penelitian yang akan dilakukan selanjutnya.

# BAB IV
# HASIL DAN PEMBAHASAN

## 4.1. Pemodelan BPNN

Dalam pemodelan BPNN, arsitektur BPNN ditentukan melalui proses optimasi menggunakan metode *Genetic Algorithm*, parameter yang akan di optimasi adalah parameter jumlah *hidden layer*, jumlah *nodes* tiap *hidden layer*, dan juga fungsi aktivasi yang digunakan di tiap *layer*. Data yang digunakan untuk proses *training* dan *validating* pada tahap ini telah melalui tahap normalisasi.

### 4.1.1. Optimisasi Arsitektur Neural Network Menggunakan Metode *Genetic Algorithm* (GA).

Optimasi arsitektur *neural network* dilakukan untuk mendapatkan performa *neural network* terbaik. Performa *neural networ*k dilihat dari nilai MSE *(Mean Squared Error)* yaitu rata-rata kesalahan kuadrat di antara nilai aktual dan nilai peramalan. Nilai MSE yang diharapkan adalah nilai MSE yang terkecil, sehingga menghasilkan *neural network* dengan performa yang terbaik.

Proses optimasi dengan metode GA akan diulangi sebanyak 10 (sepuluh) kali, dari sepuluh percobaan tersebut akan diambil nilai yang terbaik. Proses optimasi dilakukan dengan jumlah individu 300 dan jumlah generasi 1000. Nilai batas masing-masing parameter yang akan dicari dapat dilihat pada persamaan 3.16-3.18. Hasil training dapat dilihat pada Tabel 4.1 berikut menyajikan hasil yang didapatkan dari sepuluh kali percobaan optimasi.

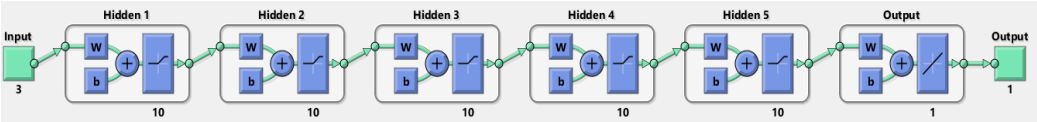
Tabel 4.1 Hasil Optimasi Parameter Arsitektur BPNN dengan Metode GA

| Percobaan ke- | Jumlah *Hidden Layer* | Jumlah *Nodes* | *Activation Function* | MSE Data Normalisasi | MSE Real |
|---|---|---|---|---|---|
| 1 | 5 | 5 | satlin | $1{,}8506 \times 10^{-6}$ | $1{,}0799 \times 10^{-14}$ |
| 2 | 4 | 10 | satlin | $1.1088 \times 10^{-7}$ | $1{,}4478 \times 10^{-13}$ |
| 3 | 5 | 10 | satlin | $1.6477 \times 10^{-8}$ | $5{,}7691 \times 10^{-15}$ |
| 4 | 5 | 10 | satlin | $1.9318 \times 10^{-8}$ | $8{,}0439 \times 10^{-15}$ |
| 5 | 3 | 10 | satlin | $6.2929 \times 10^{-7}$ | $7{,}6823 \times 10^{-14}$ |
| 6 | 4 | 10 | satlin | $9.6705 \times 10^{-8}$ | $7{,}4143 \times 10^{-15}$ |
| 7 | 5 | 5 | satlin | $7.2522 \times 10^{-6}$ | $2{,}6281 \times 10^{-14}$ |
| 8 | 5 | 8 | satlin | $9.3544 \times 10^{-8}$ | $7{,}6224 \times 10^{-15}$ |
| 9 | 4 | 6 | satlin | $1.8923 \times 10^{-7}$ | $1{,}0091 \times 10^{-13}$ |
| 10 | 5 | 9 | satlin | $9.8222e \times 10^{-7}$ | $3{,}5196 \times 10^{-14}$ |

Tabel 4.1 menunjukkan hasil optimasi arsitektur BPNN dengan metode *Genetic Algorithm*. Dari sepuluh kali percobaan yang dilakukan dapat diobservasi bahwa pada parameter *hidden layer* hasil optimasi paling sering mengeluarkan hasil 4 dan 5 sedangkan pada parameter jumlah *nodes*, jumlah *nodes* 10 menjadi yang paling banyak muncul, kemudian pada fungsi aktivasi, fungsi aktivasi *satlin* secara konsisten keluar sebagai yang paling optimal dari 10 kali percobaan. Hal ini menunjukkan bahwa proses optimasi menggunakan

GA berhasil dikarenakan hasilnya yang konsisten berada pada hasil tertentu. Hasil pada optimasi GA sangat dipengaruhi pada kromosom awal dari proses optimasi, oleh sebab itu hasil optimasi bisa berubah-ubah.

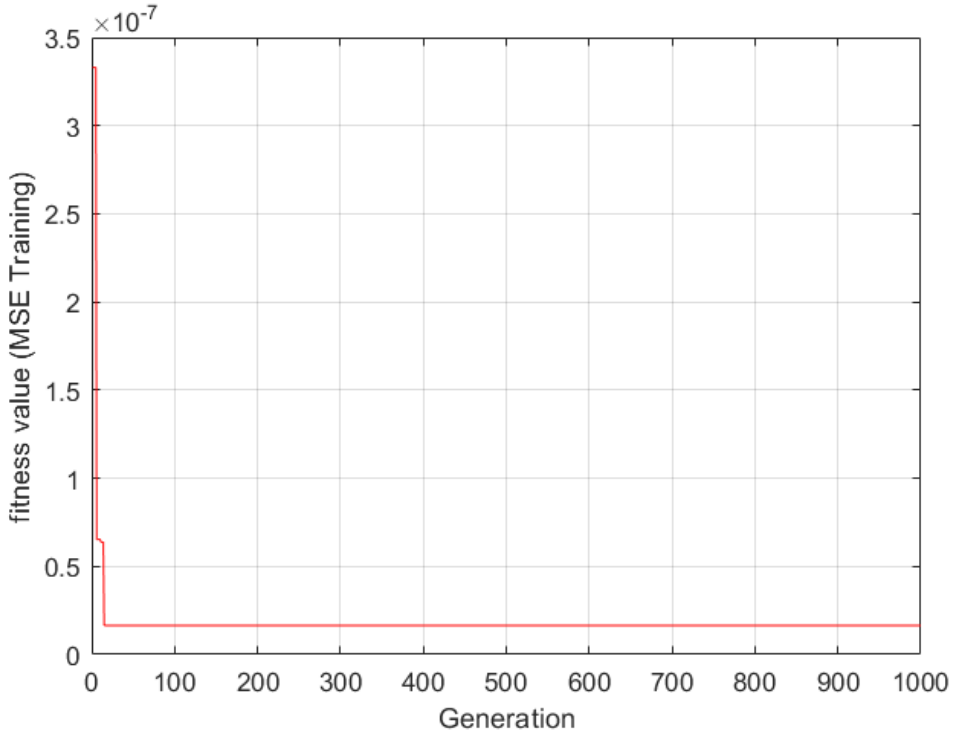
Pada tabel 4.1. didapatkan hasil terbaik pada arsitektur dengan 5 *Hidden Layer*, 10 *Nodes*, dan fungsi aktivasi *satlin* dengan nilai MSE $1{,}6477 \times 10^{-8}$. Sedangkan arsitektur terburuk yaitu menggunakan 5 *Hidden Layer*, 5 *Nodes*, dan fungsi aktivasi *satlin* dengan nilai MSE $7.2522 \times 10^{-6}$. Arsitektur Network BPNN terbaik dapat dilihat pada gambar 4.1.




Gambar 4.1 *Network* BPNN Terbaik

Apabila proses testing juga dimasukkan dalam proses optimasi, maka didapatkan hasil optimasi yang performanya hampir sama dengan performa net terbaik tanpa proses testing. Nilai MSE tanpa *testing* yaitu $8.1603 \times 10^{-7}$.

Arsitektur terbaik dengan 5 *Hidden Layer*, 10 *Nodes*, dan fungsi aktivasi *satlin* akan digunakan untuk mendapatkan net dari BPNN yang akan digunakan untuk tahap penelitian selanjutnya, yaitu optimasi nilai $K_P$, $K_I$ dan $K_D$ untuk mendapatkan respons sistem yang terbaik.




Gambar 4.2 Nilai MSE Dari Generasi 1 Sampai 1000 Proses Optimasi Arsitektur BPNN Percobaan Ketiga

Pada gambar 4.2 dapat diperhatikan penurunan *fitness value* (nilai MSE) dari proses optimasi. Seiring berjalannya generasi terjadi penurunan nilai MSE, namun pada generasi ke-15 *fitness value* konstan sampai generasi ke 1000 pada nilai MSE $1{,}6477 \times 10^{-8}$.

Tabel 4.2 Perbandingan Waktu yang Dibutuhkan Tiap Metode Untuk Optimasi Parameter BPNN

| No | Metode Optimasi Arsitektur BPNN | Waktu yang dibutuhkan |
|---|---|---|
| 1 | GA | 3,3 menit |
| 2 | Manual | 300 menit |

Waktu yang dibutuhkan metode GA untuk mendapatkan nilai MSE terkecil akan dibandingkan dengan waktu yang dibutuhkan pada metode metode manual yaitu dengan cara mencoba seluruh kombinasi *neural network* yang ada.

Berdasarkan hasil percobaan proses optimasi parameter BPNN dengan GA memakan waktu sekitar 147,2 sekon untuk satu generasi. Berdasarkan hasil percobaan hasil yang didapatkan sudah konvergen pada generasi ke dua sehingga dibutuhkan waktu total 198 sekon untuk menyelesaikan proses optimasi ini.

Dibandingkan dengan melakukan percobaan secara manual untuk setiap kombinasi yang ada, dimana berdasarkan pengalaman penulis membutuhkan waktu sekitar 60 sekon untuk proses *input* parameter, *running*, dan pencatatan hasil MSE yang didapatkan untuk satu kombinasi, sehingga dibutuhkan waktu sekitar 300 menit atau 5 jam untuk mendapatkan arsitektur terbaik dengan mencoba seluruh kombinasi yang ada untuk mendapatkan arsitektur BPNN terbaik. Tabel perbandingan waktu yang dibutuhkan antara metode manual dengan mencoba semua kombinasi yang ada dapat dilihat pada tabel 4.2.

Metode optimasi parameter arsitektur *neural network* dengan GA tentu saja dapat menghemat waktu dan tenaga dari pengguna saat proses penentuan parameter arsitektur terbaik yang akan digunakan saat menggunakan *neural network.* Hal ini dikarenakan metode ini hanya membutuhkan sekitar sepersepuluh waktu yang dibutuhkan dibandingkan dengan metode manual. Metode ini juga hanya memerlukan satu kali *running* dan langsung mendapatkan arsitektur *neural network* yang terbaik, dibandingkan dengan metode manual yang membutuhkan proses *input, running,* dan pencatatan nilai MSE, dan sortasi manual, yang tentu saja membutuhkan usaha yang lebih.

### 4.1.2. Uji Arsitektur BPNN Terbaik

Arsitektur BPNN terbaik yang didapatkan sebelumnya akan diuji untuk memprediksi nilai ITAE dari nilai $K_P$, $K_I$, $K_D$ acak dengan batasan yang telah ditentukan. Proses ini akan dilakukan pada awal proses optimasi dengan GA, dimana kromosom awal akan dievaluasi nilai *fitness value*-nya. Nilai prediksi akan dibandingkan dengan nilai ITAE *real* dan dicari nilai rata-rata persentase *error*-nya. Hasil prediksi dan nilai riil dari 1000 kromosom, yaitu kombinasi nilai $K_P$, $K_I$, dan $K_D$ acak, serta persentase *error* dapat dilihat pada tabel 4.3 berikut

Tabel 4.3 Tabel Perbandingan Nilai Riil dan Prediksi BPNN Terbaik

| No | $K_P$ | $K_I$ | $K_D$ | Nilai Riil | Prediksi | Persentase *Error* |
|---|---|---|---|---|---|---|
| 1 | 832341 | 639402 | 231757 | 0.0012198 | 0.001227 | 0.59% |
| 2 | 836556 | 620143 | 245685 | 0.0011806 | 0.001187 | 0.54% |
| 3 | 843169 | 606515 | 217915 | 0.0012625 | 0.001269 | 0.54% |
| 4 | 801132 | 635431 | 221405 | 0.0012606 | 0.001268 | 0.60% |
| 5 | 807560 | 633077 | 239825 | 0.0012046 | 0.001214 | 0.80% |
| 6 | 826260 | 613938 | 214831 | 0.001277 | 0.001284 | 0.54% |
| 7 | 811328 | 624676 | 229847 | 0.0012323 | 0.00124 | 0.67% |
| 8 | 842084 | 637094 | 241694 | 0.001189 | 0.001197 | 0.66% |
| 9 | 821502 | 623459 | 204364 | 0.0013158 | 0.001321 | 0.40% |
| 10 | 814969 | 644364 | 229370 | 0.0012315 | 0.001241 | 0.73% |
| 11 | 828859 | 617965 | 239585 | 0.0011995 | 0.001208 | 0.74% |
| 12 | 817237 | 637956 | 202678 | 0.0013224 | 0.001327 | 0.33% |
| 13 | 842851 | 603700 | 215182 | 0.0012722 | 0.001278 | 0.49% |
| 14 | 814328 | 600539 | 202746 | 0.0013251 | 0.00133 | 0.37% |
| 15 | 822048 | 629276 | 213413 | 0.0012821 | 0.001288 | 0.47% |
| 16 | 838320 | 642851 | 215158 | 0.0012713 | 0.001275 | 0.30% |
| 17 | 819151 | 628986 | 223931 | 0.0012479 | 0.001255 | 0.58% |
| 18 | 811940 | 647663 | 236547 | 0.0012115 | 0.001221 | 0.81% |
| 19 | 819696 | 616908 | 219959 | 0.0012612 | 0.001268 | 0.57% |
| … | … | … | … | … | … | … |
| 1000 | 807603 | 622614 | 232651 | 0.0012253 | 0.001236 | 0.85% |
| | | | | | **Rata-Rata** | **0.54%** |

Dari Tabel di atas didapatkan nilai rata-rata persentase *error* sebesar 0,54% maka dapat disimpulkan bahwa *net* yang digunakan dapat memberikan prediksi nilai ITAE dengan baik.

## 4.2. Optimasi Nilai $K_P$, $K_I$, dan $K_D$ Dengan Metode BPNN-GA

Setelah mendapatkan arsitektur BPNN terbaik, *net* BPNN tersebut akan digunakan untuk memprediksi nilai ITAE yang digunakan sebagai fungsi objektif.

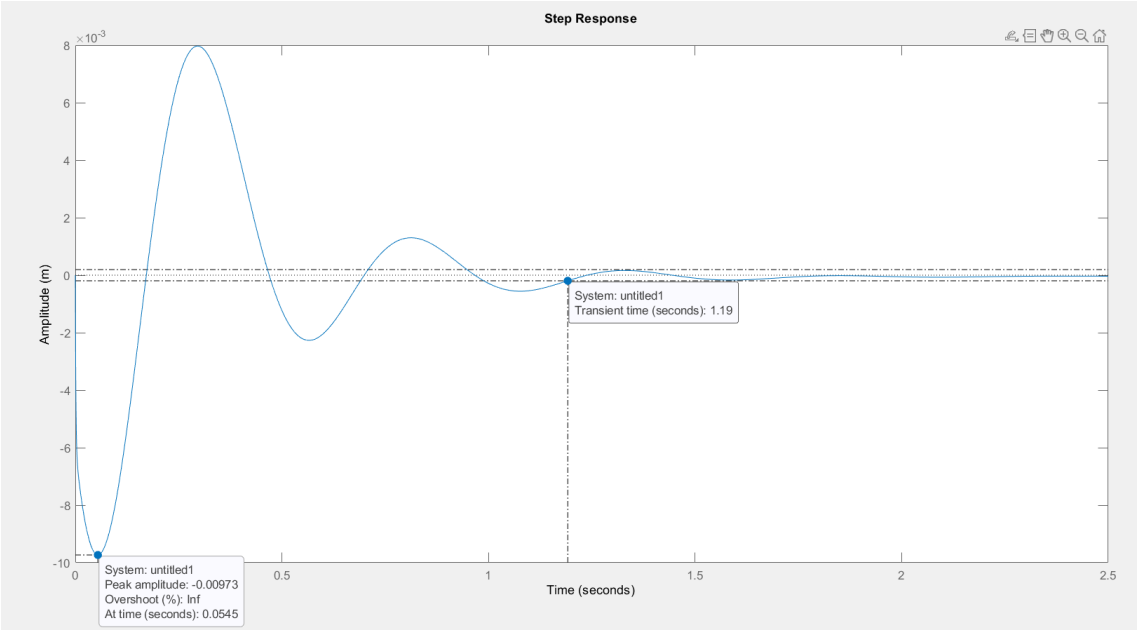
Untuk mendapatkan nilai *settling time* ($T_S$) yang paling kecil dan *peak overshoot* mutlak yang paling kecil, diperlukan pengaturan nilai variabel $K_P$, $K_I$ dan $K_D$ yang optimal, parameter-parameter tersebutlah yang akan dioptimasi menggunakan metode BPNN-GA.

Pada proses optimasi ini digunakan beberapa pengaturan parameter GA yang berbeda yaitu memvariasikan jumlah populasi dan generasi. Penggunaan populasi 220 dan generasi 220 disajikan dalam tabel 4.4. Dari Tabel 4.4. dapat dilihat bahwa hasil dari optimasi $K_P$, $K_I$, dan $K_D$ masih bervariasi. Nilai $T_S$ konsisten berada di 1,19 namun nilai *peak overshoot* masih bervariasi.

Tabel 4.4 Hasil Optimasi BPNN-GA Dengan Populasi 220 Generasi 220

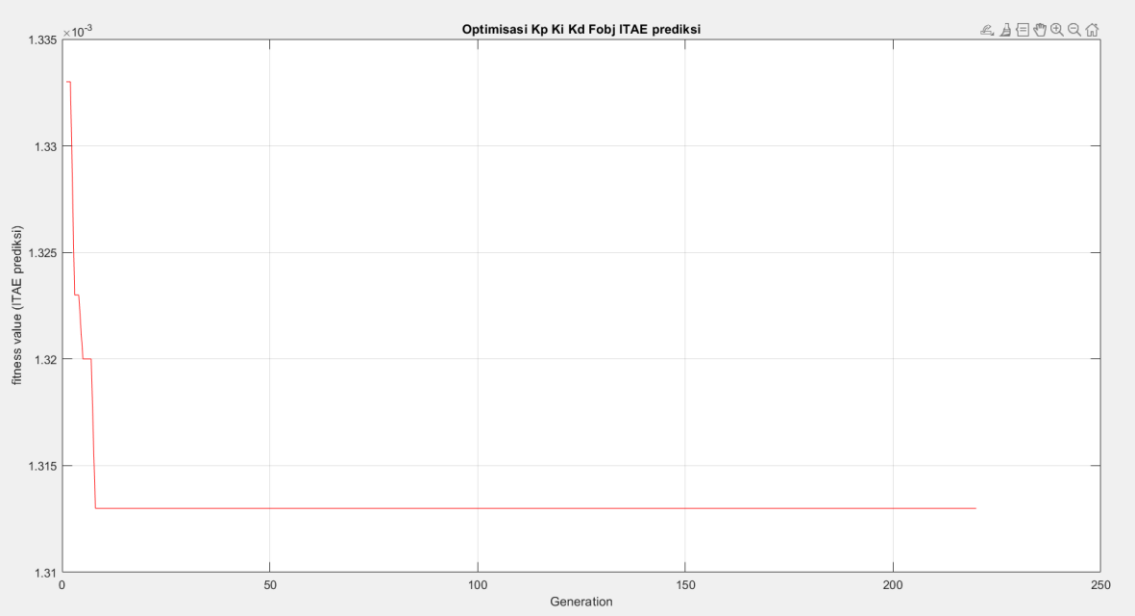
| Percobaan ke- | $K_P$ | $K_I$ | $K_D$ | $T_S$ (s) | Peak Overshoot (m) | ITAE |
|---|---|---|---|---|---|---|
| 1 | 843853 | 647546 | 200548 | 1.19 | -0,00987 | 0.001326 |
| 2 | 847879 | 649291 | 204070 | 1.19 | -0,00973 | 0.001313 |
| 3 | 848530 | 649204 | 200057 | 1,19 | -0.00987 | 0.001327 |
| 4 | 842942 | 642983 | 202455 | 1,19 | -0,00980 | 0.00132 |
| 5 | 846183 | 649424 | 203581 | 1,19 | -0.00975 | 0.001315 |
| 6 | 849860 | 648857 | 201696 | 1,19 | -0,00981 | 0.001321 |
| 7 | 847705 | 647725 | 202624 | 1,19 | -000978 | 0.001318 |
| 8 | 847154 | 649719 | 200348 | 1.19 | -0,00987 | 0.001326 |
| 9 | 845314 | 649011 | 201393 | 1,19 | -0,00983 | 0.001323 |
| 10 | 847807 | 647878 | 202609 | 1,19 | -000978 | 0.001318 |

Hal ini dipengaruhi oleh pembentukan dari kromosom awal yang dilakukan secara acak, apabila dalam kromosom tersebut tidak terdapat kromosom terbaik dari percobaan sebelumnya, maka kromosom terbaik akan berubah sesuai dengan *random generation* kromosom percobaan sekarang.




Gambar 4.3 Grafik Respons Sistem Dengan Parameter $K_P$, $K_I$ dan $K_D$ Terbaik (220 kromosom dan 220 generasi)

Dapat dilihat bahwa nilai ITAE terbaik dari sepuluh kali percobaan didapatkan 0.001313 dengan nilai $T_S$ 1,19 dan *Peak Overshoot* -0,00973 pada nilai $K_P$, $K_I$, dan $K_D$ 847879, 649291, dan 204070. Grafik respons dari sistem dapat dilihat pada gambar 4.3.




Gambar 4.4 Nilai ITAE Dari Generasi 1 Sampai 220 Proses Optimasi Nilai $K_P$, $K_I$ dan $K_D$ Percobaan Ke-2 Dengan Jumlah Populasi 220

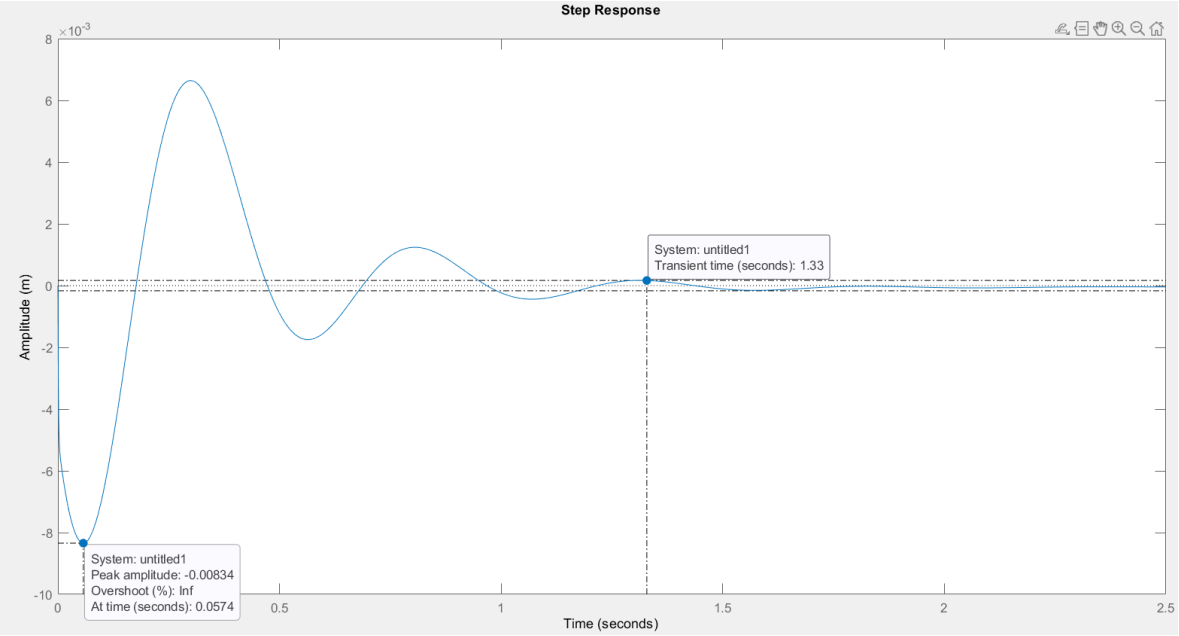
Dari proses optimasi dapat dilihat bahwa terjadi penurunan *fitness value*, namun pada generasi ke-8 *fitness value* konstan 0.001313 hingga generasi akhir. Nilai fitness value dari tiap generasi dapat dilihat pada gambar 4.4. Hasil yang didapatkan masih kurang optimal dikarenakan jumlah kromosom yang cenderung sedikit. Kromosom awal yang di-*generate* secara acak tidak mengeluarkan nilai yang optimal, oleh sebab itu jumlah kromosom harus ditingkatkan pada percobaan selanjutnya.

Peningkatan jumlah kromosom dan generasi dilakukan dengan harapan akan mendapatkan nilai $K_P$, $K_I$, dan $K_D$ yang lebih baik. Sehingga mendapatkan performa nilai $T_s$ dan *Peak Overshoot* yang lebih baik. Pada percobaan berikutnya jumlah kromosom akan ditingkatkan menjadi 6000 dengan jumlah generasi 1000. Hasil optimasi dapat dilihat pada Tabel 4.5 berikut.

Tabel 4.5 Hasil Optimasi BPNN-GA Dengan Populasi 6000 Generasi 1000

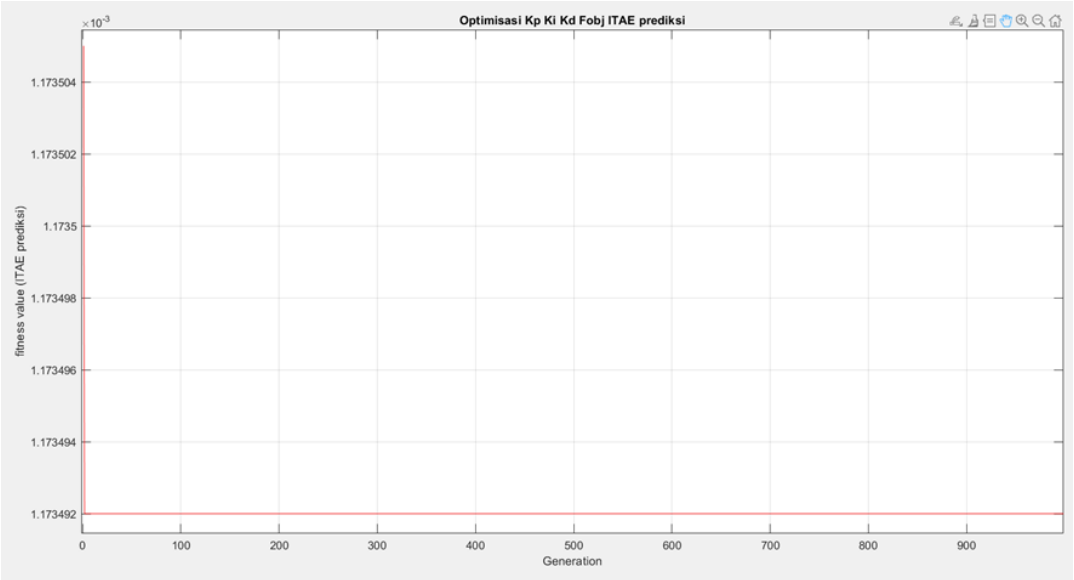
| **Percobaan ke-** | **$K_P$** | **$K_I$** | **$K_D$** | **$T_S$ (s)** | ***Peak Overshoot* (m)** | **ITAE** |
|---|---|---|---|---|---|---|
| 1 | 849439 | 645982 | 249579 | 1,33 | -0,00834 | 0.001173564 |
| 2 | 848271 | 642347 | 249425 | 1,33 | -0,00834 | 0.00117381 |
| 3 | 848626 | 644990 | 249857 | 1,33 | -0,00834 | 0.001173513 |
| 4 | 847480 | 643198 | 249817 | 1,33 | -0,00834 | 0.001173736 |
| 5 | 849439 | 645982 | 249579 | 1,33 | -0,00834 | 0.001173564 |
| 6 | 849778 | 642164 | 249214 | 1,33 | -0,00834 | 0.001173709 |
| 7 | 848352 | 645151 | 249951 | 1,33 | -0,00834 | 0.001173492 |
| 8 | 849778 | 642164 | 249214 | 1,33 | -0,00835 | 0.001173709 |
| 9 | 848480 | 629716 | 249138 | 1,33 | -0,00836 | 0.001174021 |
| 10 | 848613 | 648604 | 249963 | 1,33 | -0,00834 | 0.001173449 |

Dari Tabel 4.5. dapat dilihat bahwa hasil dari optimasi $K_P$, $K_I$, dan $K_D$ dengan sepuluh kali percobaan. Nilai dari $K_P$, $K_I$ dan $K_D$ konsisten di *range* nilai tertentu. Walaupun kromosom awal dibangkitkan secara acak, namun dikarenakan dari tingginya nilai individu, maka kromosom terbaik dari percobaan selanjutnya mendekati percobaan sebelumnya. Pada percobaan ini nilai $T_S$ konsisten di nilai 1,33 s. Nilai *peak overshoot* beragam di antara -0,00834 dan -0,00836 m.




Gambar 4.5 Grafik Respons Sistem Dengan Parameter $K_P$, $K_I$ dan $K_D$ Terbaik (6000 kromosom dan 1000 generasi)

Pada percobaan dengan variasi ini didapatkan nilai yang lebih baik dari percobaan sebelumnya. Hasil optimasi terbaik didapatkan pada percobaan ke-7, dengan nilai ITAE 0,001173492, *Peak Overshoot* pada -0,00834 dan $T_S$1,33 s. Grafik dan karakteristik respons dari sistem dengan Parameter $K_P$, $K_I$ dan $K_D$ terbaik dapat dilihat pada gambar 4.5.




Gambar 4.6 Nilai ITAE Dari Generasi 1 Sampai 1000 Proses Optimasi Nilai $K_P$, $K_I$ dan $K_D$ Percobaan Ke-2 Dengan Jumlah Populasi 6000

Dari proses optimasi dapat dilihat bahwa terjadi penurunan *fitness value*, namun pada generasi ke-1 *fitness value* konstan 0.001173492 hingga generasi akhir, hal ini dapat diperhatikan pada gambar 4.6. Dari ketiga variasi parameter hasil tersebut didapatkan bahwa parameter GA terbaik adalah jumlah populasi 6000 dan jumlah generasi 1000, dengan nilai $T_S$ 1,17 s dan *Peak Overshoot* -0,00834 m. Dari percobaan ini dapat diobservasi bahwa apabila jumlah kromosom dinaikkan maka hasil optimasi akan menjadi lebih baik.

## 4.3. Modifikasi Nilai ITAE Untuk Memberikan Prioritas Pada Salah Satu Parameter

Hasil optimasi yang didapatkan masih dapat dilakukan perbaikan disalah satu parameter performa, dengan memberikan modifikasi pada rumus ITAE. Hasil yang didapatkan dari proses optimasi menggunakan parameter ITAE, menghasilkan nilai optimasi *peak overshoot* yang jauh lebih baik dibanding *paper* pembanding, namun nilai *settling time* yang lebih buruk. Apabila kita ingin melakukan perbaikan dari parameter *settling time*, maka dapat diberikan modifikasi pada variabel t pada parameter ITAE.

$$ITAE = \int_0^\infty t^8 |e(t)| \ \ dt \qquad (4.1)$$

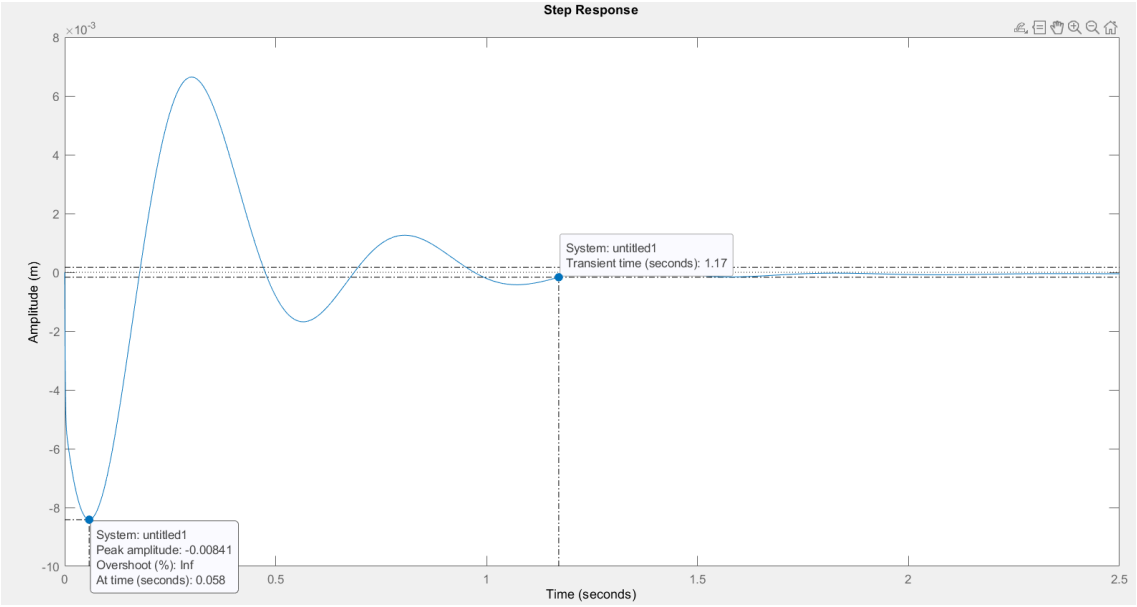
Modifikasi dari nilai ITAE dapat dilihat pada persamaan 4.1, dimana nilai t dipangkatkan dengan 8. Pertimbangan pemberian pangkat 8 dikarenakan *range* optimasi mendapatkan nilai $T_S$ yang hanya berkisar 1 sampai 1,5 s, sehingga dengan memberikan pangkat 8 dapat memberikan pengaruh yang signifikan. Untuk melakukan optimasi dengan menggunakan parameter ITAE yang telah diberikan modifikasi, maka dilakukan kembali tahapan-tahapan dari penelitian ini, yaitu *data generation* dengan ITAE yang diberikan modifikasi serta mencari arsitektur BPNN yang terbaik untuk *dataset* tersebut. Penulis telah melakukan *data generation* dengan ITAE yang telah diberi modifikasi sesuai dengan DOE pada tabel 3.1. Optimasi arsitektur BPNN dengan metode GA juga sudah berhasil didapatkan dengan 1 *Hidden Layer*, 10 *Hidden Nodes*, dan fungsi aktivasi *hardlim*.

Dengan parameter BPNN tersebut telah dilakukan dua kali percobaan optimasi GA menggunakan ITAE yang diberikan modifikasi. Hasil optimasi dapat dilihat pada tabel 4.6 berikut.

Tabel 4.6 Hasil Optimasi BPNN-GA Dengan Nilai ITAE Dengan Modifikasi

| **Percobaan ke-** | **$K_P$** | **$K_I$** | **$K_D$** | **$T_S$** | ***Peak Overshoot*** | **ITAE** |
|---|---|---|---|---|---|---|
| 1 | 800009 | 649696 | 249555 | 1,17 | -0,00842 | 14.4810 |
| 2 | 801838 | 649984 | 249681 | 1,17 | -0,00842 | 14.3651 |

Dari dua kali percobaan didapatkan nilai $T_S$ dengan nilai 1,17 s dan *peak overshoot* -0,00842 m, dengan nilai $K_P$, $K_I$, dan $K_D$ 801838, 649984, dan 249681. Hal ini menunjukkan nilai $T_S$ yang lebih baik namun nilai mutlak *peak overshoot* yang lebih tinggi dari optimasi dengan nilai ITAE tanpa modifikasi.




Gambar 4.7 Grafik Respons Sistem Dengan Parameter $K_P$, $K_I$ dan $K_D$ Terbaik (6000 kromosom dan 1000 generasi) Dengan Parameter ITAE yang Diberikan Modifikasi

Pada Gambar 4.7 dapat dilihat grafik dan karakteristik respons dari sistem dengan parameter $K_P$, $K_I$ dan $K_D$ terbaik dengan parameter optimasi ITAE yang diberikan modifikasi.

## 4.4. Pemilihan dan Perbandingan Hasil Optimasi Terbaik

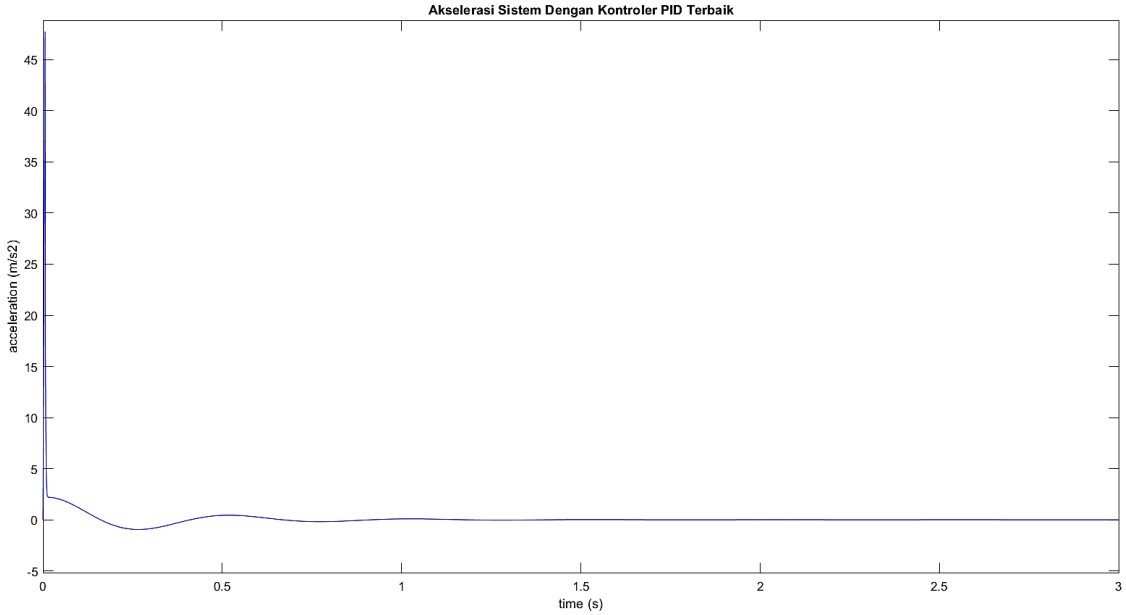
Jika dilakukan pertimbangan pemilihan antara nilai *peak overshoot* yang lebih baik atau *settling time* yang lebih baik, berdasarkan hasil-hasil optimasi yang didapatkan dari penelitian, penulis mempertimbangkan dengan analisa kondisi nyata pada jalan. Penulis memilih hasil optimasi dengan nilai ITAE tanpa modifikasi, yaitu yang memiliki nilai *peak overshoot* yang lebih baik.

Hal ini dikarenakan *peak overshoot* dapat memberikan kenyamanan yang lebih, sesaat setelah terjadi gangguan atau *disturbance*, yaitu meredam amplitudo getaran. Disisi lain nilai *settling time* menjadi tidak terlalu signifikan dikarenakan dari kontur jalan yang tidak selalu rata, sehingga *disturbance* akan selalu ada. Oleh sebab itu kemampuan dari pengendali untuk meredam nilai *peak overshoot* menjadi lebih penting.

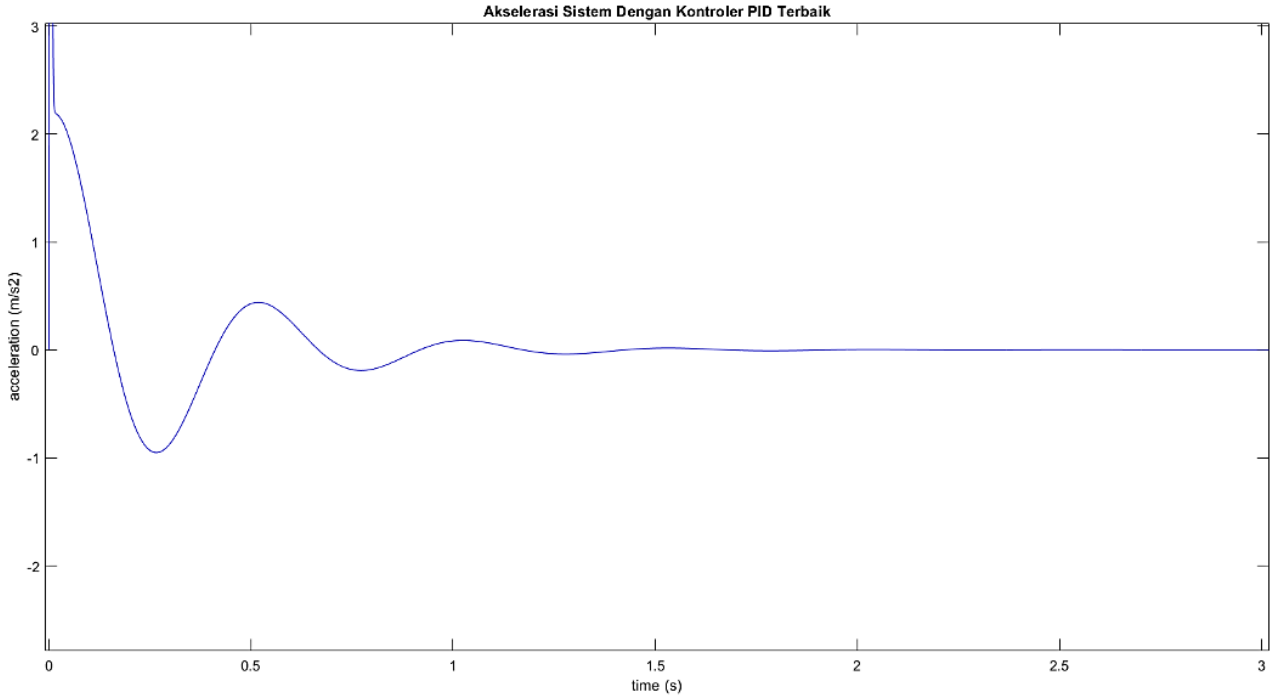
## 4.5. Analisa Kenyamanan Kendaraan Dengan Parameter Pengendali Terbaik

Dengan Nilai $K_P$, $K_I$ dan $K_D$ terbaik yang didapatkan, setelah dilakukan simulasi, didapatkan grafik respons akselerasi dari sistem, dari grafik akan dicari nilai RMS dari nilai akselerasi. Kemudian hasil ini akan dibandingkan dengan parameter kenyamanan yang sudah dijelaskan pada subbab 2.9. Hasil ini akan dibandingkan dengan sistem tanpa pengendali (pasif), sehingga dapat dilihat perbaikan nilai RMS antara sistem tanpa pengendali dan dengan pengendali.

Grafik respons akselerasi dari sistem dengan parameter pengendali terbaik dapat dilihat pada gambar 4.8 dan 4.9.

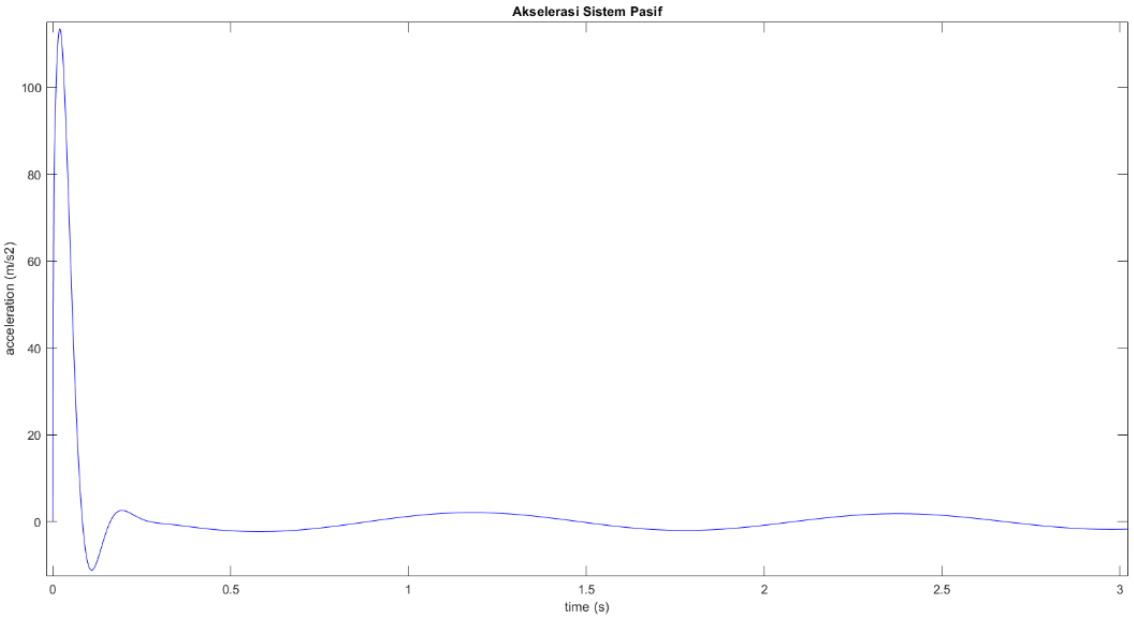



Gambar 4.8 Respons Akselerasi Sistem Dengan Pengendali PID Terbaik

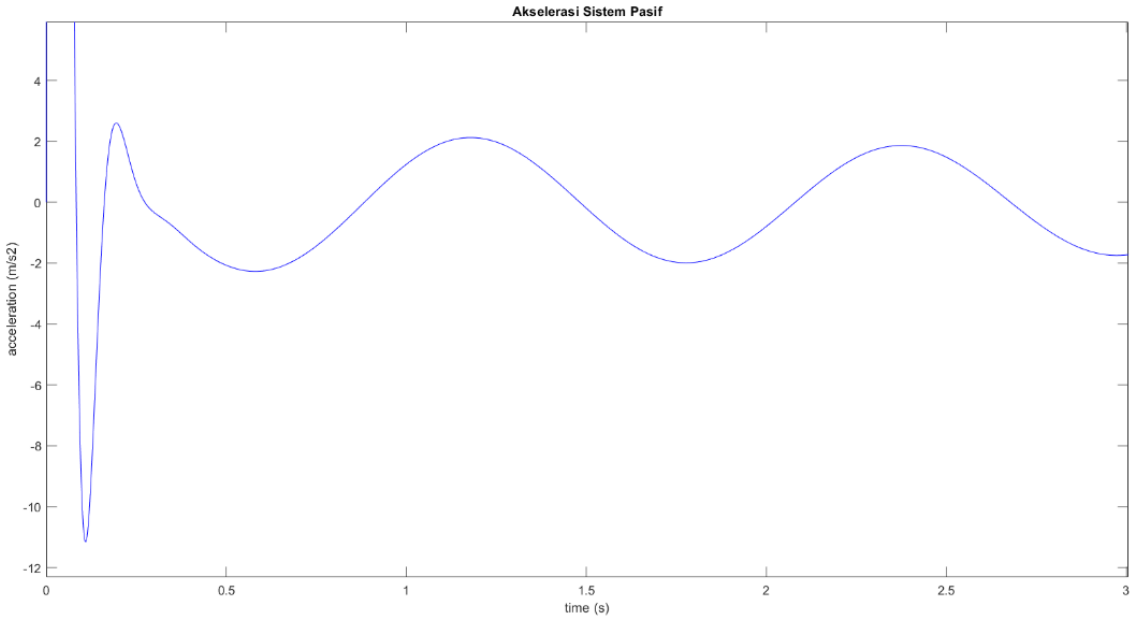



Gambar 4.9 Respons Akselerasi Sistem Dengan Pengendali PID Terbaik (Diperbesar)

Dari perhitungan nilai RMS akselerasi selama 5 detik didapatkan nilai RMS akselerasi sebesar 0,5747 $m/s^2$ . Menurut parameter standar kenyamanan pada subbab 2.9 maka sistem ini termasuk dalam kriteria sedikit tidak nyaman. Sebagai pembanding pada gambar 4.10 dan 4.11 ditampilkan grafik respons akselerasi dari sistem tanpa pengendali (pasif).




Gambar 4.10 Respons Akselerasi Sistem Tanpa Pengendali




Gambar 4.11 Respons Akselerasi Sistem Tanpa Pengendali (Diperbesar)

Dari perhitungan nilai RMS akselerasi, didapatkan nilai RMS akselerasi sebesar 7,0813 $m/s^2$ nilai ini berdasarkan parameter kenyamanan standar, termasuk dalam kriteria sangat tidak nyaman. Sehingga dapat dilihat perbaikan tingkat kenyamanan menurut standar ISO 2631 dari sistem tanpa pengendali dan dengan pengendali PID yang telah di optimasi parameter $K_P$, $K_I$ dan $K_D$ nya.

Untuk menentukan apakah kendaraan sudah memenuhi standar ambang batas kenyamanan, menurut paper *"Comparison between ISO 2631–1 Comfort Prediction Equations and Self-Reported Comfort Values during Occupational Exposure to Whole-Body Vehicular Vibration"*(Plewa et al., 2012), sangat sulit untuk menentukan tingkat

kenyamanan hanya berdasarkan perhitungan atau persamaan matematis, hal ini dikarenakan tingkat kenyamanan bersifat subjektif.

Respons setiap individu terhadap getaran sangat bervariasi. Parameter kenyamanan bisa saja menjadi bias apabila diberikan kepada individu yang bekerja atau sudah terbiasa dengan lingkungan kerja yang memiliki tingkat getaran yang tinggi, contohnya pada operator alat-alat berat, persepsi tingkat kenyamanan mereka terhadap getaran akan lebih tidak sensitif.

Sebaliknya individu yang terbiasa berada dilingkungan yang minim getaran atau individu yang memiliki penyakit atau kondisi tertentu sehingga memiliki tingkat sensitivitas yang lebih tinggi terhadap getaran seharusnya memiliki persepsi ketidaknyamanan terhadap getaran yang lebih sensitif. Oleh sebab itu perlu dilakukan penelitian lebih lanjut untuk menentukan batas getaran yang diizinkan agar kendaraan masuk dalam kategori nyaman.

## 4.6. Perbandingan Hasil Optimasi Dengan *Paper* Penelitian Sebelumnya

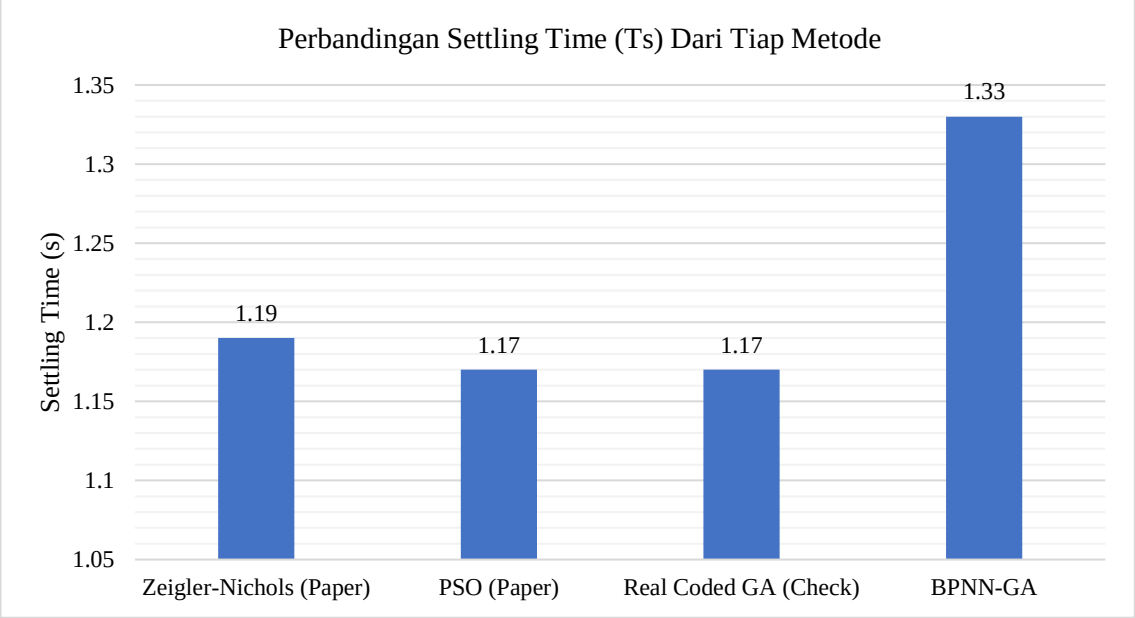
Berdasarkan hasil yang didapatkan dari penelitian, hasil optimasi terbaik dari penelitian ini dibandingkan dengan hasil optimasi pada *paper*, hasil disajikan pada Tabel 4.7 berikut

Tabel 4.7 Perbandingan Performa Metode Optimasi Dengan *Paper* Penelitian Sebelumnya

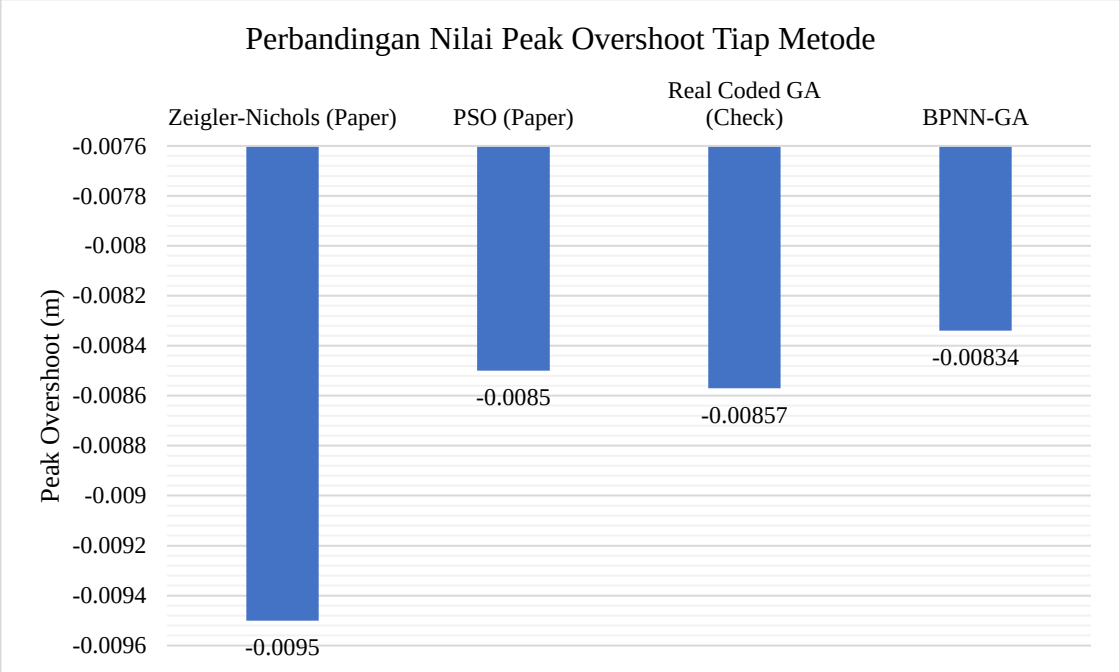
| **Metode** | *Zeigler-Nichols (Paper)* | PSO *(Paper)* | *Real Coded GA (Paper)* | *Real Coded GA (Check)* | BPNN-GA |
|---|---|---|---|---|---|
| **$K_P$** | 832100 | 817790,085 | 809133,20 | 809133,20 | 809193 |
| **$K_I$** | 624075 | 608269.494 | 621093.62 | 621093.62 | 621978 |
| **$K_D$** | 208025 | 243616.576 | 243338.32 | 243338.32 | 243984 |
| **$T_S$** | 1,19 | 1,17 | 1,17 | **1,17** | **1,33** |
| ***Peak Overshoot*** | -0,0095 | -0,0085 | -0,00825 | **-0,00857** | **-0,00834** |

Berdasarkan tabel 4.7 dapat dilihat perbandingan antara hasil dari penelitian yang berjudul *"Stochastic Algorithm for PID Tuning Of Bus Suspension System"* oleh A.Karthikraja, dkk. (2009) dengan penelitian yang dilakukan sekarang. Dapat dilihat bahwa nilai *settling time* ($T_S$) yang didapat dari BPNN-GA memiliki performa yang hampir sama dengan metode optimasi *Real Coded GA*, namun parameter ini jauh lebih baik dari metode *Zeigler-Nichols*. Namun untuk *Peak Overshoot* performa dari metode optimasi BPNN-GA lebih baik dari PSO dan *Real Coded GA*.

Berdasarkan analisa kondisi riil yang dijelaskan pada subbab 4.4, dapat disimpulkan bahwa optimasi dengan metode BPNN-GA lebih baik dari pada metode optimasi yang dilakukan pada *paper* yang berjudul *"Stochastic Algorithm for PID Tuning Of Bus Suspension System"* oleh A.Karthikraja, dkk. (2009), karena nilai *peak overshoot*-nya yang lebih baik.




Gambar 4.12 Grafik Perbandingan Nilai *Settling Time* tiap metode




Gambar 4.13 Grafik Perbandingan Nilai *Peak Overshoot* tiap metode

Pada Gambar 4.12 dan 4.13 ditampilkan grafik visualisasi perbandingan performa parameter *settling time* dan *peak overshoot* antara pengendali yang di optimasi menggunakan metode BPNN-GA dengan metode-metode optimasi yang digunakan pada *paper*. Pada grafik dapat dilihat nilai *settling time* terbaik pada metode PSO dan *Real Coded GA*, sedangkan metode BPNN-GA memiliki nilai paling tinggi yaitu 1,33 s. Sedangkan pada grafik perbandingan nilai *settling time* nilai yang ditunjukkan menunjukkan nilai negatif, dikarenakan pergerakan dari sistem menurut grafik *step response* bergerak kearah sumbu y negatif. Dari grafik nilai *peak overshoot* terbaik terdapat pada metode BPNN-GA.

(*Halaman ini sengaja dikosongkan*)

# BAB V
# KESIMPULAN DAN SARAN

## 5.1. Kesimpulan

Berdasarkan hasil yang telah didapatkan dari penelitian kali ini dengan judul "Optimasi Parameter Pengendali PID Pada Sistem Suspensi Aktif Dengan Metode *Back Propagation Neural Network-Genetic Algorithm*", dapat diambil kesimpulan sebagai berikut :

1. Pemodelan model seperempat kendaraan dapat dilakukan di *software* MATLAB R2021a.
2. Proses optimasi arsitektur BPNN dengan metode GA menggunakan waktu yang lebih sedikit, untuk mendapatkan hasil yang paling optimal, dibandingkan menggunakan metode manual. Pemodelan BPNN paling optimal yang didapatkan untuk melakukan prediksi nilai ITAE dari parameter *input* nilai $K_P$, $K_I$, dan $K_D$, adalah menggunakan model BPNN dengan 5 (lima) *hidden layer*, 10 (sepuluh) *neuron* di setiap *hidden layer*-nya, dan fungsi aktivasi *satlin* dengan nilai MSE *training* $1.6477 \times 10^{-8}$.
3. Nilai $K_P$, $K_I$, dan $K_D$ optimum yang didapatkan dengan metode BPNN-GA yaitu masing-masing 848352, 645151, dan 249951. Dari nilai $K_P$, $K_I$, dan $K_D$ tersebut didapatkan nilai *Settling Time* ($T_S$) 1,33 s dan *Peak Overshoot* -0,00834 m. Dapat disimpulkan bahwa, hasil yang didapatkan, performa *peak overshootnya* lebih baik, sedangkan performa *settling time*-nya mendekati dengan hasil optimasi dari paper *"Stochastic Algorithm for PID Tuning Of Bus Suspension System"* oleh A.Karthikraja, dkk. (2009). Berdasarkan analisa kondisi riil optimasi dengan metode BPNN-GA lebih baik dari pada metode optimasi yang digunakan pada *paper.*
4. Tingkat kenyamanan dari kendaraan yang menggunakan sistem suspensi aktif dengan pengendali PID, yang menggunakan parameter $K_P$, $K_I$ dan $K_D$ terbaik, masuk dalam kategori sedikit tidak nyaman menurut standar ISO 2631, dengan nilai RMS akselerasi 0,5747 m/s$^2$.

## 5.2. Saran

Adapun saran yang dapat diberikan setelah melakukan penelitian ini adalah sebagai berikut :

1. Metode optimasi dalam penelitian ini menggunakan *Back Propagation Neural Network–Genetic Algorithm (BPNN-GA).* Pada penelitian selanjutnya , dapat dilakukan studi untuk melakukan optimasi dengan menggunakan metode-metode optimasi yang lain sebagai pembanding.
2. Untuk penelitian selanjutnya mengenai optimasi menggunakan BPNN sebaiknya menggunakan jumlah data yang cukup besar, agar prediksi yang dihasilkan oleh jaringan BPNN lebih akurat.
3. Untuk penelitian selanjutnya dalam tahap optimasi GA, dapat digunakan nilai parameter optimasi yang berbeda, karena parameter yang cocok untuk metode GA bisa berbeda-beda untuk setiap permasalahan
4. Untuk penelitian menggunakan metode GA digunakan jumlah individu atau populasi dan generasi yang lebih banyak.

5. Untuk penelitian selanjutnya dapat dibuat program optimasi yang universal untuk semua sistem.

# DAFTAR PUSTAKA


Arumugam, K., Govindan, Dr. P., & Subramanian, R. (2009, June 4). *Stochastic algorithm for PID tuning of bus suspension system*.

Azizan, A., Fard, M., Azari, M. F., & Jazar, R. (2017). Effects of vibration on occupant driving performance under simulated driving conditions. *Applied Ergonomics, 60*, 348–355. https://doi.org/10.1016/j.apergo.2016.12.020

Chiha, I., Liouane, N., & Borne, P. (2012). Tuning PID Controller Using Multiobjective Ant Colony Optimization. *Applied Computational Intelligence and Soft Computing, 2012*. https://doi.org/10.1155/2012/536326

Fausett, L., & Fausett, L. V. (1994). *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*. Prentice-Hall.

Fayyad, S. M. (2012). Constructing Control System for Active Suspension System. *Undefined*. https://www.semanticscholar.org/paper/Constructing-Control-System-for-Active-Suspension-Fayyad/84eebc852d97263d4aa35954ef3d850f545fff86

Gu, D., & Wang, G. (2012). Application of PID Controller Based on BPNN in Temperature Control of Electrothermal Boiler. *Advances in Electronic Engineering, Communication and Management Vol.1*, 173–178. https://doi.org/10.1007/978-3-642-27287-5_28

Heidari, M., & Homaei, H. (2013). Design a PID Controller for Suspension System by Back Propagation Neural Network. *Journal of Engineering, 2013*, e421543. https://doi.org/10.1155/2013/421543

Hernández-Alvarado, R., García-Valdovinos, L. G., Salgado-Jiménez, T., Gómez-Espinosa, A., & Fonseca-Navarro, F. (2016). Neural Network-Based Self-Tuning PID Control for Underwater Vehicles. *Sensors (Basel, Switzerland), 16*(9), 1429. https://doi.org/10.3390/s16091429

Jocom, B. P. (2017). *Penerapan Genetic Algorithm Untuk Optimasi Peningkatan Laba Persediaan Produksi Pakaian*.

Kalaivani, R., & Lakshmi, K. S. and P. (2016a). Neural Network based Vibration Control for Vehicle Active Suspension System. *Indian Journal of Science and Technology*, *9*(1), 1–8. https://doi.org/10.17485/ijst/2016/v9i1/83806

Kalaivani, R., & Lakshmi, K. S. and P. (2016b). Neural Network based Vibration Control for Vehicle Active Suspension System. *Indian Journal of Science and Technology*, *9*(1), 1–8. https://doi.org/10.17485/ijst/2016/v9i1/83806

Kaleemullah, M., Faris, W. F., & Hasbullah, F. (2011). Design of robust H∞, fuzzy and LQR controller for active suspension of a quarter car model. *2011 4th International Conference on Mechatronics (ICOM)*, 1–6. https://doi.org/10.1109/ICOM.2011.5937197

Kashem, S., Nagarajah, R., & Ektesabi, M. (2018). *Vehicle Suspension Systems and Electromagnetic Dampers*. https://doi.org/10.1007/978-981-10-5478-5

Kusmaryanto, S. (2014). Jaringan Saraf Tiruan Backpropagation untuk Pengenalan Wajah Metode Ekstraksi Fitur Berbasis Histogram. *Jurnal EECCIS*, *8*(2), 193–198.

Lee, Y.-S., & Jang, D.-W. (2021). Optimization of Neural Network-Based Self-Tuning PID Controllers for Second Order Mechanical Systems. *Applied Sciences*, *11*(17), 8002. https://doi.org/10.3390/app11178002

Marino, A., & Neri, F. (2019). *PID Tuning with Neural Networks* (pp. 476–487). https://doi.org/10.1007/978-3-030-14799-0_41

Mouleeswaran, S. (2012). *Design and Development of PID Controller-Based Active Suspension System for Automobiles*. https://doi.org/10.5772/32611

Müderrisoğlu, K., Arisoy, D., Ahan, O., & Akdogan, E. (2016). PID Parameters Prediction Using Neural Network for A Linear Quarter Car Suspension Control. *International*

*Journal of Intelligent Systems and Applications in Engineering*, *4*, 20. https://doi.org/10.18201/ijisae.75361

Pekgökgöz, R., Gurel, M., Bilgehan, M., & Kisa, M. (2010). Active suspension of cars using fuzzy logic controller optimized by genetic algorithm. *International Journal of Engineering and Applied Sciences*, *2*.

Plewa, K. M., Eger, T. R., Oliver, M. L., & Dickey, J. P. (2012). Comparison between ISO 2631–1 Comfort Prediction Equations and Self-Reported Comfort Values during Occupational Exposure to Whole-Body Vehicular Vibration. *Journal of Low Frequency Noise, Vibration and Active Control*, *31*(1), 43–53. https://doi.org/10.1260/0263-0923.31.1.43

Pongfai, J., & Assawinchaichote, W. (2017). Self-tuning PID parameters using NN-GA for brush DC motor control system. *2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, 111–114. https://doi.org/10.1109/ECTICon.2017.8096185

Reeves, C. (2010). Genetic Algorithms. In *Handbook of Metaheuristics* (Vol. 146, pp. 109–139). https://doi.org/10.1007/978-1-4419-1665-5_5

Rs, A. K. (2012). IDENTIFIKASI DAN PERANCANGAN PENGENDALI PID MENGGUNAKAN PENDUGA ARX TERHADAP SISTEM PEMANAS UDARA DENGAN KRITERIA ISE, IAE, ITSE DAN ITAE. *BERKALA FISIKA*, *15*(4), 105–112.

Senaratna, N. (2005). Genetic Algorithms: The Crossover-Mutation Debate. *Undefined*. https://www.semanticscholar.org/paper/Genetic-Algorithms%3A-The-Crossover-Mutation-Debate-Senaratna/73a50124700c7b2e44e3a72a298f6279a8b54ac3

Sezgin, A., & Arslan, Y. Z. (2012). Analysis of the vertical vibration effects on ride comfort of vehicle driver. *Journal of Vibroengineering*, *14*, 559–571.

Sharkawy, A.-N., Ali, A., Ghazaly, N., & Abdel-Jaber, G. (2015). PID CONTROLLER OF ACTIVE SUSPENSION SYSTEM FOR A QUARTER CAR MODEL. *International Journal of Advances in Engineering & Technology*, *Vol. 8*, 899–909.

Tandel, A., Deshpande, A. R., Deshmukh, S. P., & Jagtap, K. R. (2014). Modeling, Analysis and PID Controller Implementation on Double Wishbone Suspension Using SimMechanics and Simulink. *Procedia Engineering*, *97*, 1274–1281. https://doi.org/10.1016/j.proeng.2014.12.406

Tang, C. yin, Zhao, G. yao, Li, H., & Zhou, S. wen. (2009). Research on Suspension System Based on Genetic Algorithm and Neural Network Control. *2009 Second International Conference on Intelligent Computation Technology and Automation*, *1*, 468–471. https://doi.org/10.1109/ICICTA.2009.120

Vinayak S. Dixit*, S. C. B. (2017). SEMI-ACTIVE SUSPENSION SYSTEM DESIGN FOR QUARTER CAR MODEL AND ITS ANALYSIS WITH PASSIVE SUSPENSION MODEL. *INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY*, *6*(2), 203–211. https://doi.org/10.5281/zenodo.290144

Yusoff, T., Atan, M. F., Abdul Rahman, N., Salleh, S., & Abdul Wahab, N. (2015). Optimization of PID Tuning Using Genetic Algorithm. *Journal of Applied Science & Process Engineering*, *2*. https://doi.org/10.33736/jaspe.168.2015

Yusuf, M., Fatoni, A., & Hady, A. (2017). Desain Kontroler PID-Genetic Algorithm untuk Sistem Pengaturan Level Air Steam Drum pada Pembangkit Listrik Tenaga Uap (PLTU). *Jurnal Teknik ITS*, *6*. https://doi.org/10.12962/j23373539.v6i1.21391

Ziegler, J. G., & Nichols, N. (1942). *Optimum Settings for Automatic Controllers*. https://doi.org/10.1115/1.2899060

# LAMPIRAN

## 1.Hasil Simulasi Pada MATLAB Untuk Data *Training* dan *Validating* BPNN

| | Parameter Input | | | Parameter Output |
|---|---|---|---|---|
| **Simulasi** | **$K_P$** | **$K_I$** | **$K_D$** | **ITAE** |
| 1 | 800000 | 600000 | 200000 | 0.001342 |
| 2 | | | 210000 | 0.001307 |
| 3 | | | 220000 | 0.001275 |
| 4 | | | 230000 | 0.001245 |
| 5 | | | 240000 | 0.001217 |
| 6 | | | 250000 | 0.001191 |
| 7 | | 610000 | 200000 | 0.001341 |
| 8 | | | 210000 | 0.001306 |
| 9 | | | 220000 | 0.001274 |
| 10 | | | 230000 | 0.001245 |
| 11 | | | 240000 | 0.001217 |
| 12 | | | 250000 | 0.001191 |
| 13 | | 620000 | 200000 | 0.00134 |
| 14 | | | 210000 | 0.001306 |
| 15 | | | 220000 | 0.001274 |
| 16 | | | 230000 | 0.001244 |
| 17 | | | 240000 | 0.001216 |
| 18 | | | 250000 | 0.001191 |
| 19 | | 630000 | 200000 | 0.00134 |
| 20 | | | 210000 | 0.001305 |
| 21 | | | 220000 | 0.001273 |
| 22 | | | 230000 | 0.001244 |
| 23 | | | 240000 | 0.001216 |
| 24 | | | 250000 | 0.00119 |
| 25 | | 640000 | 200000 | 0.001339 |
| 26 | | | 210000 | 0.001304 |
| 27 | | | 220000 | 0.001273 |
| 28 | | | 230000 | 0.001243 |
| 29 | | | 240000 | 0.001216 |
| 30 | | | 250000 | 0.00119 |
| 31 | | 650000 | 200000 | 0.001338 |
| 32 | | | 210000 | 0.001304 |
| 33 | | | 220000 | 0.001272 |
| 34 | | | 230000 | 0.001243 |
| 35 | | | 240000 | 0.001215 |
| 36 | | | 250000 | 0.001189 |
| 37 | 810000 | 600000 | 200000 | 0.001339 |
| 38 | | | 210000 | 0.001304 |
| 39 | | | 220000 | 0.001272 |
| 40 | | | 230000 | 0.001242 |
| 41 | | | 240000 | 0.001214 |
| 42 | | | 250000 | 0.001188 |
| 43 | | 610000 | 200000 | 0.001338 |
| 44 | | | 210000 | 0.001303 |
| 45 | | | 220000 | 0.001271 |

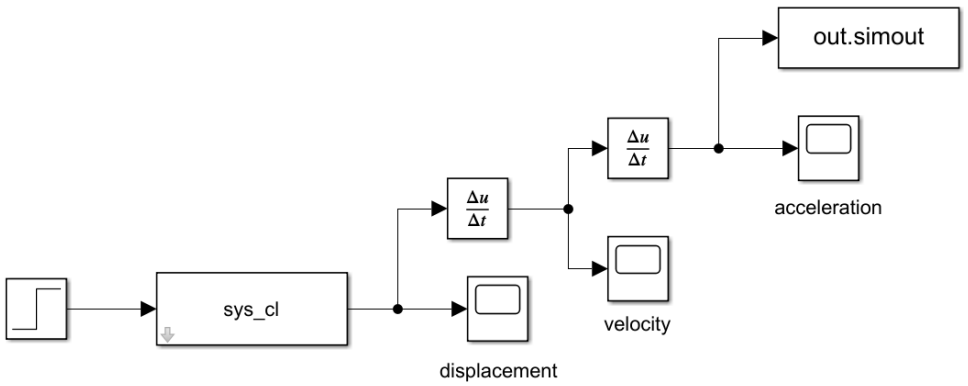
| | | | | |
|---|---|---|---|---|
| 46 | | | 230000 | 0.001241 |
| 47 | | | 240000 | 0.001213 |
| 48 | | | 250000 | 0.001187 |
| 49 | | 620000 | 200000 | 0.001338 |
| 50 | | | 210000 | 0.001303 |
| 51 | | | 220000 | 0.001271 |
| 52 | | | 230000 | 0.001241 |
| 53 | | | 240000 | 0.001213 |
| 54 | | | 250000 | 0.001187 |
| 55 | | 630000 | 200000 | 0.001337 |
| 56 | | | 210000 | 0.001302 |
| 57 | | | 220000 | 0.00127 |
| 58 | | | 230000 | 0.00124 |
| 59 | | | 240000 | 0.001213 |
| 60 | | | 250000 | 0.001187 |
| 61 | | 640000 | 200000 | 0.001336 |
| 62 | | | 210000 | 0.001301 |
| 63 | | | 220000 | 0.001269 |
| 64 | | | 230000 | 0.00124 |
| 65 | | | 240000 | 0.001212 |
| 66 | | | 250000 | 0.001186 |
| 67 | | 650000 | 200000 | 0.001335 |
| 68 | | | 210000 | 0.001301 |
| 69 | | | 220000 | 0.001269 |
| 70 | | | 230000 | 0.001239 |
| 71 | | | 240000 | 0.001212 |
| 72 | | | 250000 | 0.001186 |
| 73 | 820000 | 600000 | 200000 | 0.001336 |
| 74 | | | 210000 | 0.001301 |
| 75 | | | 220000 | 0.001269 |
| 76 | | | 230000 | 0.001238 |
| 77 | | | 240000 | 0.00121 |
| 78 | | | 250000 | 0.001184 |
| 79 | | 610000 | 200000 | 0.001336 |
| 80 | | | 210000 | 0.0013 |
| 81 | | | 220000 | 0.001268 |
| 82 | | | 230000 | 0.001238 |
| 83 | | | 240000 | 0.00121 |
| 84 | | | 250000 | 0.001184 |
| 85 | | 620000 | 200000 | 0.001335 |
| 86 | | | 210000 | 0.0013 |
| 87 | | | 220000 | 0.001268 |
| 88 | | | 230000 | 0.001238 |
| 89 | | | 240000 | 0.00121 |
| 90 | | | 250000 | 0.001184 |
| 91 | | 630000 | 200000 | 0.001334 |
| 92 | | | 210000 | 0.001299 |
| 93 | | | 220000 | 0.001267 |
| 94 | | | 230000 | 0.001237 |
| 95 | | | 240000 | 0.001209 |
| 96 | | | 250000 | 0.001183 |
| 97 | | 640000 | 200000 | 0.001334 |

| | | | | |
|---|---|---|---|---|
| 98 | | | 210000 | 0.001299 |
| 99 | | | 220000 | 0.001267 |
| 100 | | | 230000 | 0.001237 |
| 101 | | | 240000 | 0.001209 |
| 102 | | | 250000 | 0.001183 |
| 103 | | 650000 | 200000 | 0.001333 |
| 104 | | | 210000 | 0.001298 |
| 105 | | | 220000 | 0.001266 |
| 106 | | | 230000 | 0.001236 |
| 107 | | | 240000 | 0.001209 |
| 108 | | | 250000 | 0.001183 |
| 109 | 830000 | 600000 | 200000 | 0.001334 |
| 110 | | | 210000 | 0.001298 |
| 111 | | | 220000 | 0.001266 |
| 112 | | | 230000 | 0.001235 |
| 113 | | | 240000 | 0.001207 |
| 114 | | | 250000 | 0.001181 |
| 115 | | 610000 | 200000 | 0.001333 |
| 116 | | | 210000 | 0.001298 |
| 117 | | | 220000 | 0.001265 |
| 118 | | | 230000 | 0.001235 |
| 119 | | | 240000 | 0.001207 |
| 120 | | | 250000 | 0.001181 |
| 121 | | 620000 | 200000 | 0.001333 |
| 122 | | | 210000 | 0.001297 |
| 123 | | | 220000 | 0.001265 |
| 124 | | | 230000 | 0.001235 |
| 125 | | | 240000 | 0.001207 |
| 126 | | | 250000 | 0.00118 |
| 127 | | 630000 | 200000 | 0.001332 |
| 128 | | | 210000 | 0.001297 |
| 129 | | | 220000 | 0.001264 |
| 130 | | | 230000 | 0.001234 |
| 131 | | | 240000 | 0.001206 |
| 132 | | | 250000 | 0.00118 |
| 133 | | 640000 | 200000 | 0.001332 |
| 134 | | | 210000 | 0.001296 |
| 135 | | | 220000 | 0.001264 |
| 136 | | | 230000 | 0.001234 |
| 137 | | | 240000 | 0.001206 |
| 138 | | | 250000 | 0.00118 |
| 139 | | 650000 | 200000 | 0.001331 |
| 140 | | | 210000 | 0.001295 |
| 141 | | | 220000 | 0.001263 |
| 142 | | | 230000 | 0.001233 |
| 143 | | | 240000 | 0.001205 |
| 144 | | | 250000 | 0.001179 |
| 145 | 840000 | 600000 | 200000 | 0.001331 |
| 146 | | | 210000 | 0.001295 |
| 147 | | | 220000 | 0.001263 |
| 148 | | | 230000 | 0.001232 |
| 149 | | | 240000 | 0.001204 |

| | | | | |
|---|---|---|---|---|
| 150 | | | 250000 | 0.001178 |
| 151 | | 610000 | 200000 | 0.001331 |
| 152 | | | 210000 | 0.001295 |
| 153 | | | 220000 | 0.001262 |
| 154 | | | 230000 | 0.001232 |
| 155 | | | 240000 | 0.001204 |
| 156 | | | 250000 | 0.001177 |
| 157 | | 620000 | 200000 | 0.00133 |
| 158 | | | 210000 | 0.001295 |
| 159 | | | 220000 | 0.001262 |
| 160 | | | 230000 | 0.001232 |
| 161 | | | 240000 | 0.001203 |
| 162 | | | 250000 | 0.001177 |
| 163 | | 630000 | 200000 | 0.00133 |
| 164 | | | 210000 | 0.001294 |
| 165 | | | 220000 | 0.001261 |
| 166 | | | 230000 | 0.001231 |
| 167 | | | 240000 | 0.001203 |
| 168 | | | 250000 | 0.001177 |
| 169 | | 640000 | 200000 | 0.001329 |
| 170 | | | 210000 | 0.001293 |
| 171 | | | 220000 | 0.001261 |
| 172 | | | 230000 | 0.001231 |
| 173 | | | 240000 | 0.001203 |
| 174 | | | 250000 | 0.001177 |
| 175 | | 650000 | 200000 | 0.001329 |
| 176 | | | 210000 | 0.001293 |
| 177 | | | 220000 | 0.00126 |
| 178 | | | 230000 | 0.00123 |
| 179 | | | 240000 | 0.001202 |
| 180 | | | 250000 | 0.001176 |
| 181 | 850000 | 600000 | 200000 | 0.001329 |
| 182 | | | 210000 | 0.001293 |
| 183 | | | 220000 | 0.00126 |
| 184 | | | 230000 | 0.001229 |
| 185 | | | 240000 | 0.001201 |
| 186 | | | 250000 | 0.001174 |
| 187 | | 610000 | 200000 | 0.001329 |
| 188 | | | 210000 | 0.001292 |
| 189 | | | 220000 | 0.001259 |
| 190 | | | 230000 | 0.001229 |
| 191 | | | 240000 | 0.001201 |
| 192 | | | 250000 | 0.001174 |
| 193 | | 620000 | 200000 | 0.001328 |
| 194 | | | 210000 | 0.001292 |
| 195 | | | 220000 | 0.001259 |
| 196 | | | 230000 | 0.001229 |
| 197 | | | 240000 | 0.0012 |
| 198 | | | 250000 | 0.001174 |
| 199 | | 630000 | 200000 | 0.001328 |
| 200 | | | 210000 | 0.001291 |
| 201 | | | 220000 | 0.001259 |

| | | | | |
|---|---|---|---|---|
| 202 | | | 230000 | 0.001228 |
| 203 | | | 240000 | 0.0012 |
| 204 | | | 250000 | 0.001174 |
| 205 | | 640000 | 200000 | 0.001327 |
| 206 | | | 210000 | 0.001291 |
| 207 | | | 220000 | 0.001258 |
| 208 | | | 230000 | 0.001228 |
| 209 | | | 240000 | 0.0012 |
| 210 | | | 250000 | 0.001173 |
| 211 | | 650000 | 200000 | 0.001327 |
| 212 | | | 210000 | 0.00129 |
| 213 | | | 220000 | 0.001258 |
| 214 | | | 230000 | 0.001227 |
| 215 | | | 240000 | 0.001199 |
| 216 | | | 250000 | 0.001173 |

**2. Diagram Simulink Untuk Mencari Percepatan Sistem**



(*Halaman ini sengaja dikosongkan*)

## BIODATA PENULIS

Daniel Michael Radja Pande, anak kedua dari tiga bersaudara, putra kandung dari pasangan Bapak Karlin Simbolon dan Ibu Medalia Juniar Situmorang, lahir di Jakarta, 28 Mei 2000. Penulis telah menempuh Pendidikan formal di TK Santa Maria Pekanbaru, SD Santa Maria Pekanbaru, SMP Negeri 4 Pekanbaru, dan SMA Negeri 8 Pekanbaru. Setelah lulus dari jenjang SMA pada tahum 2018, penulis mengikuti SBMPTN dan diterima di Departemen Teknik Mesin FT-IRS ITS tahun 2018 dan terdaftar dengan NRP 02111840000098

Semasa Kuliah, penulis mengambil bidang sistem kontrol dan otomasi. Penulis sempat aktif dalam berbagai kegiatan berskala departemen sampai dengan internasional, penulis sempat menjadi kepala divisi *Internationalization and Development* di ITS Global Engagement Volunteers serta kepala Departemen Eksternal klub pers DIMENSI ITS, serta mengikuti Online Summer School di Istanbul Aydin University, Turki. Apabila terdapat pesan atau informasi yang hendak disampaikan kepada penulis, dapat melalui email: danielmichael285@gmail.com