



TUGAS AKHIR - TF 181801

**PERANCANGAN SISTEM KENDALI GERAK *AUTONOMOUS*  
*UNDERWATER VEHICLE (AUV)* TIPE *DOUBLE-HULL* BERBASIS  
*FUZZY-PID***

CAROLLINA KUSUMAWIDJAYA

NRP. 02311840000073

Dosen Pembimbing:

Prof. Dr. Ir. Aulia Siti Aisjah, M.T.

Ir. A. A. Masroeri, M. Eng., D. Eng.

DEPARTEMEN TEKNIK FISIKA

Fakultas Teknologi Industri dan Rekayasa Sistem

Institut Teknologi Sepuluh Nopember

Surabaya 2022

*Halaman ini sengaja dikosongkan*



**FINAL PROJECT - TF 181801**

***FUZZY-PID MOTION CONTROL DESIGN OF A DOUBLE-HULL  
AUTONOMOUS UNDERWATER VEHICLE***

CAROLLINA KUSUMAWIDJAYA

NRP. 02311840000073

Supervisors:

Prof. Dr. Ir. Aulia Siti Aisjah, M.T.

Ir. A. A. Masroeri, M. Eng. D. Eng.

*DEPARTMENT OF ENGINEERING PHYSICS*

*Faculty of Industrial Technology and System Engineering*

*Institut Teknologi Sepuluh Nopember*

*Surabaya 2022*

*Halaman ini sengaja dikosongkan*

## PERNYATAAN BEBAS PLAGIASI

Saya yang bertanda tangan di bawah ini.

Nama : Carollina Kusumawidjaya  
NRP : 02311840000073  
Departemen / Prodi : Teknik Fisika / S1 Teknik Fisika  
Fakultas : Fakultas Teknologi Industri & Rekayasa Sistem (FTIRS)  
Perguruan Tinggi : Institut Teknologi Sepuluh Nopember

Dengan ini menyatakan bahwa Tugas Akhir dengan judul **“PERANCANGAN SISTEM KENDALI GERAK *AUTONOMOUS UNDERWATER VEHICLE (AUV) TIPE DOUBLE-HULL BERBASIS FUZZY-PID*”** adalah benar karya saya sendiri dan bukan plagiat dari karya orang lain. Apabila di kemudian hari terbukti terdapat plagiat pada Tugas Akhir ini, maka saya bersedia menerima sanksi sesuai ketentuan yang berlaku.

Demikian surat pernyataan ini saya buat dengan sebenarnya-benarnya.

Surabaya, 09 Juni 2022

Yang membuat pernyataan,



Carollina Kusumawidjaya

NRP. 02311840000073

*Halaman ini sengaja dikosongkan*

**LEMBAR PENGESAHAN  
TUGAS AKHIR**

**PERANCANGAN SISTEM KENDALI GERAK *AUTONOMOUS*  
*UNDERWATER VEHICLE* (AUV) TIPE *DOUBLE-HULL* BERBASIS  
*FUZZY-PID***

Oleh:

Carollina Kusumawidjaya  
NRP. 0231184000073

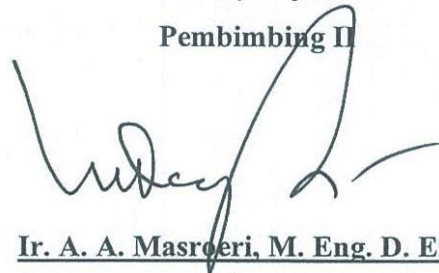
Surabaya,

Menyetujui,  
Pembimbing I



Prof. Dr. Ir. Aulia Siti Aisjah, M.T.  
NIP. 19660116 198903 2 001

Menyetujui,  
Pembimbing II



Ir. A. A. Masroeri, M. Eng. D. Eng.  
NIP. 19580807 198403 1 004

Mengetahui,

Kepala Departemen  
Teknik Fisika FTIRS-ITS



Dr. Suvanto, S.T., M.T.  
NIP. 19711111 199512 1 002

*Halaman ini sengaja dikosongkan*



## LEMBAR PENGESAHAN

### PERANCANGAN SISTEM KENDALI GERAK *AUTONOMOUS UNDERWATER VEHICLE* (AUV) TIPE *DOUBLE-HULL* BERBASIS *FUZZY-PID*

#### TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat  
Memperoleh Gelar Sarjana Teknik  
pada  
Program Studi S-1 Departemen Teknik Fisika  
Fakultas Teknologi Industri & Rekayasa Sistem (FTIRS)  
Institut Teknologi Sepuluh Nopember

Oleh:

**CAROLLINA KUSUMAWIDJAYA**

**NRP. 02311840000073**

Disetujui oleh Tim Penguji Tugas Akhir:

1. Prof. Dr. Ir. Aulia Siti Aisjah, M.T. .... (Pembimbing I)
2. Ir. A. A. Masroeri, M. Eng. D. Eng ..... (Pembimbing II)
3. Prof. Totok Ruki Biyanto, S.T., M.T., PhD ..... (Ketua Penguji)
4. Dr. Ir. Purwadi Agus Darwito, M.Sc. .... (Penguji I)
5. Dr. Katherin Indriawati, S.T., M.T. .... (Penguji II)

**SURABAYA**

**2022**

*Halaman ini sengaja dikosongkan*

**PERANCANGAN SISTEM KENDALI GERAK *AUTONOMOUS UNDERWATER VEHICLE* (AUV) TIPE *DOUBLE-HULL* BERBASIS *FUZZY-PID***

**Nama** : Carollina Kusumawidjaya  
**NRP** : 02311840000073  
**Departemen** : Teknik Fisika FTIRS – ITS  
**Dosen Pembimbing** : Prof. Dr. Ir. Aulia Siti Aisjah, M.T.  
Ir. A. A. Masroeri, M. Eng. D. Eng.

**ABSTRAK**

*Autonomous Underwater Vehicle* (AUV) merupakan salah satu alat yang dikembangkan untuk melaksanakan misi eksplorasi laut dalam. Tidak adanya operator manusia dalam AUV membuat sistem kendali gerak yang *robust* menjadi penting untuk AUV dalam menjalankan misi. AUV tipe *double-hull* memiliki kemampuan operasi jangka panjang yang efektif digunakan dalam eksplorasi laut dalam, namun umumnya bersifat *under-actuated* dan memiliki karakteristik hidrodinamika yang rumit. Tantangan lainnya dalam merancang kendali gerak untuk AUV tipe *double-hull* adalah adanya gangguan eksternal seperti arus laut yang dapat menurunkan performansi sistem kontrol. Kontroler PID yang umumnya digunakan kurang adaptif dalam mengendalikan sistem nonlinier seperti AUV di bawah gangguan. Penelitian terdahulu membuktikan bahwa algoritma adaptif seperti logika *fuzzy* terbukti mampu membantu PID melakukan *self-tuning* dan meningkatkan performansi sistem kendali gerak. Pada penelitian ini dirancang sistem kendali gerak AUV tipe *double-hull* berbasis *fuzzy-PID* dan disimulasikan untuk mencapai *waypoint* yang dituju. Hasil menunjukkan bahwa sistem kendali yang diajukan bekerja dengan lebih baik dibandingkan kontroler PID biasa, bahkan di bawah gangguan arus laut ekstrim berkecepatan 41 m/s.

**Kata Kunci:** *Autonomous Underwater Vehicle* (AUV), PID, *Fuzzy*

*Halaman ini sengaja dikosongkan*

## ***FUZZY-PID MOTION CONTROL DESIGN OF A DOUBLE-HULL AUTONOMOUS UNDERWATER VEHICLE***

***Name*** : Carrollina Kusumawidjaya  
***NRP*** : 0231184000073  
***Department*** : Engineering Physics FTIRS - ITS  
***Supervisors*** : Prof. Dr. Ir. Aulia Siti Aisjah, S.T., M.T.  
Ir. A. A. Masroeri, M. Eng., D. Eng.

### ***ABSTRACT***

*One of the technologies created to conduct deep sea exploration expeditions is the autonomous underwater vehicle (AUV). The AUV's lack of a human operator necessitates the use of a reliable motion control system. The double-hull type AUV has a long-term operational capability that is useful for deep water exploration, although it is typically under-actuated and has complex hydrodynamic characteristics. Another difficulty in designing motion control for a double-hull AUV is the existence of external disturbances like ocean currents, which might degrade the control system's performance. PID controllers are less adaptive in nonlinear systems like AUVs that are subjected to disturbances. Adaptive algorithms such as fuzzy logic have been shown in previous study to aid PID in self-tuning and improving motion control system performance. To attain the intended waypoint, a fuzzy-PID-based double-hull AUV motion control system was devised and simulated in this study. Even under extreme sea current disturbances of 41 m/s, the suggested control system outperforms typical PID controllers, according to the results.*

***Keywords:*** Autonomous Underwater Vehicle (AUV), PID, Fuzzy

*Halaman ini sengaja dikosongkan*

## KATA PENGANTAR

Segala puji syukur penulis panjatkan kehadirat Tuhan Yang Maha Esa karena berkat karunia dan arahan-Nya sehingga penulis mampu untuk menyelesaikan laporan Tugas Akhir yang berjudul:

**“PERANCANGAN SISTEM KENDALI GERAK *AUTONOMOUS UNDERWATER VEHICLE (AUV) TIPE DOUBLE-HULL* BERBASIS *FUZZY-PID*”**

Laporan Tugas Akhir ini disusun untuk memenuhi persyaratan agar dapat memperoleh gelar Sarjana Teknik di Departemen Teknik Fisika Institut Teknologi Sepuluh Nopember. Penulis juga menerima banyak sekali bantuan yang datang dari berbagai pihak selama penyusunan laporan Tugas Akhir ini, khususnya kepada:

1. Bapak Dr. Suyanto, S.T., M.T. selaku kepala Departemen Teknik Fisika ITS yang memberikan dukungan secara moral dan administratif.
2. Ibu Prof. Dr. Ir. Aulia Siti Aisjah, MT. dan Bapak Dr. Ir. A. A. Masroeri, M. Eng. selaku dosen pembimbing tugas akhir yang mendampingi selama proses pengerjaan Tugas Akhir.
3. Bapak dan Ibu dosen serta staff tata usaha Teknik Fisika yang telah membantu penulis selama masa perkuliahan baik berupa ilmu maupun bantuan administratif.
4. Kedua orang tua penulis yang selalu memberikan usaha terbaiknya dalam mendoakan dan memfasilitasi kebutuhan penulis.

Serta pihak-pihak lain yang tidak dapat disebutkan satu-persatu. Semoga laporan tugas akhir ini dapat dipergunakan dengan sebaik-baiknya.

Surabaya, 09 Juni 2022

Penulis

*Halaman ini sengaja dikosongkan*



## DAFTAR ISI

HALAMAN JUDUL.....	i
COVER PAGE.....	iii
PERNYATAAN BEBAS PLAGIASI .....	v
LEMBAR PENGESAHAN .....	vii
LEMBAR PENGESAHAN .....	ix
ABSTRAK .....	xi
<i>ABSTRACT</i> .....	xiii
KATA PENGANTAR .....	xv
DAFTAR ISI.....	xvii
DAFTAR GAMBAR .....	xix
DAFTAR TABEL.....	xxv
DAFTAR SIMBOL.....	xxvii
BAB I PENDAHULUAN .....	1
1.1 Latar Belakang .....	1
1.2 Rumusan Masalah .....	4
1.3 Tujuan.....	4
1.4 Batasan Masalah.....	4
1.5 Sistematika Laporan .....	5
BAB II TINJAUAN PUSTAKA DAN DASAR TEORI.....	7
2.1 Pengendalian AUV .....	7
2.2 Pemodelan Dinamik <i>Autonomous Underwater Vehicle (AUV)</i> .....	9
2.3 Gangguan Arus Laut .....	15
2.4 <i>Guidance</i> dan Kontrol .....	17
2.5 Analisa Performansi Sistem Pengendali.....	18

2.6	Pengendali PID .....	21
2.7	<i>Anti-windup</i> .....	23
2.8	Logika <i>Fuzzy</i> .....	24
BAB III METODOLOGI PENELITIAN .....		27
3.1	Identifikasi Masalah.....	28
3.2	Studi Literatur .....	28
3.3	Pengumpulan Data Spesifikasi AUV.....	28
3.4	Pemodelan Dinamik AUV .....	34
3.5	Perancangan Pengendali PID .....	41
3.6	Perancangan Pengendali <i>Fuzzy</i> -PID .....	53
3.7	Simulasi dengan Gangguan Arus Laut .....	64
3.8	Simulasi dengan <i>Waypoint Following Guidance</i> .....	65
3.9	Analisa Perbandingan Hasil Simulasi PID dan <i>Fuzzy</i> -PID .....	70
BAB IV HASIL DAN PEMBAHASAN.....		73
4.1	Analisa Hasil Perancangan Sistem Pengendali PID .....	73
4.2	Analisa Hasil Perancangan Sistem Pengendali <i>Fuzzy</i> -PID.....	83
4.3	Perbandingan Performansi Uji <i>Closed Loop</i> .....	90
4.4	Perbandingan Hasil Simulasi <i>Waypoint Following</i> .....	95
BAB V KESIMPULAN DAN SARAN .....		129
5.1	Kesimpulan .....	129
5.2	Saran .....	130
DAFTAR PUSTAKA.....		131
LAMPIRAN .....		135
BIODATA PENULIS.....		141

## DAFTAR GAMBAR

<b>Gambar 1. 1</b> AUV <i>Single-body</i> (Keller, 2016) dan <i>Multi-body</i> (Ferrini, Singh, Clarke, Wakefield, & K., 2006) .....	2
<b>Gambar 2. 1</b> <i>Body-fixed Reference Frame</i> dan <i>NED Reference Frame</i> .....	9
<b>Gambar 2. 2</b> Pengukuran <i>Angle of Attack</i> dan <i>Side Slip Angle</i> pada AUV (Hammad, Elshenawy, & Singaby, 2017) .....	17
<b>Gambar 2. 3</b> Respons Waktu (Jennings, 2016) .....	18
<b>Gambar 2. 4</b> Diagram Bode (Bolton W. , 2004) .....	20
<b>Gambar 2. 5</b> Skema Rangkaian PID (WangReady, 2012) .....	21
<b>Gambar 2. 6</b> Skema Rangkaian <i>Closed-Loop</i> dengan Aktuator Bersaturasi (Alaoui, Ayad, & Doubabi, 2006).....	23
<b>Gambar 2. 7</b> Skema <i>Anti-Windup Back-Calculation</i> (Åström & Wittenmark, <i>Computer-Controlled Systems</i> , 2011).....	24
<b>Gambar 2. 8</b> Arsitektur Logika <i>Fuzzy</i> (Usta, Akyazi, & Altaş, 2011) .....	25
<b>Gambar 3. 1</b> Diagram Alir Metodologi Penelitian .....	27
<b>Gambar 3. 2</b> AUV Tipe <i>Double-Hull</i> yang Dikembangkan di CEiiA.....	29
<b>Gambar 3. 3</b> Diagram Blok Perancangan Pengendali PID .....	41
<b>Gambar 3. 4</b> Diagram Blok Subsistem Pengendali Kecepatan ( <i>Surge</i> ) .....	43
<b>Gambar 3. 5</b> Diagram Blok Subsistem Pengendali Kedalaman ( <i>Heave</i> ) .....	45
<b>Gambar 3. 6</b> Diagram Blok Subsistem Pengendali Kedalaman ( <i>Pitch</i> ).....	46
<b>Gambar 3. 7</b> Diagram Blok Subsistem Pengendali Haluan ( <i>Yaw</i> ) .....	47
<b>Gambar 3. 8</b> <i>Anti-Windup Back Calculation</i> .....	49
<b>Gambar 3. 9</b> Subsistem Pengendali PID dengan <i>Anti-Windup Back-Calculation</i> .....	50
<b>Gambar 3. 10</b> Diagram Blok Perancangan Pengendali <i>Fuzzy</i> -PID .....	53
<b>Gambar 3. 11</b> Struktur Logika <i>Fuzzy</i> untuk <i>Self-Tuning</i> PID .....	54
<b>Gambar 3. 12</b> Fungsi Keanggotaan Masukan 1 ( <i>Error</i> ).....	54
<b>Gambar 3. 13</b> Fungsi Keanggotaan Masukan 2 ( <i>Error Rate</i> ).....	55

<b>Gambar 3. 14</b> Fungsi Keanggotaan Keluaran $dKp$ , $dKi$ , atau $dKd$ .....	55
<b>Gambar 3. 15</b> Representasi Simulink dari Struktur Logika <i>Fuzzy</i> yang Digunakan .....	56
<b>Gambar 3. 16</b> Penampang Permukaan <i>Rule Base</i> $dKp$ .....	57
<b>Gambar 3. 17</b> Penampang Permukaan <i>Rule Base</i> $dKi$ .....	58
<b>Gambar 3. 18</b> Penampang Permukaan <i>Rule Base</i> $dKd$ .....	59
<b>Gambar 3. 19</b> Grafik Contoh <i>Error</i> dan <i>Error Rate</i> terhadap Waktu.....	59
<b>Gambar 3. 20</b> Grafik Perubahan $Kp$ terhadap <i>Error</i> yang Telah Dinormalisasi .	60
<b>Gambar 3. 21</b> Grafik Perubahan $Ki$ terhadap <i>Error</i> yang Telah Dinormalisasi ..	61
<b>Gambar 3. 22</b> Grafik Perubahan $Kd$ terhadap <i>Error</i> yang Telah Dinormalisasi .	62
<b>Gambar 3. 23</b> Skema <i>Self-Tuning</i> PID.....	63
<b>Gambar 3. 24</b> Kecepatan Arus Laut Minimum, Maksimum dan Rata-rata di Lepas Pantai (Driscoll, et al., 2008) .....	65
<b>Gambar 3. 25</b> Diagram Alir <i>Waypoint Following Guidance</i> .....	66
<b>Gambar 3. 26</b> Hubungan antara Parameter $ks$ dengan Referensi Kecepatan dan Jarak (Mendes, 2017).....	68
<b>Gambar 3. 27</b> Fungsi Mode Pembebanan dengan $ucmS = udmS = 0.5ms$ ; $ucmI = udmI = 0.3$ ; dan $\sigma cm *= \sigma dm *= 0.05$ (Mendes, 2017) .....	69
<b>Gambar 4. 1</b> Respon Uji <i>Open Loop</i> pada Subsistem Pengendali <i>Surge</i> .....	74
<b>Gambar 4. 2</b> Uji <i>Closed Loop</i> pada Subsistem Pengendali <i>Surge</i> .....	74
<b>Gambar 4. 3</b> Diagram Bode Uji Stabilitas Subsistem Pengendali <i>Surge</i> .....	75
<b>Gambar 4. 4</b> Diagram <i>Root Locus</i> Subsistem Pengendali <i>Surge</i> .....	76
<b>Gambar 4. 5</b> Respon Uji <i>Open Loop</i> pada Subsistem Pengendali <i>Heave</i> .....	77
<b>Gambar 4. 6</b> Uji <i>Closed Loop</i> pada Subsistem Pengendali <i>Heave</i> .....	77
<b>Gambar 4. 7</b> Diagram Bode Uji Stabilitas Subsistem Pengendali <i>Heave</i> .....	78
<b>Gambar 4. 8</b> Diagram <i>Root Locus</i> Subsistem Pengendali <i>Heave</i> .....	78
<b>Gambar 4. 9</b> Respon Uji <i>Open Loop</i> pada Subsistem Pengendali <i>Pitch</i> .....	79
<b>Gambar 4. 10</b> Uji <i>Closed Loop</i> pada Subsistem Pengendali <i>Pitch</i> .....	80
<b>Gambar 4. 11</b> Diagram Bode Uji Stabilitas Subsistem Pengendali <i>Pitch</i> .....	80

<b>Gambar 4. 12</b>	Diagram <i>Root Locus</i> Subsistem Pengendali <i>Pitch</i> .....	81
<b>Gambar 4. 13</b>	Respon Uji <i>Open Loop</i> pada Subsistem Pengendali <i>Yaw</i> .....	81
<b>Gambar 4. 14</b>	Uji <i>Closed Loop</i> pada Subsistem Pengendali <i>Yaw</i> .....	82
<b>Gambar 4. 15</b>	Diagram Bode Uji Stabilitas Subsistem Pengendali <i>Yaw</i> .....	82
<b>Gambar 4. 16</b>	Diagram <i>Root Locus</i> Subsistem Pengendali <i>Yaw</i> .....	83
<b>Gambar 4. 17</b>	Respons Sistem terhadap Waktu – Iterasi Variasi <i>Range dKp</i> dan <i>dKi</i> Kontroler <i>Fuzzy-PI Surge</i> .....	85
<b>Gambar 4. 18</b>	Respons Sistem terhadap Waktu – Iterasi Variasi <i>Range dKp</i> dan <i>dKi</i> Kontroler <i>Fuzzy-PI Heave</i> .....	86
<b>Gambar 4. 19</b>	Respons Sistem terhadap Waktu – Iterasi Variasi <i>Range dKp</i> , <i>dKi</i> , dan <i>dKd</i> Kontroler <i>Fuzzy PID Pitch</i> .....	87
<b>Gambar 4. 20</b>	Respons Sistem terhadap Waktu – Iterasi Variasi <i>Range dKp</i> dan <i>dKd</i> Kontroler <i>Fuzzy PD Yaw</i> .....	89
<b>Gambar 4. 21</b>	Perbandingan <i>Fuzzy-PI</i> dengan <i>PI</i> untuk Pengendali <i>Surge</i> .....	90
<b>Gambar 4. 22</b>	Perbandingan <i>Fuzzy-PI</i> dengan <i>PI</i> untuk Pengendali <i>Heave</i> .....	91
<b>Gambar 4. 23</b>	Perbandingan <i>Fuzzy-PID</i> dengan <i>PID</i> untuk Pengendali <i>Pitch</i> ....	92
<b>Gambar 4. 24</b>	Uji <i>Closed Loop Fuzzy-PID Pitch</i> .....	93
<b>Gambar 4. 25</b>	Perbandingan <i>Fuzzy-PD</i> dengan <i>PD</i> untuk Pengendali <i>Yaw</i> .....	94
<b>Gambar 4. 26</b>	Lintasan X-Y-Z Simulasi tanpa Gangguan Arus Laut dengan.....	96
<b>Gambar 4. 27</b>	Lintasan X-Y Simulasi tanpa Gangguan Arus Laut dengan (a) <i>PID</i> ; (b) <i>Fuzzy-PID</i> .....	97
<b>Gambar 4. 28</b>	Lintasan X-Z Simulasi tanpa Gangguan Arus Laut dengan (a) <i>PID</i> ; (b) <i>Fuzzy-PID</i> .....	98
<b>Gambar 4. 29</b>	Respon <i>Surge</i> pada Simulasi <i>Waypoint Following</i> tanpa Gangguan dengan: (a) <i>PI</i> ; (b) <i>Fuzzy-PI</i> .....	99
<b>Gambar 4. 30</b>	Respon <i>Heave</i> pada Simulasi <i>Waypoint Following</i> tanpa Gangguan dengan: (a) <i>PI</i> ; (b) <i>Fuzzy-PI</i> .....	100
<b>Gambar 4. 31</b>	Respon <i>Pitch</i> pada Simulasi <i>Waypoint Following</i> tanpa Gangguan dengan: (a) <i>PID</i> ; (b) <i>Fuzzy-PID</i> .....	101
<b>Gambar 4. 32</b>	Respon <i>Yaw</i> pada Simulasi <i>Waypoint Following</i> tanpa Gangguan dengan: (a) <i>PD</i> ; (b) <i>Fuzzy-PD</i> .....	102

<b>Gambar 4. 33</b>	Kecepatan Arus Laut terhadap Waktu (Variasi 0,2 m/s) .....	103
<b>Gambar 4. 34</b>	Lintasan X-Y-Z Simulasi dengan Gangguan Arus Laut $V_c = 0,2 \text{ m/s}$ Menggunakan: (a) PID; (b) <i>Fuzzy</i> -PID .....	104
<b>Gambar 4. 35</b>	Lintasan X-Y Simulasi dengan Gangguan Arus Laut $V_c = 0,2 \text{ m/s}$ Menggunakan: (a) PID; (b) <i>Fuzzy</i> -PID .....	105
<b>Gambar 4. 36</b>	Lintasan X-Z Simulasi dengan Gangguan Arus Laut $V_c = 0,2 \text{ m/s}$ Menggunakan: (a) PID; (b) <i>Fuzzy</i> -PID .....	106
<b>Gambar 4. 37</b>	Respon <i>Surge</i> pada Simulasi <i>Waypoint Following</i> dengan Gangguan Arus Laut $V_c = 0,2 \text{ m/s}$ Menggunakan: (a) PI; (b) <i>Fuzzy</i> -PI ....	107
<b>Gambar 4. 38</b>	Respon <i>Heave</i> pada Simulasi <i>Waypoint Following</i> dengan Gangguan Arus Laut $V_c = 0,2 \text{ m/s}$ Menggunakan: (a) PI; (b) <i>Fuzzy</i> -PI .....	108
<b>Gambar 4. 39</b>	Respon <i>Pitch</i> pada Simulasi <i>Waypoint Following</i> dengan Gangguan Arus Laut $V_c = 0,2 \text{ m/s}$ Menggunakan: (a) PID; (b) <i>Fuzzy</i> -PID .....	109
<b>Gambar 4. 40</b>	Respon <i>Yaw</i> pada Simulasi <i>Waypoint Following</i> dengan Gangguan Arus Laut $V_c = 0,2 \text{ m/s}$ Menggunakan: (a) PD; (b) <i>Fuzzy</i> -PD .	110
<b>Gambar 4. 41</b>	Kecepatan Arus Laut terhadap Waktu (Variasi 40 m/s) .....	111
<b>Gambar 4. 42</b>	Lintasan X-Y-Z Simulasi dengan Gangguan Arus Laut $V_c = 40 \text{ m/s}$ Menggunakan: (a) PID; (b) <i>Fuzzy</i> -PID .....	112
<b>Gambar 4. 43</b>	Lintasan X-Y Simulasi dengan Gangguan Arus Laut $V_c = 40 \text{ m/s}$ Menggunakan: (a) PID; (b) <i>Fuzzy</i> -PID .....	113
<b>Gambar 4. 44</b>	Lintasan X-Z Simulasi dengan Gangguan Arus Laut $V_c = 40 \text{ m/s}$ Menggunakan: (a) PID; (b) <i>Fuzzy</i> -PID .....	114
<b>Gambar 4. 45</b>	Respon <i>Surge</i> pada Simulasi <i>Waypoint Following</i> dengan Gangguan Arus Laut $V_c = 40 \text{ m/s}$ Menggunakan: (a) PI; (b) <i>Fuzzy</i> -PI .....	115
<b>Gambar 4. 46</b>	Respon <i>Heave</i> pada Simulasi <i>Waypoint Following</i> dengan Gangguan Arus Laut $V_c = 40 \text{ m/s}$ Menggunakan: (a) PI; (b) <i>Fuzzy</i> -PI .....	116
<b>Gambar 4. 47</b>	Respon <i>Pitch</i> pada Simulasi <i>Waypoint Following</i> dengan Gangguan Arus Laut $V_c = 40 \text{ m/s}$ Menggunakan: (a) PID; (b) <i>Fuzzy</i> -PID	117

<b>Gambar 4. 48</b>	Respon <i>Yaw</i> pada Simulasi <i>Waypoint Following</i> dengan Gangguan Arus Laut $V_c = 40 \text{ m/s}$ Menggunakan: (a) PD; (b) <i>Fuzzy</i> -PD..	118
<b>Gambar 4. 49</b>	Kecepatan Arus Laut terhadap Waktu (Variasi 41 m/s).....	119
<b>Gambar 4. 50</b>	Lintasan X-Y-Z Simulasi dengan Gangguan Arus Laut $V_c = 41 \text{ m/s}$ Menggunakan: (a) PID; (b) <i>Fuzzy</i> -PID.....	120
<b>Gambar 4. 51</b>	Lintasan X-Y Simulasi dengan Gangguan Arus Laut $V_c = 41 \text{ m/s}$ Menggunakan: (a) PID; (b) <i>Fuzzy</i> -PID .....	121
<b>Gambar 4. 52</b>	Lintasan X-Z Simulasi dengan Gangguan Arus Laut $V_c = 41 \text{ m/s}$ Menggunakan: (a) PID; (b) <i>Fuzzy</i> -PID .....	122
<b>Gambar 4. 53</b>	Respon <i>Surge</i> pada Simulasi <i>Waypoint Following</i> dengan Gangguan Arus Laut $V_c = 41 \text{ m/s}$ Menggunakan: (a) PI; (b) <i>Fuzzy</i> -PI.....	123
<b>Gambar 4. 54</b>	Respon <i>Heave</i> pada Simulasi <i>Waypoint Following</i> dengan Gangguan Arus Laut $V_c = 41 \text{ m/s}$ Menggunakan: (a) PI; (b) <i>Fuzzy</i> -PI.....	124
<b>Gambar 4. 55</b>	Respon <i>Pitch</i> pada Simulasi <i>Waypoint Following</i> dengan Gangguan Arus Laut $V_c = 41 \text{ m/s}$ Menggunakan: (a) PID; (b) <i>Fuzzy</i> -PID	125
<b>Gambar 4. 56</b>	Respon <i>Yaw</i> pada Simulasi <i>Waypoint Following</i> dengan Gangguan Arus Laut $V_c = 41 \text{ m/s}$ Menggunakan: (a) PD; (b) <i>Fuzzy</i> -PD..	126
<b>Gambar A. 1</b>	Rangkaian Simulink untuk Sistem Kendali Berbasis PID.....	135
<b>Gambar A. 2</b>	Rangkaian Simulink untuk Sistem Kendali Berbasis <i>Fuzzy</i> -PID	135

*Halaman ini sengaja dikosongkan*



## DAFTAR TABEL

<b>Tabel 2. 1</b> Notasi 6 DOF AUV (Hammad, Elshenawy, & Singaby, 2017).....	10
<b>Tabel 3. 1</b> Dimensi AUV Tipe <i>Double-Hull</i> di CEiiA dari Notasi Gambar 3.2 ..	29
<b>Tabel 3. 2</b> Massa dan Inersia dari AUV Tipe <i>Double-Hull</i> di CEiiA .....	30
<b>Tabel 3. 3</b> Koefisien Hidrodinamis AUV Tipe <i>Double-Hull</i> di CEiiA akibat <i>Added Mass</i> .....	30
<b>Tabel 3. 4</b> Koefisien Hidrodinamis AUV Tipe <i>Double-Hull</i> di CEiiA akibat <i>Damping</i> .....	31
<b>Tabel 3. 5</b> Data Spesifikasi <i>Thruster</i> .....	34
<b>Tabel 3. 6</b> Tabel Iterasi untuk Memilih <i>Range dKp, dKi, dan dKd</i> Terbaik .....	56
<b>Tabel 3. 7</b> <i>Rule Base</i> untuk <i>dKp</i> .....	57
<b>Tabel 3. 8</b> <i>Rule Base</i> untuk <i>dKi</i> .....	58
<b>Tabel 3. 9</b> <i>Rule Base</i> untuk <i>dKd</i> .....	58
<b>Tabel 3. 10</b> Koordinat <i>Waypoint</i> yang Dituju dalam Simulasi.....	70
<b>Tabel 3. 11</b> Parameter Simulasi <i>Waypoint Following Guidance</i> (Mendes, 2017).....	70
<b>Tabel 4. 1</b> Parameter PID dan <i>Gain Anti-Windup</i> .....	73
<b>Tabel 4. 2</b> <i>Range Error</i> dan <i>Error Rate</i> untuk Menormalisasi Masukan Logika <i>Fuzzy</i> .....	84
<b>Tabel 4. 3</b> Data Iterasi Variasi <i>Range dKp</i> dan <i>dKi</i> Kontroler <i>Fuzzy-PI Surge</i> ..	84
<b>Tabel 4. 4</b> Data Iterasi Variasi <i>Range dKp</i> dan <i>dKi</i> Kontroler <i>Fuzzy-PI Heave</i> .	86
<b>Tabel 4. 5</b> Data Iterasi Variasi <i>Range dKp , dKi , dan dKd</i> Kontroler <i>Fuzzy-PID Pitch</i> .....	87
<b>Tabel 4. 6</b> Data Iterasi Variasi <i>Range dKp</i> dan <i>dKd</i> Kontroler <i>Fuzzy-PD Yaw</i> ..	88
<b>Tabel 4. 7</b> Parameter <i>Fuzzy-PID</i> .....	89
<b>Tabel 4. 8</b> Perbandingan Performansi <i>Fuzzy-PI</i> dengan <i>PI</i> untuk <i>Surge</i> .....	91
<b>Tabel 4. 9</b> Perbandingan Performansi <i>Fuzzy-PI</i> dengan <i>PI</i> untuk <i>Heave</i> .....	92
<b>Tabel 4. 10</b> Perbandingan Performansi <i>Fuzzy-PID</i> dengan <i>PID</i> untuk <i>Pitch</i> .....	93
<b>Tabel 4. 11</b> Perbandingan Performansi <i>Fuzzy-PD</i> dengan <i>PD</i> untuk <i>Yaw</i> .....	94

*Halaman ini sengaja dikosongkan*

## DAFTAR SIMBOL

$\{n\}$	NED ( <i>North-East-Down</i> ) reference frame
$\{b\}$	<i>Body-fixed</i> reference frame
$\eta$	Vektor posisi dan sudut Euler
$v$	Vektor kecepatan linier dan angular
$\tau$	Vektor gaya dan momen
$X$	Gaya yang bekerja pada AUV dan sejajar sumbu-x
$Y$	Gaya yang bekerja pada AUV dan sejajar sumbu-y
$Z$	Gaya yang bekerja pada AUV dan sejajar sumbu-z
$K$	Torsi yang bekerja pada AUV dan sejajar sumbu-x
$M$	Torsi yang bekerja pada AUV dan sejajar sumbu-y
$N$	Torsi yang bekerja pada AUV dan sejajar sumbu-z
$u$	<i>Surge</i> (kecepatan linier sumbu-x)
$v$	<i>Sway</i> (kecepatan linier sumbu-y)
$w$	<i>Heave</i> (kecepatan linier sumbu-z)
$x$	Koordinat posisi AUV di sumbu-x
$y$	Koordinat posisi AUV di sumbu-y
$z$	Koordinat posisi AUV di sumbu-z
$\phi$	<i>Roll</i> (sudut Euler)
$\theta$	<i>Pitch</i> (sudut Euler)
$\psi$	<i>Yaw</i> (sudut Euler)

$J$	Matriks transformasi
$M_{RB}$	Matriks inersia <i>rigid body</i>
$C_{RB}$	Matriks Coriolis <i>rigid body</i>
$\tau_{RB}$	Vektor gaya dan momen eksternal pada <i>rigid body</i>
$\tau_A$	Vektor gaya dan momen akibat massa tambahan hidrodinamis
$\tau_D$	Vektor gaya dan momen akibat <i>damping</i>
$L$	Matriks <i>mapping</i> (alokasi kontrol)
$U$	Vektor kontrol
$M_A$	Matriks massa tambahan
$C_A$	Matriks Coriolis akibat massa tambahan
$D$	Matriks redaman hidrodinamis ( <i>damping</i> )
$D_l$	Redaman hidrodinamis ( <i>damping</i> ) linier
$D_q$	Redaman hidrodinamis ( <i>damping</i> ) kuadratik
$r_g^B$	Vektor pusat gravitasi terhadap <i>body-fixed frame</i>
$g$	Gaya pemulih
$v_r$	Vektor kecepatan relatif AUV
$v_c^B$	Vektor kecepatan gangguan arus laut terhadap <i>body-fixed frame</i>
$v_c^E$	Vektor kecepatan gangguan arus laut terhadap NED- <i>frame</i>
$V_c$	Besar kecepatan arus laut
$t$	Waktu simulasi
$\omega$	<i>Zero-mean Gaussian white noise</i>
$\mu_0$	Konstanta pada gangguan arus laut

$\alpha_c$	<i>Angle of attack</i>
$\beta_c$	<i>Sideslip angle</i>
$T_r$	<i>Rise time</i>
$T_p$	<i>Peak time</i>
$M_p$	<i>Maximum overshoot</i>
$T_s$	<i>Settling time</i>
$e_{ss}$	<i>Error steady-state</i>
$e$	<i>Error</i>
$K_p$	<i>Gain proporsional</i>
$K_i$	<i>Gain integral</i>
$K_d$	<i>Gain derivatif</i>
$T_d$	<i>Derivative time</i>
$T_i$	<i>Integral time</i>
$T_t$	<i>Time constant untuk anti-windup back-calculation</i>
$I$	<i>Inersia</i>
$r_{Tn}^g$	<i>Vektor alokasi thruster relatif terhadap center of gravity (CG)</i>
$\tau_n$	<i>Sinyal kontrol yang dihasilkan oleh kontroler (<math>n = 1,3,5,6</math>)</i>
$dK_p$	<i>Nilai tambahan untuk gain proporsional</i>
$dK_i$	<i>Nilai tambahan untuk gain integral</i>
$dK_d$	<i>Nilai tambahan untuk gain derivatif</i>
$d_k$	<i>Jarak antara AUV dengan titik pusat COA waypoint ke-k</i>
$\rho_k$	<i>Radius center of acceptance (COA) waypoint ke-k bidang X-Y</i>

$k_u$	Batas maksimum kecepatan AUV <i>surge</i>
$k_s$	Parameter <i>tuning</i> untuk <i>guidance</i>
$d_u$	Jarak planar bidang X-Y antara AUV ke <i>waypoint</i>
$\rho_u$	Radius <i>center of acceptance</i> untuk <i>surge</i>
$W_{cm}$	Pembebanan untuk <i>common mode</i> vertikal
$W_{dm}$	Pembebanan untuk <i>differential mode</i> vertikal
$u_{cmI}$	Batas bawah transisi kecepatan untuk <i>common mode</i> vertikal
$u_{cmS}$	Batas atas transisi kecepatan untuk <i>common mode</i> vertikal
$u_{dmI}$	Batas bawah transisi kecepatan untuk <i>differential mode</i> vertikal
$u_{dmS}$	Batas atas transisi kecepatan untuk <i>differential mode</i> vertikal
$x_k$	Koordinat <i>waypoint</i> ke- $k$ pada sumbu-x
$y_k$	Koordinat <i>waypoint</i> ke- $k$ pada sumbu-y
$z_k$	Koordinat <i>waypoint</i> ke- $k$ pada sumbu-z
$u_d$	Nilai <i>surge</i> yang diinginkan
$w_d$	Nilai <i>heave</i> yang diinginkan
$\theta_d$	Nilai <i>pitch</i> yang diinginkan
$\psi_d$	Nilai <i>yaw</i> yang diinginkan
$k_w$	Batas maksimum kecepatan AUV <i>heave</i>
$e_z$	<i>Error</i> kedalaman (sumbu-z)
$\varepsilon_r$	Nilai maksimum <i>error</i> kedalaman yang ditoleransi

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Laut menyimpan sumber daya alam dan potensi yang melimpah untuk dimanfaatkan. Eksplorasi laut merupakan upaya yang telah dilakukan dan selalu dikembangkan demi memperluas pengetahuan manusia akan apa yang ada di dalam laut dan bagaimana memanfaatkannya, namun data menunjukkan bahwa baru 5% laut berhasil dieksplorasi oleh manusia (Petsko, 2020). Secara fisik, manusia memiliki keterbatasan dalam melakukan eksplorasi laut. Manusia hanya dapat menyelam hingga kedalaman tertentu dan dalam jangka waktu yang singkat karena memerlukan oksigen untuk pernafasan. Oleh sebab itu, dikembangkan suatu alat yang mampu menyelam dalam jangka waktu tertentu dan dapat dikendalikan untuk melaksanakan misi eksplorasi laut, khususnya laut dalam, seperti inspeksi pipa migas dan pemetaan kondisi dasar laut untuk eksplorasi sumber daya mineral.

Alat yang dikembangkan untuk menggantikan manusia dalam menyelam dan melakukan misi bawah laut adalah *Remotely Operated Vehicle* (ROV) dan *Automated Underwater Vehicle* (AUV). Keduanya sama-sama mampu bergerak di bawah laut dengan enam derajat kebebasan (6 DOF). Perbedaan yang signifikan di antara keduanya adalah ROV masih perlu dikendalikan oleh manusia melalui suatu pos kendali di permukaan dan umumnya memakai kabel untuk penyaluran informasi maupun suplai daya sehingga pemakaiannya masih memiliki banyak keterbatasan (Mendes, 2017). AUV tidak memerlukan operator manusia untuk mengendalikan, dan mampu membawa suplai daya serta menyimpan informasi sehingga ruang geraknya tidak dibatasi oleh kabel seperti ROV. Kendati demikian, tidak adanya operator manusia membuat AUV memerlukan sistem kontrol yang *robust* sehingga mampu menjalankan skenario kontrol tertentu seperti menyelam (*diving*), mempertahankan posisinya (*station-keeping*), mengikuti jalur yang telah ditentukan (*path following*), dan mencapai titik posisi yang diinginkan (*waypoint following*).

AUV sendiri memiliki beberapa macam jenis. Berdasarkan jumlah badan penyusunnya, AUV dibedakan menjadi *single-body* dan *multi-body* (Kang, Yu, Zhang, & Jin, 2020). AUV tipe *single-body*, seperti AUV tipe REMUS, mampu bergerak dengan kecepatan tinggi dan memiliki detection range yang cukup luas, namun AUV tipe *multi-body* memiliki kemampuan operasi jangka panjang dan sangat efektif untuk digunakan dalam eksplorasi laut dalam, seperti AUV tipe SeaBED yang memiliki dua lambung (*double hull*) yang terkoneksi secara *rigid* atau kaku. Kendati demikian, AUV tipe *multi-body* pada umumnya bersifat *under-actuated*, yakni jumlah aktuator lebih sedikit dibandingkan derajat kebebasannya, dan memiliki karakteristik hidrodinamika yang lebih rumit sehingga mempengaruhi perancangan sistem kontrol.



**Gambar 1. 1** AUV *Single-body* (Keller, 2016) dan *Multi-body* (Ferrini, Singh, Clarke, Wakefield, & K., 2006)

Tantangan lainnya dalam merancang strategi kontrol AUV adalah adanya gangguan eksternal dari lingkungan yang mampu mengurangi performa dari sistem kontrol dalam menggerakkan AUV, seperti gelombang laut, arus laut, dan angin. Di laut dalam, gangguan gelombang laut dapat diabaikan karena efek dari energi yang dihasilkan oleh gelombang laut berkurang seiring dengan bertambahnya kedalaman (Garcia, Medina, Hernandez, Alain, & Yuniesky, 2019), sedangkan gangguan arus laut yang dihasilkan oleh gravitasi dan variasi densitas air masih mempengaruhi (Fossen, *How to Incorporate Wind, Waves, and Ocean Currents in the Marine Craft Equations of Motion*, 2012). Memodelkan gangguan arus laut dan mengetahui pengaruhnya bagi gerakan AUV dapat memberikan pertimbangan dalam merancang strategi kontrol maupun navigasi yang *robust* (Vu, et al., 2021).



Pengendali PID merupakan algoritma kendali gerak AUV yang klasik dan masih banyak digunakan (Yang, 2016). Cara kerja PID adalah dengan menentukan parameter *gain* dari proporsional, integral, dan derivatif untuk menentukan sinyal kontrol yang tepat sesuai dengan nilai *error* sebagai masukan. Kendati sederhana dan mudah diimplementasikan, pengendali PID memiliki beberapa kekurangan, seperti: *gain* yang tidak dapat diubah setelah proses *tuning* sehingga kurang adaptif dan performanya kurang memuaskan untuk mengendalikan sistem yang non-linier seperti AUV (Xiang, Yu, Lapierre, & Zhang, 2017). Pengendali PID untuk AUV tipe *double hull* yang sedang dikembangkan di CEiiA, Portugal untuk menjalankan misi pemetaan dasar laut (*seabed mapping*) masih memiliki kemampuan yang inferior bila dibandingkan dengan sistem pengendali lain, yakni *Linear Quadratic Regulator* (LQR) dan belum mempertimbangkan gangguan eksternal seperti arus laut dalam menganalisa performansi kontroler yang dirancang (Mendes, 2017).

Untuk mengatasi kelemahan dari sistem kendali PID yang linear, biasanya sistem kendali PID dikombinasikan dengan sistem kecerdasan buatan yang mampu membantu *tuning* parameter PID secara *real-time*. Sistem kecerdasan buatan terbagi menjadi dua, yakni jaringan saraf tiruan (*artificial neural network*) dan logika *fuzzy*. Keduanya merupakan algoritma yang terbukti mampu mengatasi non-linearitas dan ketidakpastian parameter. Kelebihan dari logika *fuzzy* bila dibandingkan dengan *artificial neural network* adalah mampu merekam pengetahuan pakar dalam menentukan nilai parameter *gain* PID secara sederhana. Untuk itu, dalam tugas akhir ini akan dirancang pengendali PID yang diintegrasikan dengan logika *fuzzy* untuk membantu *tuning* parameter PID.

Sebagai solusi dari permasalahan tersebut, pada penelitian ini akan dirancang sistem kendali gerak berbasis *fuzzy*-PID untuk AUV tipe *double hull* dalam menjalankan skenario kontrol *waypoint following*. Selain itu, akan dibandingkan pula performansi dari sistem kendali yang diusulkan dengan sistem kendali PID biasa ketika ada gangguan dari arus laut.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka rumusan masalah pada penelitian ini adalah:

- a) Bagaimana merancang sistem pengendali gerak AUV tipe *double-hull* menggunakan *fuzzy*-PID?
- b) Bagaimana menentukan parameter terbaik dalam sistem pengendali gerak AUV tipe *double-hull* menggunakan *fuzzy*-PID?
- c) Bagaimana analisa performansi sistem kendali gerak AUV tipe *double-hull* menggunakan *fuzzy*-PID?

## 1.3 Tujuan

Tujuan dilakukannya penelitian ini adalah:

- a) Merancang sistem pengendali gerak AUV tipe *double-hull* menggunakan *fuzzy*-PID.
- b) Menentukan parameter terbaik dalam sistem pengendali gerak AUV tipe *double-hull* menggunakan *fuzzy*-PID.
- c) Menganalisa performansi sistem kendali gerak AUV tipe *double-hull* menggunakan *fuzzy*-PID.

## 1.4 Batasan Masalah

Batasan masalah pada penelitian ini adalah:

- a) Objek yang digunakan pada penelitian adalah *Autonomous Underwater Vehicle* (AUV) tipe *double-hull* yang sedang dikembangkan di CEiiA (Mendes, 2017).
- b) Perancangan sistem pengendali menggunakan *fuzzy*-PID.
- c) Metode *Fuzzy Inference System* (FIS) yang digunakan adalah logika *fuzzy* Mamdani tipe 1.
- d) Variabel yang dikendalikan adalah *surge*, *heave*, *pitch*, dan *yaw*.
- e) Gangguan (*disturbance*) yang dimodelkan pada penelitian ini adalah arus laut *irrotational* dalam bentuk 3D.

- f) Perancangan dan simulasi pada penelitian menggunakan *software* MATLAB 2016a dan Simulink.
- g) Dinamika sensor dan aktuator tidak dimodelkan dalam penelitian ini.

### **1.5 Sistematika Laporan**

Sistematika laporan Tugas Akhir terdiri atas tiga bagian besar, yaitu: Bagian Awal yang terdiri dari halaman judul, halaman pengesahan, halaman pernyataan orisinalitas, abstrak, kata pengantar, daftar isi, daftar gambar, daftar tabel, daftar simbol, dan daftar singkatan; Bagian Inti atau Pokok yang berisi pendahuluan, tinjauan pustaka, metodologi, hasil dan pembahasan, serta kesimpulan dan saran; dan Bagian Akhir yang terdiri dari daftar pustaka, lampiran, dan biodata penulis.

Bagian Inti atau Pokok terdiri atas lima bab utama, di mana Bab I Pendahuluan akan membahas latar belakang, rumusan masalah, tujuan, dan sistematika laporan. Bab II Tinjauan Pustaka akan membahas hasil penelitian terdahulu dan teori atau konsep dasar yang digunakan sebagai pedoman dalam mengerjakan Tugas Akhir ini. Bab III Metodologi akan membahas mengenai metode yang digunakan dan penjelasan tahap-tahap pelaksanaan penelitian secara berurutan. Bab IV Hasil dan Pembahasan akan mendiskusikan data yang diperoleh dari pelaksanaan metodologi serta menganalisa hasil tersebut. Bab V Kesimpulan dan Saran akan merangkum hasil penelitian yang menjawab permasalahan serta saran-saran terkait hal-hal yang masih dapat dikerjakan lebih baik dan dapat dikembangkan lebih lanjut, atau kendala yang dialami saat proses pengerjaan Tugas Akhir.

*Halaman ini sengaja dikosongkan*

## **BAB II**

### **TINJAUAN PUSTAKA DAN DASAR TEORI**

#### **2.1 Pengendalian AUV**

Penelitian untuk perancangan sistem kendali gerak pada AUV *over-actuated* yang *robust* dengan mempertimbangkan adanya gangguan arus laut dan ketidakpastian pemodelan (Vu, et al., 2021). Metode yang digunakan dalam jurnal ini adalah mengintegrasikan algoritma *Line of Sight* (LOS) sebagai sistem navigasi dengan sistem kontrol *dynamic sliding mode* (DSMC). Kemudian, untuk mengoptimalkan alokasi sinyal kontrol kepada *thruster*, dibandingkan performa antara dua algoritma berbeda, yakni *Least Square* (LS) dan *Quadratic Programming* (QP). Hasil simulasi yang diperoleh menunjukkan bahwa sistem dengan algoritma QP menghasilkan stabilitas yang lebih tinggi dan *steady-state error* yang lebih kecil.

Investigasi pengaruh dari gelombang laut dan arus laut terhadap kontrol gerak dari *underwater glider* dilakukan oleh Ullah, et al. (2019). Metode yang digunakan adalah mengintegrasikan algoritma *Line of Sight* (LOS) untuk merancang rute berbentuk gerigi untuk dilalui oleh *glider* dengan suatu sistem kontrol, kemudian memvariasikan parameter gangguan gelombang laut dan arus laut yang dimasukkan ke dalam pemodelan dinamis *glider*. Hasil yang diperoleh menunjukkan bahwa adanya gangguan arus laut sebesar 4 cm/s tidak terlalu mempengaruhi gerakan vertikal dari *glider*, namun mempegaruhi gerakan horizontal.

Perancangan sistem kendali untuk *trajectory following* dari sebuah AUV yang *fully actuated* telah berhasil dilakukan (Hammad, Elshenawy, & Singaby, 2017). Penelitian membagi sistem kontrol menjadi dua subsistem, yakni subsistem untuk kontrol posisi dan subsistem untuk kontrol kecepatan. *Inverse kinematic* digunakan untuk menentukan referensi kecepatan anguler masing-masing *thruster*, sedangkan logika *fuzzy* digunakan untuk membantu *tuning* parameter PID. Hasil yang diperoleh adalah sistem *fuzzy* PID yang dirancang untuk mengendalikan *fully actuated* AUV menunjukkan *overshoot* yang lebih rendah bila dibandingkan

dengan sistem kendali PID biasa, baik tanpa gangguan maupun dengan pengaruh gangguan.

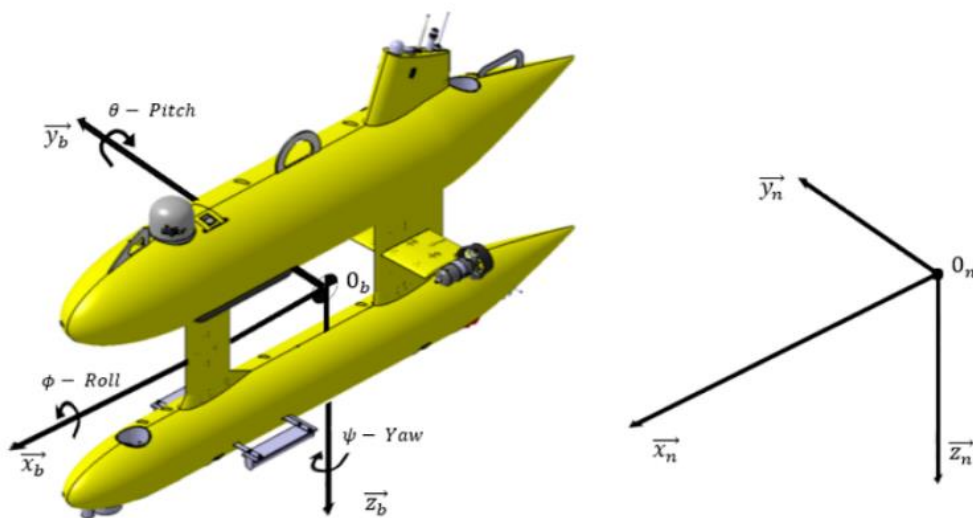
Desain kontroler untuk 3D *path following* AUV yang sederhana dan mudah diaplikasikan kendati ada parameter yang tidak pasti, telah dilakukan oleh Xiang, Yu, & Zhang (2017). Metode yang digunakan dalam jurnal ini adalah mendefinisikan 3D *path* yang akan dilalui sebagai rangkaian titik di mana pusat massa AUV harus berada, kemudian menggunakan 3D *guidance law* untuk menentukan kecepatan *surge*, sudut azimuth, dan sudut elevasi yang diperlukan untuk mencapai titik tersebut. Keluaran 3D *guidance laws* akan menjadi *input* bagi kontroler *fuzzy*-PID. Logika *fuzzy* berperan untuk melakukan *tuning* PID secara adaptif agar PID mampu menentukan gaya dan torsi yang diperlukan oleh aktuator supaya AUV bergerak menuju posisi yang diinginkan meskipun ada gangguan lingkungan yang bervariasi terhadap waktu (Xiang, Yu, & Zhang, *Robust Fuzzy 3D Path Following for Autonomous Underwater Vehicle Subject to Uncertainties*, 2017). Hasil yang diperoleh menunjukkan bahwa kontroler 3D *path following* yang diusulkan dapat berfungsi secara efektif, dibuktikan melalui data kuantitatif berupa perbandingan *Mean Square Error* (MSE) dan *Mean Average Error* (MAE) antara kontroler yang diusulkan dengan kontroler PID biasa dan *backstepping*.

*State of the art* implementasi logika *fuzzy* dalam *guidance* maupun kontrol dari AUV, dituliskan oleh Xiang, Yu, Lapierre, & Zhang (2017). Jurnal ini membandingkan antara kontrol logika *fuzzy* konvensional dengan *hybrid* (Xiang, Yu, Lapierre, & Zhang, 2017). Kontrol logika *fuzzy* konvensional biasanya menggunakan dua *input* berupa *error* dan *error rate* untuk menentukan tegangan yang disuplai ke aktuator. Hasil yang diperoleh menunjukkan bahwa logika *fuzzy* yang digabungkan dengan kontroler seperti PID atau SMC menunjukkan hasil yang lebih *robust* dalam kondisi yang dipengaruhi oleh gangguan. Hal ini disebabkan karena logika *fuzzy* mampu mengatasi sistem yang non-linear dan parameter yang tidak pasti, yang pada umumnya menyebabkan *tuning* kontroler seperti PID dan SMC jadi lebih menantang.

## 2.2 Pemodelan Dinamik *Autonomous Underwater Vehicle* (AUV)

Pemodelan dinamika suatu proses merupakan langkah awal dalam perancangan sistem kontrol. Dalam menyusun pemodelan dinamika dari *Autonomous Underwater Vehicle* (AUV) diperlukan pemahaman yang mendasar terkait bagaimana pergerakan AUV dideskripsikan.

Dalam menyusun pemodelan untuk kontrol gerak AUV, kita perlu mengetahui di mana posisi AUV tersebut berada, ke mana AUV saat ini menghadap (berorientasi), pada kecepatan berapa AUV tersebut bergerak, serta gaya atau momen yang menyebabkan AUV bergerak. Jika diperhatikan, besaran fisis di atas merupakan besaran-besaran vektor yang perlu dinyatakan dalam nilai dan arah. Arah merupakan hal yang relatif, seperti halnya sisi kiri dan sisi kanan berbeda bagi dua orang yang sedang berhadapan. Maka dari itu, diperlukan acuan sistem koordinat (*reference frame*) yang obyektif.



**Gambar 2.1** *Body-fixed Reference Frame* dan *NED Reference Frame* (Mendes, 2017)

Gambar 2.1 menunjukkan ada 2 *reference frame* yang umum digunakan dalam pemodelan AUV, yakni NED (*North-East-Down*) *reference {n} frame* dan *body-fixed reference {b} frame* (Yazdani, Sammut, Lammas, & Tang, 2016). Ada perbedaan titik *origin* (0,0,0) antara *{b} frame* dan *{n} frame*. Pada *{n} frame*, sumbu  $\vec{x}_n$  menunjuk arah utara; sumbu  $\vec{y}_n$  menunjuk arah timur; dan sumbu  $\vec{z}_n$  menunjuk arah pusat bumi.

Besaran vektor posisi, orientasi (yang dinyatakan dengan sudut Euler), kecepatan linear, kecepatan angular, gaya, dan momen dari AUV dinyatakan secara matematis dalam matriks vektor di bawah ini:

$$\begin{aligned} \boldsymbol{\eta} &= [\boldsymbol{\eta}_1^T, \boldsymbol{\eta}_2^T]^T; & \boldsymbol{\eta}_1 &= [x, y, z]^T; & \boldsymbol{\eta}_2 &= [\phi, \theta, \psi]^T & \text{dalam } \{n\} \text{ frame} \\ \boldsymbol{v} &= [\boldsymbol{v}_1^T, \boldsymbol{v}_2^T]^T; & \boldsymbol{v}_1 &= [u, v, w]^T; & \boldsymbol{v}_2 &= [p, q, r]^T & \text{dalam } \{b\} \text{ frame} \\ \boldsymbol{\tau} &= [\boldsymbol{\tau}_1^T, \boldsymbol{\tau}_2^T]^T; & \boldsymbol{\tau}_1 &= [X, Y, Z]^T; & \boldsymbol{\tau}_2 &= [K, M, N]^T & \text{dalam } \{b\} \text{ frame} \end{aligned} \quad (2.1)$$

di mana  $\boldsymbol{\eta}_1$  menunjukkan posisi;  $\boldsymbol{\eta}_2$  menunjukkan orientasi;  $\boldsymbol{v}_1$  menunjukkan kecepatan linier;  $\boldsymbol{v}_2$  menunjukkan kecepatan angular;  $\boldsymbol{\tau}_1$  menunjukkan gaya; dan  $\boldsymbol{\tau}_2$  menunjukkan momen.

**Tabel 2. 1** Notasi 6 DOF AUV (Hammad, Elshenawy, & Singaby, 2017)

DOF	Gerakan	Gaya dan momen	Kecepatan linier dan angular	Posisi dan sudut Euler
1	<i>Surge</i> (translasi sumbu $\vec{x}_b$ )	$X$	$u$	$x$
2	<i>Sway</i> (translasi sumbu $\vec{y}_b$ )	$Y$	$v$	$y$
3	<i>Heave</i> (translasi sumbu $\vec{z}_b$ )	$Z$	$w$	$z$
4	<i>Roll</i> (rotasi sumbu $\vec{x}_b$ )	$K$	$p$	$\phi$
5	<i>Pitch</i> (rotasi sumbu $\vec{y}_b$ )	$M$	$q$	$\theta$
6	<i>Yaw</i> (rotasi sumbu $\vec{z}_b$ )	$N$	$r$	$\psi$

Tabel 2.1 menunjukkan penjelasan dari notasi koordinat AUV. Posisi, kecepatan linier, dan gaya menunjukkan dinamika translasi dari AUV, sedangkan orientasi, kecepatan angular, dan momen menunjukkan dinamika rotasi dari AUV.



### 2.2.1 Persamaan Kinematik

Matriks kecepatan linear  $\mathbf{v}_1$  yang dinyatakan dalam  $\{b\}$  frame harus ditransformasi menjadi *position rate*  $\dot{\boldsymbol{\eta}}_1$  agar dapat dihubungkan dengan matriks posisi yang dinyatakan dalam  $\{n\}$  frame menggunakan persamaan berikut:

$$\dot{\boldsymbol{\eta}}_1 = \mathbf{J}_1(\boldsymbol{\eta}_2) \mathbf{v}_1 \quad (2.2)$$

Matriks  $\mathbf{J}_1(\boldsymbol{\eta}_2)$  merupakan matriks transformasi kecepatan linear yang dinyatakan sebagai berikut:

$$\mathbf{J}_1(\boldsymbol{\eta}_2) = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\phi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & s\phi s\theta s\psi + c\psi c\phi & -c\psi s\phi + s\theta s\psi c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \quad (2.3)$$

di mana  $s \cdot = \sin(\cdot)$  dan  $c \cdot = \cos(\cdot)$ .

Demikian pula dengan matriks kecepatan angular. Matriks kecepatan angular  $\mathbf{v}_2$  yang dinyatakan dalam  $\{b\}$  frame juga harus ditransformasi menjadi *euler rate*  $\dot{\boldsymbol{\eta}}_2$  agar dapat dihubungkan dengan matriks posisi yang dinyatakan dalam  $\{n\}$  frame menggunakan persamaan berikut:

$$\dot{\boldsymbol{\eta}}_2 = \mathbf{J}_2(\boldsymbol{\eta}_2) \mathbf{v}_2 \quad (2.4)$$

Matriks  $\mathbf{J}_2(\boldsymbol{\eta}_2)$  merupakan matriks transformasi kecepatan angular yang dinyatakan sebagai berikut:

$$\mathbf{J}_2(\boldsymbol{\eta}_2) = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta} \end{bmatrix} \quad (2.5)$$

di mana  $s \cdot = \sin(\cdot)$  ;  $c \cdot = \cos(\cdot)$  ; dan  $t \cdot = \tan(\cdot)$ . Berikut hasil matriks transformasi kecepatan linier dan angular yang sudah diperoleh:

$$\begin{bmatrix} \dot{\boldsymbol{\eta}}_1 \\ \dot{\boldsymbol{\eta}}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{J}_1(\boldsymbol{\eta}_2) & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{J}_2(\boldsymbol{\eta}_2) \end{bmatrix} \Rightarrow \dot{\boldsymbol{\eta}} = \mathbf{J}(\boldsymbol{\eta})\mathbf{v} \quad (2.6)$$

di mana  $\mathbf{0}_{3 \times 3}$  adalah matriks  $3 \times 3$  yang seluruh elemennya adalah 0.

### 2.2.2 Persamaan Kinetik

Persamaan kinetik menggambarkan gaya-gaya yang bekerja pada AUV sehingga menyebabkan AUV bergerak. Dengan mengaplikasikan hukum Newton, diperoleh persamaan kinetik AUV dalam *body-fixed frame*  $\{b\}$  sebagai berikut:

$$\mathbf{M}_{RB}\dot{\mathbf{v}} + \mathbf{C}_{RB}(\mathbf{v})\mathbf{v} = \boldsymbol{\tau}_{RB} \quad (2.7)$$

di mana  $\mathbf{M}_{RB}$  menunjukkan *rigid body mass matrix*, dan diperoleh menggunakan persamaan:

$$\mathbf{M}_{RB} = \begin{bmatrix} m & 0 & 0 & 0 & mz_g & -my_g \\ 0 & m & 0 & -mz_g & 0 & mx_g \\ 0 & 0 & m & my_g & -mx_g & 0 \\ 0 & -mz_g & my_g & I_x & -I_{xy} & -I_{xz} \\ mz_g & 0 & -mx_g & -I_{yx} & I_y & -I_{yz} \\ -my_g & mx_g & 0 & -I_{zx} & -I_{zy} & I_z \end{bmatrix} \quad (2.8)$$

dan  $\mathbf{C}_{RB}$  merupakan Coriolis dan *centripetal matrix* yang diperoleh menggunakan persamaan:

$$\mathbf{C}_{RB}(\mathbf{v}) \triangleq \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -m(y_g q + z_g r) & m(y_g p + w) & m(z_g p - v) \\ m(x_g q - w) & -m(z_g r + x_g p) & m(z_g q + u) \\ m(x_g r + v) & m(y_g r - u) & -m(x_g p + y_g q) \\ m(y_g q + z_g r) & -m(x_g q - w) & -m(x_g r + u) \\ -m(y_g p + w) & m(z_g r + x_g p) & -m(y_g r - u) \\ -m(z_g p - v) & -m(z_g q + u) & m(x_g p + y_g q) \\ 0 & -I_{yz}q - I_{xz}p + I_z r & I_{yz}r + I_{xy}p - I_y q \\ I_{yz}q - I_{xz}p - I_z r & 0 & -I_{xz}r - I_{xy}q + I_x p \\ -I_{yz}r - I_{xy}p + I_y q & I_{xz}r + I_{xy}q - I_x p & 0 \end{bmatrix} \quad (2.9)$$

sedangkan  $\boldsymbol{\tau}_{RB}$  merupakan vektor yang menunjukkan gaya dan momen eksternal yang bekerja pada AUV. Vektor  $\boldsymbol{\tau}_{RB}$  dapat diperoleh dengan persamaan:

$$\boldsymbol{\tau}_{RB} = \boldsymbol{\tau} + \boldsymbol{\tau}_A + \boldsymbol{\tau}_D + \boldsymbol{\tau}_R \quad (2.10)$$

$\tau$  —merupakan vektor dari gaya dan torsi yang timbul dari permukaan atau *thruster*, diperoleh menggunakan persamaan:

$$\tau = L \cdot U \quad (2.11)$$

di mana  $L$  merupakan *mapping matrix* dan  $U$  merupakan gaya dari masing-masing *thruster*.

$\tau_A$  —merupakan vektor dari gaya hidrodinamis akibat *added mass* dari volume fluida yang ikut berpindah saat badan AUV bergerak dalam fluida, dan diperoleh menggunakan persamaan:

$$\tau_A = -M_A \dot{v} - C_A(v)v \quad (2.12)$$

$M_A$  merupakan *added mass matrix* yang diperoleh dengan persamaan:

$$M_A = - \begin{bmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{w}} & X_{\dot{p}} & X_{\dot{q}} & X_{\dot{r}} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{w}} & Y_{\dot{p}} & Y_{\dot{q}} & Y_{\dot{r}} \\ Z_{\dot{u}} & Z_{\dot{v}} & Z_{\dot{w}} & Z_{\dot{p}} & Z_{\dot{q}} & Z_{\dot{r}} \\ K_{\dot{u}} & K_{\dot{v}} & K_{\dot{w}} & K_{\dot{p}} & K_{\dot{q}} & K_{\dot{r}} \\ M_{\dot{u}} & M_{\dot{v}} & M_{\dot{w}} & M_{\dot{p}} & M_{\dot{q}} & M_{\dot{r}} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{w}} & N_{\dot{p}} & N_{\dot{q}} & N_{\dot{r}} \end{bmatrix} \quad (2.13)$$

di mana  $X_{\dot{u}}$  menunjukkan gaya hidrodinamis akibat *added mass*  $X$  dengan percepatan  $\dot{u}$  di arah sumbu  $x$ .

Untuk matriks gaya Coriolis dan sentripetal akibat *added mass*  $C_A$  diperoleh dengan menggunakan persamaan:

$$C_A(v) = \begin{bmatrix} 0 & 0 & 0 & 0 & -a_3 & a_2 \\ 0 & 0 & 0 & a_3 & 0 & -a_1 \\ 0 & 0 & 0 & -a_2 & a_1 & 0 \\ 0 & -a_3 & a_2 & 0 & -b_3 & b_2 \\ a_3 & 0 & -a_1 & b_3 & 0 & -b_1 \\ -a_2 & a_1 & 0 & -b_2 & b_1 & 0 \end{bmatrix} \quad (2.14)$$

di mana

$$a_1 = X_{\dot{u}}u + X_{\dot{v}}v + X_{\dot{w}}w + X_{\dot{p}}p + X_{\dot{q}}q + X_{\dot{r}}r$$

$$a_2 = Y_{\dot{u}}u + Y_{\dot{v}}v + Y_{\dot{w}}w + Y_{\dot{p}}p + Y_{\dot{q}}q + Y_{\dot{r}}r$$

$$\begin{aligned}
a_3 &= Z_{\dot{u}}u + Z_{\dot{v}}v + Z_{\dot{w}}w + Z_{\dot{p}}p + Z_{\dot{q}}q + Z_{\dot{r}}r \\
b_1 &= K_{\dot{u}}u + K_{\dot{v}}v + K_{\dot{w}}w + K_{\dot{p}}p + K_{\dot{q}}q + K_{\dot{r}}r \\
b_2 &= M_{\dot{u}}u + M_{\dot{v}}v + M_{\dot{w}}w + M_{\dot{p}}p + M_{\dot{q}}q + M_{\dot{r}}r \\
b_3 &= N_{\dot{u}}u + N_{\dot{v}}v + N_{\dot{w}}w + N_{\dot{p}}p + N_{\dot{q}}q + N_{\dot{r}}r
\end{aligned} \tag{2.15}$$

$\tau_D$  —merupakan vektor gaya akibat *lift*, *drag*, gesekan kulit AUV dengan fluida, dan lainnya yang didefinisikan sebagai

$$\tau_D = -D(v)v \tag{2.16}$$

dengan  $D(v)$  sebagai *hydrodynamic damping matrix*. Jika AUV bergerak lambat, maka pendekatan kuadratik dapat digunakan untuk merumuskan:

$$D(v) = D_l + D_q(v) \tag{2.17}$$

di mana

$$D_l = \begin{bmatrix} X_u & X_v & X_w & X_p & X_q & X_r \\ Y_u & Y_v & Y_w & Y_p & Y_q & Y_r \\ Z_u & Z_v & Z_w & Z_p & Z_q & Z_r \\ K_u & K_v & K_w & K_p & K_q & K_r \\ M_u & M_v & M_w & M_p & M_q & M_r \\ N_u & N_v & N_w & N_p & N_q & N_r \end{bmatrix} \tag{2.18}$$

dan

$$D_q(v) = \begin{bmatrix} X_{u|u}|u| & X_{v|v}|v| & X_{w|w}|w| & X_{p|p}|p| & X_{q|q}|q| & X_{r|r}|r| \\ Y_{u|u}|u| & Y_{v|v}|v| & Y_{w|w}|w| & Y_{p|p}|p| & Y_{q|q}|q| & Y_{r|r}|r| \\ Z_{u|u}|u| & Z_{v|v}|v| & Z_{w|w}|w| & Z_{p|p}|p| & Z_{q|q}|q| & Z_{r|r}|r| \\ K_{u|u}|u| & K_{v|v}|v| & K_{w|w}|w| & K_{p|p}|p| & K_{q|q}|q| & K_{r|r}|r| \\ M_{u|u}|u| & M_{v|v}|v| & M_{w|w}|w| & M_{p|p}|p| & M_{q|q}|q| & M_{r|r}|r| \\ N_{u|u}|u| & N_{v|v}|v| & N_{w|w}|w| & N_{p|p}|p| & N_{q|q}|q| & N_{r|r}|r| \end{bmatrix} \tag{2.19}$$

$\tau_R$  —merupakan gaya pemulih hidrostatis akibat gaya gravitasi  $f_g$  yang arahnya ke bawah dari titik berat atau *center of gravity* (CG)  $r_g = [x_g \ y_g \ z_g]^T$ , dan gaya apung  $f_b$  yang arahnya ke atas dari titik pusat gaya apung atau *center of buoyancy* (CB)

$r_b = [x_b \ y_b \ z_b]^T$  ; keduanya dalam  $\{b\}$  frame. Letak CG dan CB tergantung geometri AUV.

Secara singkat,  $\tau_R$  dinyatakan sebagai:

$$\tau_R = -g(\eta) \quad (2.20)$$

di mana

$$g(\eta) = \begin{bmatrix} (W - B) s\theta \\ -(W - B) c\theta s\phi \\ -(W - B) c\theta c\phi \\ -(y_g W - y_b B) c\theta c\phi + (z_g W - z_b B) c\theta s\phi \\ (z_g W - z_b B) s\theta - (x_g W - x_b B) c\theta c\phi \\ -(x_g W - x_b B) c\theta s\phi + (y_g W - y_b B) s\theta \end{bmatrix} \quad (2.21)$$

di mana  $s \cdot = \sin(\cdot)$  dan  $c \cdot = \cos(\cdot)$ .

### 2.2.3 Pemodelan Dinamika AUV yang Lengkap

Pemodelan dinamik menggabungkan antara persamaan kinematik dan persamaan kinetik. Pemodelan dinamik AUV secara lengkap dapat ditulis ulang sebagai:

$$M\dot{v} + C(v)v + D(v)v + g(\eta) = \tau \quad (2.22)$$

$$\dot{\eta} = J(\eta)v \quad (2.23)$$

di mana  $M = M_{rb} + M_A$  dan  $C = C_{rb} + C_A$ .

Pemodelan dinamik di atas dapat disusun ulang dalam bentuk  $\dot{x} = f(x, t)$  sebagai berikut:

$$\dot{v} = M^{-1}[-C(v)v - D(v)v - g(\eta) + \tau] \quad (2.24)$$

## 2.3 Gangguan Arus Laut

Arus laut menyerupai sistem air laut yang bersirkulasi baik secara horizontal maupun vertikal akibat adanya gravitasi, gesekan dengan angin, dan variasi massa jenis air karena perbedaan salinitas pada area yang berbeda-beda di laut (Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*, 2011).

Jika adanya gangguan arus laut (secara 3 dimensional) dipertimbangkan, maka persamaan dinamik dari AUV akan berubah menjadi persamaan berikut:

$$M_{RB}\dot{v} + C_{RB}(v)v + g(\eta) + g_0 + M_A\dot{v}_r + C_A(v_r)v_r + D(v_r)v_r = \tau \quad (2.25)$$

di mana

$$v_r = v - v_c^B \quad (2.26)$$

dengan  $v_r$  adalah kecepatan relatif AUV terhadap kecepatan arus laut, dan

$$v_c^E = [v_x \quad v_y \quad v_z \quad 0 \quad 0 \quad 0]^T \quad (2.27)$$

dengan  $v_c^E$  adalah matriks kecepatan arus laut dalam *earth-frame*, dan

$$v_x = V_c \cos \alpha_c \cos \beta_c \quad (2.28)$$

$$v_y = V_c \sin \beta_c \quad (2.29)$$

$$v_z = V_c \sin \alpha_c \cos \beta_c \quad (2.30)$$

di mana  $\alpha_c$  adalah *angle of attack*,  $\beta_c$  adalah *sideslip angle*, dan  $V_c$  adalah kecepatan arus laut yang ditentukan dengan persamaan berikut (Fossen, *How to Incorporate Wind, Waves, and Ocean Currents in the Marine Craft Equations of Motion*, 2012):

$$\dot{V}_c(t) + \mu_0 V_c(t) = \omega(t) \quad (2.31)$$

Dengan menggunakan deret Euler, persamaan 2.31 diubah menjadi:

$$V_c(t) = \frac{V_c(0)}{0!} + \frac{t}{1!} \cdot \frac{\partial V_c}{\partial t} \quad (2.32)$$

dan diperoleh besar  $V_c(t)$  sebagai berikut:

$$V_c(t) = V_c(0) + t(\omega(t) - \mu_0 V_c(0)) \quad (2.33)$$

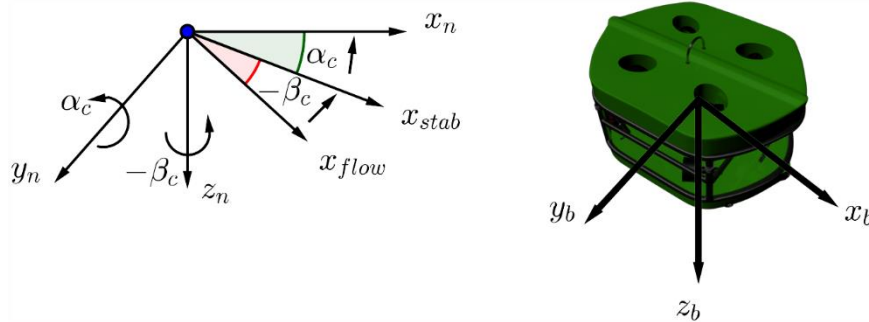
di mana  $V_c(t)$  biasanya dipasangkan dengan elemen saturasi untuk menjaga kecepatan arus laut tetap pada *range*  $V_{min} \leq V_c(t) \leq V_{max}$ ;  $\omega(t)$  adalah *zero-mean Gaussian white noise*; dan  $\mu_0$  adalah konstanta bernilai  $\geq 0$ , namun pada umumnya  $\mu_0 = 0$  (Fossen, *Guidance and Control of Ocean Vehicles*, 1994).

Persamaan di atas masih dalam *earth-frame*, maka untuk digabungkan dengan pemodelan dinamik AUV harus ditransformasi ke dalam  $\{b\}$  *frame* dengan persamaan:

$$v_c^B = \begin{bmatrix} u_c \\ v_c \\ w_c \end{bmatrix} = J_1(\eta_2)^T v_c^E \quad (2.34)$$

di mana nilai  $[u_c \ v_c \ w_c]^T = v_c^B$  dan dapat disubstitusikan ke persamaan 2.26 untuk memperoleh nilai  $v_r$ .

Arah *side slip angle* dan *angle of attack* dapat ditentukan dengan melihat Gambar 2.2 berikut ini:



**Gambar 2. 2** Pengukuran *Angle of Attack* dan *Side Slip Angle* pada AUV  
(Hammad, Elshenawy, & Singaby, 2017)

## 2.4 Guidance dan Kontrol

Sistem autopilot dalam AUV sejatinya merupakan gabungan antara sistem *guidance*, sistem kendali gerak, dan sistem navigasi. Sistem kendali gerak memerlukan data referensi trayektori yang dihasilkan oleh sistem *guidance*, maka dari itu dalam mengembangkan sistem kendali gerak, masalah *guidance* perlu dibahas.

Menurut Fossen (1994), *guidance* merupakan aksi menentukan ke mana AUV harus belok, dengan sudut berapa, kecepatan berapa, dan jalur mana yang sebaiknya dilalui, relatif terhadap *reference frame* tertentu. *Guidance* menentukan trayektori atau rute terbaik untuk diikuti oleh AUV berdasarkan lokasi target dan/atau kapasitas AUV. *Guidance* biasanya dapat dicapai menggunakan *Line of Sight* (LOS) untuk menentukan sudut yang harus ditambahkan dengan

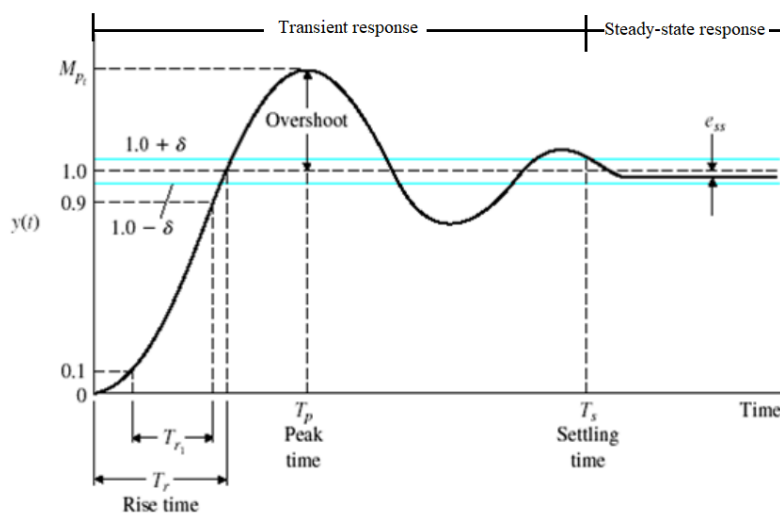
mempertimbangkan posisi AUV saat ini dan *waypoint* berikutnya. Sistem kontrol atau kendali gerak dalam AUV bertugas menghitung berapa gaya dan momen yang diperlukan untuk mencapai tujuan pengendalian, hingga menentukan berapa gaya yang harus diberikan oleh masing-masing aktuator dalam AUV.

## 2.5 Analisa Performansi Sistem Pengendali

Ada dua metode dalam menganalisa performansi suatu sistem pengendalian, yakni melalui domain waktu dan melalui domain frekuensi.

### 2.5.1 Respons Waktu

Keluaran suatu sistem dinamik  $y(t)$  akan berubah terhadap waktu. Fungsi yang menggambarkan perubahan keluaran suatu sistem terhadap waktu disebut respons waktu dari sistem kontrol yang dianalisa.



**Gambar 2. 3** Respons Waktu (Jennings, 2016)

Gambar 2.3 menunjukkan bahwa respons waktu terdiri atas dua bagian, yakni respons transien dan respons *steady-state*. Respons transien menggambarkan proses suatu variabel keluaran berubah dari kondisi awal ke kondisi *steady* ketika diberi suatu sinyal masukan, sedangkan respons *steady state* adalah nilai keluaran yang didapatkan seiring waktu mendekati tak terhingga ketika sistem diberi suatu sinyal masukan.

Respons transien dapat dianalisa menggunakan 5 parameter, yakni *delay time* ( $T_d$ ), *rise time* ( $T_r$ ), *peak time* ( $T_p$ ), *maximum overshoot* ( $M_p$ ), dan *settling time*



( $T_s$ ), sedangkan respons *steady-state* dapat dinilai melalui *error steady-state* (Bazoune, n.d.). Berikut merupakan penjelasan dari masing-masing parameter tersebut:

- a. *Delay time* ( $T_d$ ) adalah waktu yang diperlukan oleh respons untuk mencapai setengah dari nilai akhir atau masukan atau *setpoint* yang diinginkan (Bazoune, n.d.).
- b. *Rise time* ( $T_r$ ) adalah waktu yang diperlukan oleh respons untuk meningkat dari 10% ke 90% dari jarak kondisi inisial menuju nilai masukan atau *setpoint* yang diinginkan (The MathWorks, Inc., 2022).
- c. *Peak time* ( $T_p$ ) adalah waktu saat nilai tertinggi dari respons terjadi (The MathWorks, Inc., 2022).
- d. *Maximum overshoot* ( $M_p$ ) adalah persentase *overshoot* ( $OS$ ) yang dihitung menggunakan persamaan:

$$\%OS = \max \left( \frac{y(t) - y_{initial}}{y_{final} - y_{initial}} \right) \times 100\% \quad (2.35)$$

di mana  $y_{initial}$  adalah nilai keluaran pada kondisi awal, umumnya ketika  $t = 0$ ,  $y(t)$  adalah nilai keluaran pada waktu  $t$ , dan  $y_{final}$  adalah nilai akhir dari keluaran (The MathWorks, Inc., 2022).

- e. *Settling time* ( $T_s$ ) adalah waktu di mana selisih antara  $y(t)$  dengan  $y_{final}$  berada di dalam batas yang dapat ditolerir, biasanya 2% dari selisih antara  $y_{initial}$  dengan  $y_{final}$  (The MathWorks, Inc., 2022).
- f. *Error steady-state* ( $e_{ss}$ ) adalah selisih antara nilai keluaran dengan nilai masukan atau *setpoint* yang diinginkan saat respon sistem sudah *steady* (Bazoune, n.d.). Nilai *error steady-state* ditentukan menggunakan persamaan:

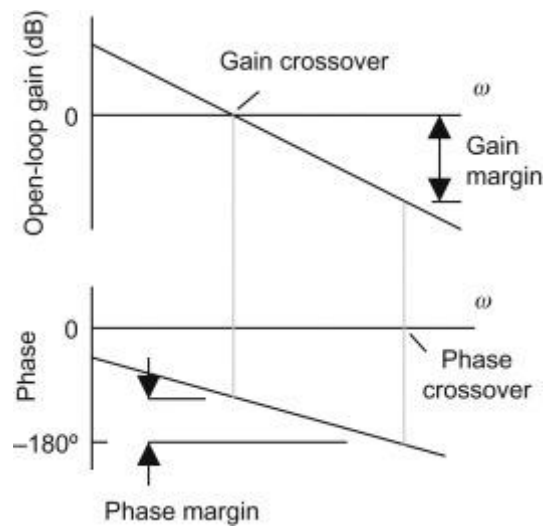
$$e_{ss} = \lim_{t \rightarrow \infty} e(t) = \lim_{s \rightarrow 0} sE(s) \quad (2.36)$$

di mana  $e(t)$  adalah nilai *error* pada waktu  $t$ , sedangkan  $E(s)$  adalah bentuk Laplace dari  $e(t)$ .

### 2.5.2 Respons Frekuensi

Respons frekuensi memberikan gambar kualitatif terhadap respon transien suatu sistem kontrol. Dalam merancang sistem kontrol *closed-loop* (sistem kontrol dengan *feedback*), respons frekuensi dari sistem kontrol *open-loop* (tanpa *feedback*) disesuaikan dengan kriteria stabilitas demi mencapai respons transien yang diinginkan (Ogata, 2002).

Salah satu cara untuk menganalisa respons frekuensi adalah dengan menggunakan diagram Bode. Diagram Bode biasa digunakan untuk menentukan stabilitas sistem kontrol. Diagram Bode terdiri atas dua grafik, yakni diagram magnitude Bode dan diagram fase Bode. Dalam menganalisa diagram Bode, ada dua istilah yang perlu dipahami, yakni *gain margin* dan *phase margin*.



**Gambar 2. 4** Diagram Bode (Bolton W. , 2004)

Gambar 2.4 menunjukkan perbedaan kurva magnitude, kurva fase, *gain crossover frequency*, *phase crossover frequency*, *gain margin*, dan *phase margin* pada diagram Bode.

*Gain margin* adalah nilai *gain* yang dapat ditambahkan atau dikurangkan tanpa membuat sistem menjadi tidak stabil. *Gain margin* dinyatakan sebagai magnitude dengan satuan dB. Nilai *gain margin* dapat ditentukan secara langsung dari diagram Bode, yakni dengan menghitung jarak vertikal antara kurva magnitude dengan sumbu-x menunjukkan frekuensi di mana diagram fase Bode =  $180^\circ$ , atau

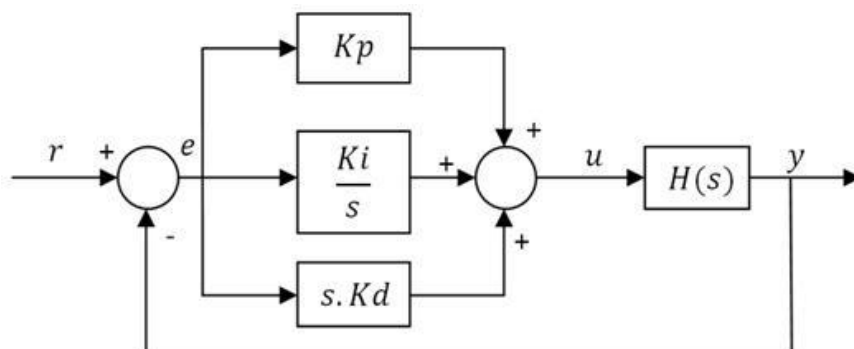
disebut *phase crossover frequency*. Semakin besar nilai *gain margin* maka akan semakin stabil suatu sistem.

*Phase margin* adalah nilai fase yang bisa ditambahkan atau dikurangkan tanpa membuat sistem menjadi tidak stabil. *Phase margin* dinyatakan sebagai fase dalam satuan derajat. *Phase margin* dapat ditentukan secara langsung dari diagram Bode, yakni dengan menghitung jarak vertikal antara kurva fase pada diagram fase Bode dengan sumbu x pada frekuensi di mana magnitude diagram Bode = 0 dB, atau disebut sebagai *gain crossover frequency*. Semakin besar nilai *phase margin* maka akan semakin stabil suatu sistem (Electrical4U, 2021).

Salah satu syarat lainnya dalam menentukan kestabilan suatu sistem dengan menggunakan diagram Bode adalah dengan menganalisa: sistem *closed-loop* dapat dikatakan stabil apabila sistem *open-loop*-nya stabil (Hahn, Edison, & Edgar, 2001).

## 2.6 Pengendali PID

Pengendali PID merupakan sistem pengendalian yang populer karena mudah diaplikasikan dalam berbagai *plant*, bahkan ketika model matematis dari *plant* tidak diketahui dengan jelas. Pengendali PID memiliki tiga fungsi utama, yaitu *proportional*, *integral*, dan *derivative*. *Proportional* linear terhadap *error* yang dialami oleh sistem pada suatu waktu, *integral* mengakumulasi *error* selama jangka waktu tertentu, sedangkan *derivative* mengukur seberapa cepat *error* terjadi. Gambar 2.5 menunjukkan skema pengendali PID.



**Gambar 2. 5** Skema Rangkaian PID (WangReady, 2012)

Pengendali PID bekerja dengan meminimalkan nilai kesalahan setiap waktu dengan penyetelan variabel kontrol, ke nilai baru yang ditentukan dengan persamaan berikut:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt} \quad (2.37)$$

$$u(t) = K_p \left( e(t) + \frac{1}{T_i} \int_0^t e(t)dt + T_d \frac{de(t)}{dt} \right) \quad (2.38)$$

di mana  $K_p$ ,  $K_i$ , dan  $K_d$  masing-masing secara berurutan adalah *gain* proporsional, integral, dan derivatif;  $e(t)$  adalah *error*;  $u(t)$  adalah variabel yang dimanipulasi;  $T_i$  adalah *integral time*, dan  $T_d$  adalah *derivative time*.

*Error* diperoleh menggunakan persamaan berikut:

$$e(t) = \eta_d(t) - \eta(t) \quad (2.39)$$

di mana  $\eta_d(t)$  merupakan trayektori yang menjadi acuan dan  $\eta(t)$  adalah trayektori aktual (Gan, Zhu, Xu, & Sun, 2017).

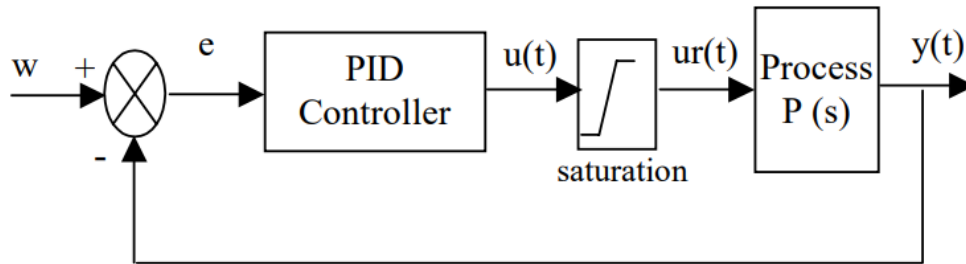
Persamaan 2.33 dapat diubah ke dalam bentuk fungsi transfer sebagai berikut:

$$\frac{U(s)}{E(s)} = K_p \left( 1 + \frac{1}{T_i s} + T_d s \right) \quad (2.40)$$

Pada pengendali PID konvensional, nilai *gain* proporsional, integral dan derivatif sudah ditetapkan sejak awal sehingga kurang mampu mengkompensasi gangguan yang berubah terhadap waktu. Untuk itu, pengendali PID perlu dikombinasikan dengan kecerdasan buatan agar mampu mengubah nilai *gain* proporsional, integral, dan derivatif dengan sesuai. Menambah nilai *gain* dari *proportional* akan meningkatkan responsivitas sistem, namun dapat menyebabkan sistem menjadi tidak stabil dan tidak dapat menghilangkan *error steady state* ( $e_{ss}$ ). Mengkombinasikan *proportional* dengan *integral* dapat mengeliminasi  $e_{ss}$ , namun dapat menimbulkan *overshoot*. Mengkombinasikan keduanya dengan *derivative* dapat menambah *damping* sekaligus mengurangi *overshoot* (Mendes, 2017).

## 2.7 Anti-windup

Salah satu masalah yang kerap terjadi dalam perancangan kontroler linear seperti PID untuk *plant* atau proses yang bersifat non-linear adalah fenomena *windup*. Fenomena *windup* biasanya terjadi ketika aktuator memiliki batasan atau limitasi pada keluarannya.



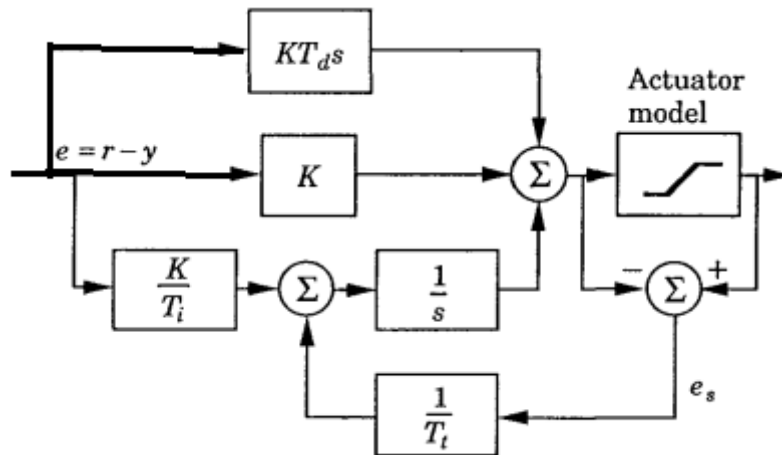
**Gambar 2. 6** Skema Rangkaian *Closed-Loop* dengan Aktuator Bersaturasi  
(Alaoui, Ayad, & Doubabi, 2006)

Gambar 2.6 merupakan diagram blok sistem kontroler dengan aktuator yang memiliki batasan keluaran, di mana  $w$  adalah nilai *setpoint*,  $e$  adalah selisih antara keluaran  $y(t)$  dengan *setpoint*  $w$ ,  $u(t)$  adalah sinyal kontrol, dan  $ur(t)$  adalah keluaran aktuator. Ketika terjadi lonjakan nilai  $w$ , maka sinyal kontrol akan menjadi sangat besar. Jika nilainya melebihi batas maksimum  $ur(t)$ , maka keluaran aktuator akan stagnan di batas maksimumnya, sehingga nilai  $ur(t)$  kurang dari  $u(t)$ . Akibatnya, nilai integrator akan terus bertambah dan dapat menimbulkan *overshoot* yang tidak terkendali, sedangkan keluaran aktuator akan selalu stagnan dan respon  $y(t)$  lebih lambat dari yang diharapkan (Alaoui, Ayad, & Doubabi, 2006).

Untuk mengatasi fenomena *windup* ini, ada dua cara yang umum digunakan, yakni: *clamping* dan *back-calculation*. *Clamping* membuat integrator bekerja secara kondisional sehingga mencegah akumulasi sinyal kontrol ketika aktuator tersaturasi, sedangkan *back-calculation* menghitung selisih antara sinyal kontrol dengan keluaran aktuator, kemudian mengalikan hasilnya dengan nilai 1 dibagi suatu konstanta yang disebut *time tracking constant*  $T_t$ , dan menjumlahkan hasilnya

dengan nilai integrator sehingga respon integral dari kontroler tetap terkendali, stabil, dan tidak mengakumulasi.

Skema *back-calculation* untuk mencegah fenomena *windup* dapat dilihat pada Gambar 2.7 berikut:



**Gambar 2.7** Skema *Anti-Windup Back-Calculation* (Åström & Wittenmark, *Computer-Controlled Systems*, 2011)

Untuk kontroler PID, nilai  $T_t$  umumnya dihitung dengan persamaan berikut (Åström & Hägglund, *Advanced PID Control*, 2005):

$$T_t = \sqrt{T_i/T_d} \quad (2.41)$$

sedangkan untuk kontroler PI dapat menggunakan persamaan (Astrom & Rundqwist, 1989):

$$T_t = 0.8T_i \quad (2.42)$$

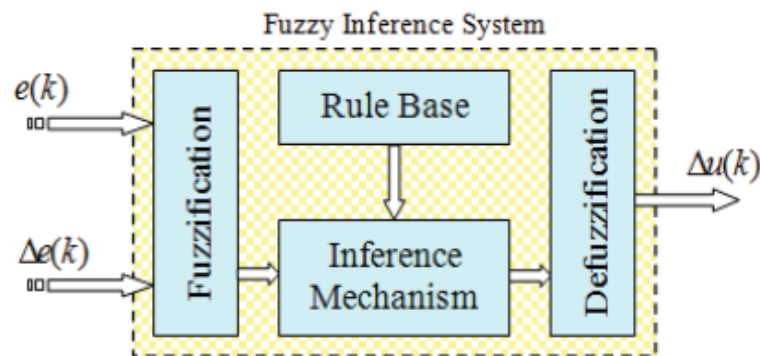
di mana  $T_i$  adalah periode integral dan  $T_d$  adalah periode derivatif.

## 2.8 Logika Fuzzy

Teori *fuzzy* pertama kali dikemukakan oleh Lotfi A. Zadeh pada tahun 1965. Istilah *fuzzy* mengacu pada hal-hal yang tidak jelas atau kabur. Perbedaan antara logika tegas (*crisp*) dan logika *fuzzy* terletak pada keanggotaan elemen dalam suatu himpunan. Jika dalam logika tegas suatu elemen mempunyai dua pilihan yaitu terdapat dalam himpunan atau bernilai 1 yang berarti benar dan tidak pada

himpunan atau bernilai 0 yang berarti salah, dalam logika *fuzzy* keanggotaan elemen dapat berada di interval  $[0, 1]$ .

Perancangan sistem pengambilan keputusan menggunakan logika *fuzzy*, yang dilakukan pertama kali adalah menentukan variabel *input* dan *output* dari logika *fuzzy*. Variabel *input* dan *output* akan digunakan untuk menentukan fuzzifikasi, *rule base*, dan defuzzifikasi.



**Gambar 2. 8** Arsitektur Logika *Fuzzy* (Usta, Akyazi, & Altaş, 2011)

Gambar 2.8 menunjukkan terdapat empat elemen utama dalam menyusun sistem logika *fuzzy*, yakni *fuzzification*, *rule base*, *inference mechanism*, dan *defuzzification*.

1) *Fuzzification* atau fuzzifikasi

Fuzzifikasi adalah suatu proses perubahan nilai tegas/*real* ke dalam fungsi keanggotaan *fuzzy* dengan menentukan himpunan logika *fuzzy* terlebih dahulu. Himpunan *fuzzy* memiliki dua atribut, yaitu bahasa (linguistik) untuk penamaan grup yang mewakili kondisi tertentu, dan angka (numerik) yang menunjukkan ukuran dari suatu variabel. Langkah berikutnya adalah menentukan semesta pembicaraan dan domain himpunan *fuzzy*. Kemudian, dapat ditentukan fungsi keanggotaan dari masing-masing himpunan *fuzzy*. Fungsi keanggotaan adalah suatu kurva yang menunjukkan pemetaan titik-titik *input* ke dalam nilai keanggotaannya atau derajat keanggotaannya yang memiliki interval antara 0 sampai 1. Beberapa fungsi atau kurva yang dapat digunakan untuk mendapatkan

nilai keanggotaan antara lain: kurva segitiga, kurva S, kurva *phi*, kurva trapezium, dan kurva Gaussian.

2) *Rule base*

Setelah menentukan fungsi keanggotaan, langkah berikutnya adalah menentukan *rule base*. Secara sederhana, *rule-base* merupakan suatu bentuk aturan relasi/implikasi “IF THEN”.

3) *Inference Mechanism*

*Inference mechanism* adalah proses implikasi dalam menalar nilai *input* untuk menentukan nilai *output* sebagai pengambilan keputusan. Ada tiga metode yang umum digunakan dalam *inference mechanism*, yaitu metode Mamdani, metode Takagi-Sugeno, dan metode Tsukamoto.

4) *Defuzzification* atau defuzzifikasi

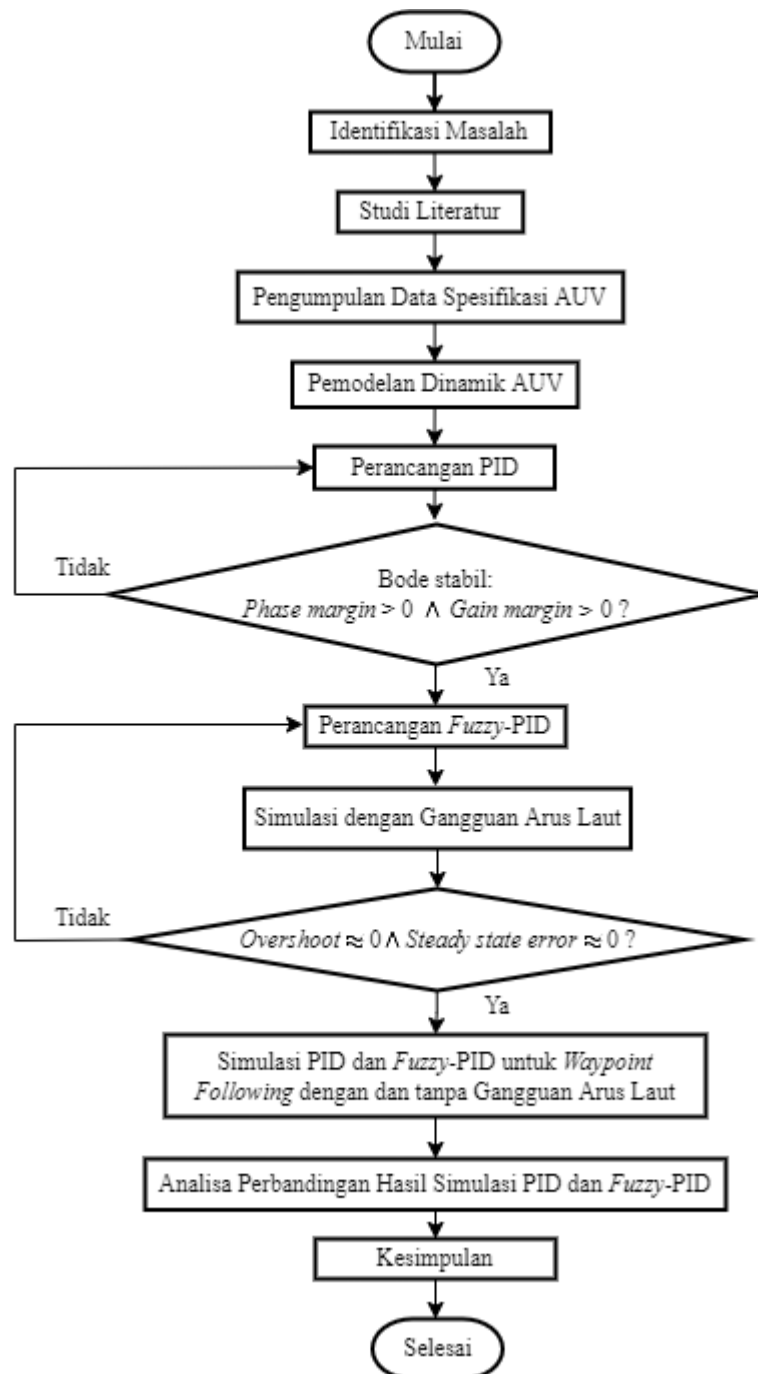
*Defuzzification* adalah proses pemetaan dari himpunan *fuzzy* ke himpunan tegas. Domain *crisp* sering disebut sebagai *defuzzification*.



### BAB III

## METODOLOGI PENELITIAN

Tahapan penelitian yang digunakan dalam penyusunan Proposal Tugas Akhir ini disusun sesuai diagram alir yang ditunjukkan pada Gambar 3.1 berikut:



**Gambar 3. 1** Diagram Alir Metodologi Penelitian

### 3.1 Identifikasi Masalah

AUV merupakan sistem yang rumit karena memiliki 6 derajat kebebasan, non-linear, dan banyak dipengaruhi oleh ketidakpastian baik itu dalam parameter maupun gangguan eksternal dari lingkungan, salah satunya gangguan dari arus laut yang sangat kuat pengaruhnya ketika AUV berada di laut dalam. Sistem autopilot AUV terdiri atas sistem *guidance* dan sistem pengendali. Untuk sistem *guidance* yang diteliti adalah *waypoint following*. Terkait sistem pengendali, pengendali PID yang umumnya dipakai perlu dimodifikasi supaya mampu beradaptasi terhadap lingkungan yang berubah terhadap waktu dan tetap beroperasi secara stabil. Untuk itu, perlu disusun suatu sistem kendali gerak AUV yang mampu beradaptasi terhadap gangguan arus laut dan tetap stabil.

### 3.2 Studi Literatur

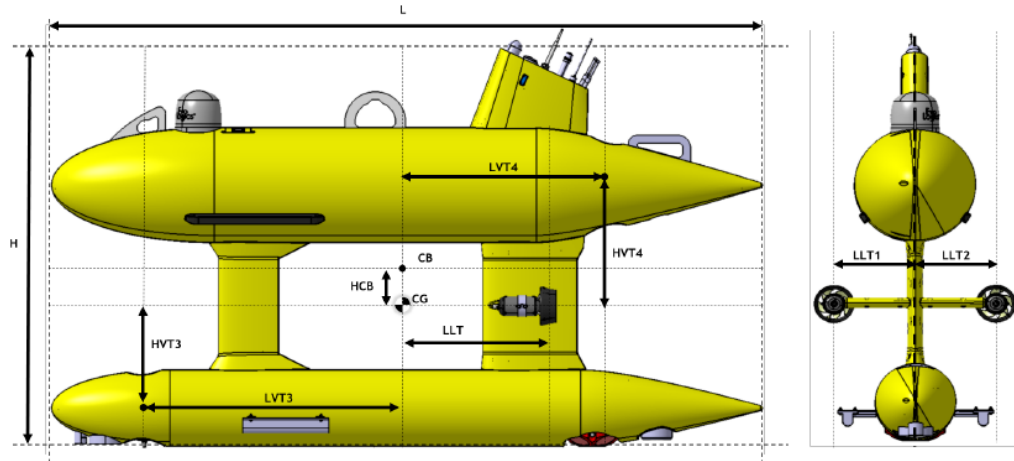
Untuk dapat merancang sistem kendali AUV diperlukan pemahaman yang cukup mengenai dinamika dan kinematika AUV. Dengan memahami pergerakan AUV maka dapat ditentukan pemodelan matematis yang sesuai. Selain itu, studi literatur juga dilakukan untuk mencari tahu strategi kontrol apa saja yang sudah pernah dikembangkan dan bagaimana performa yang dihasilkan. Setelah mengetahui strategi kontrol yang ingin dirancang, studi literatur terhadap strategi kontrol tersebut perlu dilakukan. Dalam proposal ini, strategi kontrol yang dimaksud adalah PID dan logika *fuzzy* untuk membantu *tuning* parameter PID.

### 3.3 Pengumpulan Data Spesifikasi AUV

AUV yang digunakan dalam penelitian ini adalah AUV yang dikembangkan oleh *Centre of Engineering and Product Development* di (CEiiA). AUV tersebut dirancang untuk mengumpulkan data untuk pemetaan kondisi di dasar laut dan memiliki kapasitas menyelam hingga kedalaman 3000 meter.

Penelitian sebelumnya terkait AUV di CEiiA dilakukan oleh dua orang peneliti, di mana peneliti pertama yakni Bentes (2016) menurunkan pemodelan dinamik untuk AUV di CEiiA yang memiliki dua lambung, sedangkan peneliti kedua yakni Mendes (2017) merancang sistem pengendali gerak *surge*, *heave*, *yaw*,

dan *pitch* untuk AUV di CEiiA dengan menggunakan kontroler PID dan LQR, namun belum meneliti respon sistem yang dirancang di bawah gangguan arus laut.



**Gambar 3. 2** AUV Tipe *Double-Hull* yang Dikembangkan di CEiiA (Mendes, 2017)

Gambar 3.2 memperlihatkan bahwa AUV yang dikembangkan di CEiiA memiliki dua lambung kapal atau *hull*, di mana lambung bagian bawah menyimpan baterai dan lambung bagian atas menyimpan sensor dan sistem lain yang diperlukan untuk navigasi (Bentes, 2016). Data ukuran dari variabel pada Gambar 3.2 dapat dilihat pada tabel berikut:

**Tabel 3. 1** Dimensi AUV Tipe *Double-Hull* di CEiiA dari Notasi Gambar 3.2

Notasi	Nilai	Satuan
L	2.800	m
H	1.563	m
HCB	0.093	m
LLT	0.689	m
HVT3	0.326	m
HVT4	0.549	m
LLT1	0.300	m
LLT2	0.300	m
W	3482.6	N
B	3486.5	N

Berikut merupakan data perhitungan massa dan inersia dari AUV tipe *double-hull* di CEiiA berdasarkan *software* CAD menurut Bentes (2016):

**Tabel 3. 2** Massa dan Inersia dari AUV Tipe *Double-Hull* di CEiiA

Notasi	Nilai	Satuan
$M$	440.48	kg
$I_x$	58.35	kg.m <sup>2</sup>
$I_y$	148.71	kg.m <sup>2</sup>
$I_z$	94.04	kg.m <sup>2</sup>
$I_{xz}$	-2.44	kg.m <sup>2</sup>

Sedangkan di bawah ini merupakan nilai dari koefisien yang menunjukkan gaya hidrodinamis akibat *added mass*:

**Tabel 3. 3** Koefisien Hidrodinamis AUV Tipe *Double-Hull* di CEiiA akibat *Added Mass*

Notasi	Nilai	Satuan
$X_{\dot{u}}$	-11.64	kg
$X_{\dot{q}}$	1.74	kg.m
$Y_{\dot{v}}$	-450.58	kg
$Y_{\dot{p}}$	-23.33	kg.m
$Y_{\dot{r}}$	-82.23	kg.m
$Z_{\dot{w}}$	-419.13	kg
$Z_{\dot{q}}$	-76.78	kg.m
$K_{\dot{p}}$	-60.82	kg.m <sup>2</sup>
$K_{\dot{r}}$	-2.92	kg.m <sup>2</sup>
$M_{\dot{q}}$	-155.56	kg.m <sup>2</sup>
$N_{\dot{r}}$	-181.84	kg.m <sup>2</sup>

Koefisien hidrodinamik AUV di CEiiA dikelompokkan menjadi 4 berdasarkan arah gerak yang menimbulkan adanya redaman atau *damping*, yakni

*surge*, *sway*, *heave*, dan rotasi. Berikut merupakan nilai dari masing-masing koefisien hidrodinamik dari AUV yang digunakan:

**Tabel 3. 4** Koefisien Hidrodinamis AUV Tipe *Double-Hull* di CEiiA akibat *Damping*

Pengelompokan	Notasi	Nilai
(1)	(2)	(3)
<i>Surge</i>		
Koefisien redaman linear	$X_u$	-24.6
	$Y_u$	0
	$Z_u$	1.8
	$K_u$	0
	$M_u$	0
	$N_u$	0
Koefisien redaman kuadratik	$X_{u u }$	-5.2
	$Y_{u u }$	0
	$Z_{u u }$	3.1
	$K_{u u }$	0
	$M_{u u }$	0
	$N_{u u }$	0
Koefisien redaman kubik	$X_{uu u }$	-21.1
	$Y_{uu u }$	0
	$Z_{uu u }$	1.4
	$K_{uu u }$	0
	$M_{uu u }$	0
	$N_{uu u }$	0
Koefisien redaman kuartik	$X_{uuu u }$	3.79
	$Y_{uuu u }$	0
	$Z_{uuu u }$	0
	$K_{uuu u }$	0

Pengelompokan	Notasi	Nilai
(1)	(2)	(3)
Koefisien redaman kuartik	$M_{uuu u }$	0
	$N_{uuu u }$	0
<i>Sway</i>		
Koefisien redaman linear	$X_v$	0.0
	$Y_v$	-219.5
	$Z_v$	0.0
	$K_v$	79.7
	$M_v$	171.8
	$N_v$	-246.1
Koefisien redaman kuadratik	$X_{v v}$	0.0
	$Y_{v v}$	-1844.4
	$Z_{v v}$	0.0
	$K_{v v}$	-170.7
	$M_{v v}$	-52.6
	$N_{v v}$	-215.3
<i>Heave</i>		
Koefisien redaman linear	$X_w$	0.0
	$Y_w$	0.0
	$Z_w$	-284.8
	$K_w$	0.0
	$M_w$	100.3
	$N_w$	0.0
Koefisien redaman kuadratik	$X_{w w }$	0.0
	$Y_{w w }$	0.0
	$Z_{w w }$	-43.8
	$K_{w w }$	0.0
	$M_{w w }$	56.1

Pengelompokan	Notasi	Nilai
(1)	(2)	(3)
Koefisien redaman kuadratik	$N_{w w}$	0.0
Koefisien redaman kubik	$X_{ww w}$	0.0
	$Y_{ww w}$	0.0
	$Z_{ww w}$	-255.5
	$K_{ww w}$	0.0
	$M_{ww w}$	206.9
	$N_{ww w}$	0.0
Koefisien redaman kuartik	$X_{www w}$	0.0
	$Y_{www w}$	0.0
	$Z_{www w}$	10.8
	$K_{www w}$	0.0
	$M_{www w}$	-55.1
	$N_{www w}$	0.0
Rotasi		
Koefisien redaman linear	$K_p$	-34.0
	$M_q$	-59.0
	$N_r$	-45.0
Koefisien redaman kuadratik	$K_{p p}$	-84.0
	$M_{q q}$	-148.0
	$N_{r r}$	-100.0

AUV yang dikembangkan di CEiiA memiliki empat *thruster* identik sebagai sistem propulsif yang memberikan gaya dorong bagi AUV untuk bergerak. Keempat *thruster* dikategorikan menjadi dua, yakni dua *thruster* vertikal dan dua *thruster* horizontal. Tabel 3.5 berikut menunjukkan spesifikasi dan limitasi dari *range* keluaran keempat *thruster* identikal yang digunakan pada AUV di CEiiA (Mendes, 2017):

**Tabel 3. 5** Data Spesifikasi *Thruster*

Spesifikasi	Nilai	Satuan
Gaya dorong maksimum ke arah depan	120	N
Gaya dorong maksimum ke arah belakang	85	N
Tegangan operasi	120	V DC
Suplai arus	7	A
Konsumsi daya maksimum	800	W
Kecepatan rotasi	2000	rpm

### 3.4 Pemodelan Dinamik AUV

Pemodelan dinamik AUV dilakukan dengan memasukkan data dari spesifikasi AUV ke dalam matriks yang sudah dirumuskan pada bagian Teori Penunjang (Mendes, 2017).

#### 3.4.1 *Rigid Body*

Matriks untuk massa *rigid body*  $M_{RB}$  serta matriks Coriolis dan sentripetal  $C_{RB}$  dari AUV di CEiiA diperoleh dengan memasukkan data momen dan inersia pada Tabel 3.2 ke dalam persamaan 2.8 dan 2.9 sehingga didapatkan persamaan sebagai berikut:

$$M_{RB} = \begin{bmatrix} 440.48 & 0 & 0 & 0 & 0 & 0 \\ 0 & 440.48 & 0 & 0 & 0 & 0 \\ 0 & 0 & 440.48 & 0 & 0 & 0 \\ 0 & 0 & 0 & 58.35 & 0 & 2.44 \\ 0 & 0 & 0 & 0 & 148.71 & 0 \\ 0 & 0 & 0 & 2.44 & 0 & 94.04 \end{bmatrix} \quad (3.1)$$

dan

$$C_{RB}(v) \triangleq \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 440.48 w & -440.48 v \\ -440.48 w & 0 & 440.48 u \\ 440.48 v & -440.48 u & 0 \end{bmatrix}$$



$$\begin{bmatrix}
 0 & 440.48 w & -440.48 v \\
 -440.48 w & 0 & 440.48 u \\
 440.48 v & -440.48 u & 0 \\
 0 & -2.44 p - 94.04 r & -148.71 q \\
 -2.44 p - 94.04 r & 0 & -2.44 r + 58.35 p \\
 148.71 q & -2.44 r - 58.35 p & 0
 \end{bmatrix} \quad (3.2)$$

### 3.4.2 Added Mass

Untuk matriks massa serta matriks Coriolis dan sentripetal akibat tambahan massa dari fluida yang ikut dipindahkan, data dari Tabel 3.3 disubstitusikan ke dalam persamaan 2.13, 2.14, dan 2.15 sehingga diperoleh persamaan sebagai berikut:

$$\mathbf{M}_A = \begin{bmatrix}
 11.64 & 0 & 0 & 0 & -1.74 & 0 \\
 0 & 450.58 & 0 & 23.33 & 0 & 82.23 \\
 0 & 0 & 419.13 & 0 & 76.78 & 0 \\
 0 & 23.33 & 0 & 60.82 & 0 & 2.92 \\
 -1.74 & 0 & 76.78 & 0 & 155.56 & 0 \\
 0 & 82.23 & 0 & 2.92 & 0 & 181.84
 \end{bmatrix} \quad (3.3)$$

dan

$$\begin{aligned}
 a_1 &= -11.64 u + 1.74 q \\
 a_2 &= -450.58 v - 23.33 p - 82.23 r \\
 a_3 &= -419.13 w - 76.78 q \\
 b_1 &= -23.33 v - 60.82 p - 2.92 r \\
 b_2 &= 1.74 u - 76.78 w - 155.56 q \\
 b_3 &= -82.23 v - 2.92 p - 181.84 r
 \end{aligned} \quad (3.4)$$

### 3.4.3 Hidrodinamik

Matriks *damping* linear dan kuadratik akibat gaya hidrodinamik diperoleh dengan mensubstitusikan data pada Tabel 3.4 ke dalam persamaan 2.18 dan 2.19, serta persamaan tambahan untuk matriks *damping* kubik dan kuartik sehingga diperoleh persamaan berikut:

$$\mathbf{D}_l = \begin{bmatrix}
 24.6 & 0 & 0 & 0 & 0 & 0 \\
 0 & 219.5 & 0 & 0 & 0 & 0 \\
 -1.8 & 0 & 284.8 & 0 & 0 & 0 \\
 0 & -79.7 & 0 & 34.0 & 0 & 0 \\
 0 & -171.8 & -100.3 & 0 & 59.0 & 0 \\
 0 & 246.1 & 0 & 0 & 0 & 45.0
 \end{bmatrix} \quad (3.5)$$

$$D_q(v) = \begin{bmatrix} 5.2|u| & 0 & 0 & 0 & 0 & 0 \\ 0 & 1844.4|v| & 0 & 0 & 0 & 0 \\ -3.1|u| & 0 & 43.8|w| & 0 & 0 & 0 \\ 0 & -170.7|v| & 0 & 84.0 & 0 & 0 \\ 0 & 52.6|v| & -56.1|w| & 0 & 148.0|q| & 0 \\ 0 & 215.3|v| & 0 & 0 & 0 & 100|r| \end{bmatrix} \quad (3.6)$$

$$D_{cub} = \begin{bmatrix} 21.1 uu & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1.4 uu & 0 & 255 ww & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -206.9 ww & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.7)$$

$$D_{qt}(v) = \begin{bmatrix} -3.79|u| uu & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -10.8 |w| ww & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 55.1 |w| ww & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.8)$$

#### 3.4.4 Hidrostatik

Berdasarkan Gambar 3.2, titik berat atau *center of gravity* (CG) terletak pada posisi di sumbu x dan y yang sama dengan pusat apung atau *center of buoyancy* (CB), namun terpisahkan sebesar HCB di sumbu z. Berdasarkan Tabel 3.1, besar HCB adalah 0.093 meter, sehingga koordinat CG dan CB dapat dituliskan sebagai berikut:

$$r_g^b = [0 \ 0 \ 0]^T \quad (3.9)$$

$$r_B^b = [0 \ 0 \ -0.093]^T \quad (3.10)$$

Adanya jarak sebesar 0.093 meter di antara kedua pusat menimbulkan momen yang mampu mengontrol *roll* dan *pitch* secara pasif (Mendes, 2017).

Berdasarkan Tabel 3.1, AUV di CEiiA memiliki berat sebesar 3482.6 N dan gaya apung sebesar 3486.5 N sehingga selisih keduanya adalah 3.9240 N. Dengan mensubstitusikan nilai tersebut ke dalam persamaan 2.21 diperoleh persamaan berikut:

$$g(\eta) = \begin{bmatrix} -3.9240 s\theta \\ 3.9240 c\theta s\phi \\ 3.9240 c\theta c\phi \\ -324.24 c\theta s\phi \\ -324.4 s\theta \\ 0 \end{bmatrix} \quad (3.11)$$

di mana  $s \cdot = \sin(\cdot)$  dan  $c \cdot = \cos(\cdot)$ .

### 3.4.5 Thruster

Sebagaimana telah disebutkan sebelumnya, AUV yang dikembangkan di CEiiA memiliki 4 buah *thruster*, di mana masing-masing *thruster* mampu menghasilkan gaya dorong ke depan sebesar 120 N dan gaya dorong ke belakang sebesar 85 N (Mendes, 2017). Berikut merupakan matriks vektor lokasi dari masing-masing *thruster*, relative terhadap titik berat atau *center of gravity* (CG):

- *Thruster* longitudinal bagian kanan (*Thruster 1*):

$$r_{T1}^g = [-0.689 \quad 0.300 \quad 0] \quad (3.12)$$

- *Thruster* longitudinal bagian kiri (*Thruster 2*):

$$r_{T2}^g = [-0.689 \quad -0.300 \quad 0] \quad (3.13)$$

- *Thruster* vertikal bagian depan (*Thruster 3*):

$$r_{T3}^g = [0.910 \quad 0 \quad 0.326] \quad (3.14)$$

- *Thruster* vertikal bagian belakang (*Thruster 4*):

$$r_{T4}^g = [-0.910 \quad 0 \quad -0.549] \quad (3.15)$$

Nilai dari koordinat *thruster* pada persamaan 3.12, 3.13, 3.14, dan 3.15 disubstitusikan ke dalam persamaan berikut:

$$L = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -|r_{T3}^g(1)| & |r_{T4}^g(1)| & 0 & 0 \\ -|r_{T1}^g(2)| & |r_{T2}^g(2)| & 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.16)$$

di mana  $|r_{T1}^g(2)|$  menunjukkan nilai mutlak dari matriks vektor *Thruster* 1 kolom kedua, dan seterusnya sehingga diperoleh matriks  $L$  untuk persamaan 2.11 sebagai berikut:

$$L = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.910 & -0.910 & 0 & 0 \\ 0.300 & -0.300 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.17)$$

Supaya dapat mengkonstruksi persamaan 2.11 perlu matriks vektor kontrol  $U$  yang dituliskan sebagai berikut:

$$U = [U_1 \ U_2 \ U_3 \ U_4 \ 0 \ 0]^T \quad (3.18)$$

di mana  $U_1$ ,  $U_2$ ,  $U_3$ , dan  $U_4$  merupakan gaya dorong (dalam Newton) yang dihasilkan dari *Thruster* 1, 2, 3, dan 4 secara berurutan (Mendes, 2017).

Dengan menggabungkan matriks  $L$  dan  $U$  yang telah diperoleh, maka persamaan 2.11 dapat dikonstruksi sebagai berikut:

$$\tau = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.910 & -0.910 & 0 & 0 \\ 0.300 & -0.300 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ 0 \\ 0 \end{bmatrix} \quad (3.19)$$

di mana

$$\tau = [\tau_1 \ 0 \ \tau_3 \ 0 \ \tau_5 \ \tau_6]^T \quad (3.20)$$

merupakan sinyal kontrol yang dihasilkan oleh kontroler yang akan dibahas pada sub-bab berikutnya. Pada pemodelan yang dilakukan oleh Mendes (2017), dinamika aktuator dan sensor tidak dipertimbangkan.

### 3.4.6 Linearisasi Pemodelan Dinamik

Kontroler PID merupakan kontroler linear sehingga pemodelan dinamik AUV yang telah dirumuskan pada persamaan 2.22 dan 2.23 perlu dilinearisasi.

Secara sederhana, linearisasi dapat digambarkan memiliki tujuan untuk ‘memisahkan’ variabel *state* sehingga persamaan 2.22 dan 2.23 yang dituliskan kembali dalam bentuk persamaan 3.21:

$$\begin{bmatrix} \dot{v} \\ \dot{\eta} \end{bmatrix} = \begin{bmatrix} -M^{-1}(C(v)v + D(v)v + g(\eta)) \\ J(\eta)v \end{bmatrix} + \begin{bmatrix} M^{-1} \\ 0 \end{bmatrix} \tau \quad (3.21)$$

berubah bentuknya menjadi seperti persamaan *state space* berikut:

$$\boxed{\dot{x} = Ax + Bu} \quad (3.22)$$

Mendes (2017) telah melinearisasikan persamaan AUV di CEiiA menggunakan deret Taylor, di mana:

$$\Delta v = v - v_0 \quad (3.23)$$

$$\Delta \eta = \eta - \eta_0 \quad (3.24)$$

dengan  $v_0$  dan  $\eta_0$ , menurut Isiyel (2007) adalah titik linearisasi yang didefinisikan sebagai:

$$v_0(t) = [u_0(t), v_0(t), w_0(t), p_0(t), q_0(t), r_0(t)] \quad (3.25)$$

$$\eta_0(t) = [x_0(t), y_0(t), z_0(t), \phi_0(t), \theta_0(t), \psi_0(t)] \quad (3.26)$$

dan anggap:

$$f_c(v) = C(v)v \quad (3.27)$$

$$f_d(v) = D(v)v \quad (3.28)$$

Suku elemen matriks pada persamaan 3.21 diturunkan secara parsial terhadap variabel yang menjadi fungsinya ( $v$  atau  $\eta$ ), kemudian disubstitusikan dengan nilai variabel saat mencapai titik kesetimbangan ( $v_0$  atau  $\eta_0$ ) sehingga diperoleh persamaan:

$$M\Delta\dot{v} + \left. \frac{\partial f_c(v)}{\partial v} \right|_{v_0} \Delta v + \left. \frac{\partial f_d(v)}{\partial v} \right|_{v_0} \Delta v + \left. \frac{\partial g(\eta)}{\partial \eta} \right|_{\eta_0} \Delta \eta = \Delta \tau \quad (3.29)$$

untuk baris pertama dari matriks persamaan 3.21, dan:

$$\dot{\eta} = J(\eta_0)v_0 + \left. \frac{\partial J(\eta)v}{\partial v} \right|_{v_0\eta_0} \Delta v + \left. \frac{\partial J(\eta)v}{\partial \eta} \right|_{v_0\eta_0} \Delta \eta \quad (3.30)$$

untuk baris kedua dari matriks persamaan 3.21.

Karena  $\dot{\eta} = \dot{\eta}_0 + \Delta\dot{\eta}$  dan  $\dot{\eta}_0 = J(\eta_0)v_0$ , maka persamaan 3.28 dapat dipersingkat menjadi:

$$\Delta\dot{\eta} \approx J(\eta_0)\Delta v + \left. \frac{\partial J(\eta)}{\partial \eta} \right|_{\eta_0} v_0\Delta \eta \quad (3.31)$$

Dengan mendefinisikan matriks  $x = (x_1, x_2)^T$  di mana  $x_1 = \Delta v$  dan  $x_2 = \Delta \eta$ , maka persamaan 3.27 dan 3.29 dapat dituliskan kembali menjadi:

$$M\dot{x}_1 + Cx_1 + Dx_1 + Gx_2 = \tau \quad (3.32)$$

$$\dot{x}_2 = Jx_1 + J^* \quad (3.33)$$

di mana  $C = \left. \frac{\partial f_c(v)}{\partial v} \right|_{v_0}$ ;  $D = \left. \frac{\partial f_d(v)}{\partial v} \right|_{v_0}$ ;  $G = \left. \frac{\partial g(\eta)}{\partial \eta} \right|_{\eta_0}$ ;  $J = J(\eta_0)$ ;  $J^* = \left. \frac{\partial J(\eta)}{\partial \eta} \right|_{\eta_0} v_0$ .

Selanjutnya, dengan mengasumsikan  $\tau = u$ , maka persamaan dinamik AUV di CEiiA yang telah dilinearisasikan dapat ditulis kembali dalam bentuk persamaan *state space* seperti pada persamaan 3.22 menjadi sebagai berikut:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -M^{-1}[C + D] & -M^{-1}G \\ J & J^* \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} M^{-1} \\ 0 \end{bmatrix} u \quad (3.34)$$

di mana  $M$  adalah jumlah dari massa *rigid body* dan *added mass* sesuai persamaan 3.1 dan 3.3,  $D$  adalah koefisien *damping* linier ( $D_l$ ) sesuai persamaan 3.5. Namun untuk matriks  $C$  diperbarui menjadi:

$$C = \begin{bmatrix} 0 & C_{12} \\ -C_{12}^T & C_{22} \end{bmatrix} \quad (3.35)$$

$C_{12} =$

$$\begin{bmatrix} 0 & -X_{\dot{w}}u_0 + (m - Z_{\dot{w}})w_0 & X_{\dot{v}}u_0 + Y_{\dot{w}}w_0 \\ X_{\dot{w}}u_0 - (m - Z_{\dot{w}})w_0 & 0 & (m - X_{\dot{u}})u_0 - X_{\dot{w}}w_0 \\ -X_{\dot{v}}u_0 - Y_{\dot{w}}w_0 & -(m - X_{\dot{u}})u_0 + X_{\dot{w}}w_0 & 0 \end{bmatrix} \quad (3.36)$$

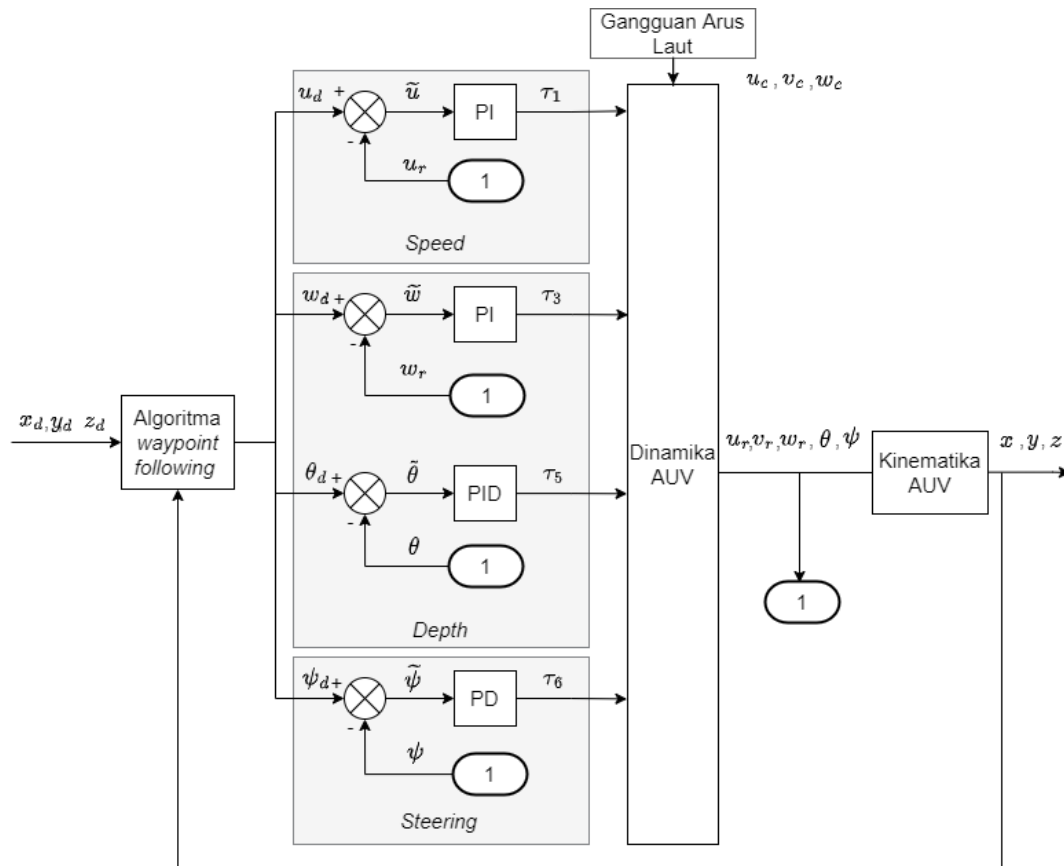
$$C_{22} = \begin{bmatrix} 0 & -(X_{\dot{r}}u_0 - Z_{\dot{r}}w_0) & X_{\dot{q}}u_0 + Z_{\dot{q}}w_0 \\ X_{\dot{r}}u_0 + Z_{\dot{r}}w_0 & 0 & -(X_{\dot{p}}u_0 + Z_{\dot{p}}w_0) \\ -(X_{\dot{q}}u_0 - Z_{\dot{q}}w_0) & X_{\dot{p}}u_0 + Z_{\dot{p}}w_0 & 0 \end{bmatrix} \quad (3.37)$$

Sedangkan matriks  $G$  menjadi:

$$G = \begin{bmatrix} 0 & 0 & 0 & 0 & (W - B) & 0 \\ 0 & 0 & 0 & -(W - B) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & (z_g W - z_B B) & 0 & 0 \\ 0 & 0 & 0 & 0 & (z_g W - z_B B) & 0 \\ 0 & 0 & 0 & -(x_g W - x_B B) & -(y_g W - y_B B) & 0 \end{bmatrix} \quad (3.38)$$

### 3.5 Perancangan Pengendali PID

Persamaan di atas masih kurang praktikal untuk sistem pengendali yang umumnya terpisah, sistem 6 DOF dipisahkan ke dalam 3 subsistem dengan DOF yang memiliki satu fungsi yang sama (Jalving, 1994).



**Gambar 3.3** Diagram Blok Perancangan Pengendali PID

Gambar 3.3 menunjukkan diagram blok sistem pengendali PID yang terbagi ke dalam tiga subsistem dan terdiri atas empat pengendali. Subsistem *speed* akan mengendalikan  $u(t)$ ; subsistem *steering* akan mengendalikan  $v(t)$ ,  $r(t)$ , dan  $\psi(t)$ ; sedangkan subsistem *depth* akan mengendalikan  $w(t)$ ,  $q(t)$ , dan  $\theta(t)$ . Untuk gerakan *roll* yakni  $p(t)$  dan  $\phi(t)$  dapat diabaikan karena bentuk AUV yang simetris (Xu, Zhang, Cao, Pang, & Sun, *Tracking Control Based on Command Filter and Disturbance Observer*, 2019). Perancangan dari masing-masing subsistem mengutip dari penelitian sebelumnya (Mendes, 2017):

### 3.5.1 Subsistem Pengendali Kecepatan (*Speed*)

Dari matriks persamaan *state space* yang telah diturunkan sebelumnya, diperoleh bahwa untuk mendapatkan nilai keluaran berupa sinyal kontrol  $\tau_1$  yang mengendalikan kecepatan digunakan persamaan sebagai berikut:

$$(m - X_{\dot{u}})\dot{u} - X_u u = \tau_1 \quad (3.39)$$

Dengan mensubstitusikan nilai  $m$ ,  $X_{\dot{u}}$ , dan  $X_u$  dari spesifikasi AUV ke dalam persamaan 3.39, diperoleh persamaan fungsi transfer sebagai berikut:

$$\frac{u(s)}{\tau_1(s)} = \frac{0.0022}{s+0.0544} \quad (3.40)$$

di mana  $u(s)$  adalah nilai *surge* yang dihasilkan oleh masukan berupa sinyal kontrol  $\tau_1$ .

Nilai dari sinyal kontrol  $\tau_1$  sendiri ditentukan oleh kontroler *surge*. Dalam penelitian kali ini, kontroler *surge* menggunakan kontroler PI karena terbukti mampu mencapai kecepatan yang diinginkan (Jalving, 1994) sehingga dapat dirumuskan sebagai berikut:

$$\tau_1 = K_p \tilde{u} + K_i \int_0^t \tilde{u}(\tau) d\tau \quad (3.41)$$

atau bila dinyatakan dalam bentuk Laplace menjadi:

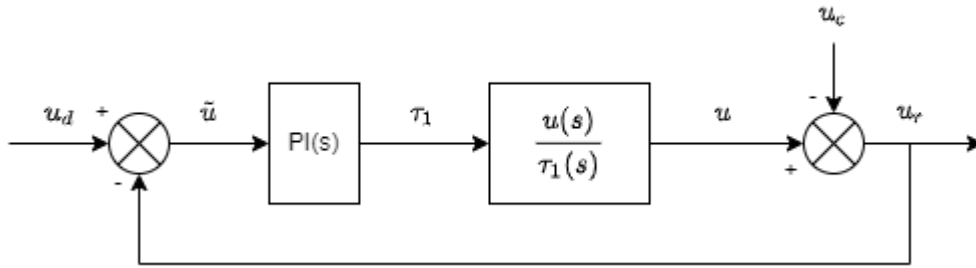
$$\tau_1(s) = K_p \tilde{u}(s) + \frac{K_i}{s} \tilde{u}(s) \quad (3.42)$$



di mana

$$\tilde{u} = u_d - u_r \quad (3.43)$$

dengan  $u_d$  adalah referensi *surge* yang ditentukan oleh algoritma *waypoint following*, sedangkan  $u_r$  adalah nilai *surge* aktual sebagai keluaran dari pembacaan sensor kecepatan AUV.



**Gambar 3. 4** Diagram Blok Subsistem Pengendali Kecepatan (*Surge*)

Gambar 3.4 menunjukkan diagram blok subsistem pengendali kecepatan (*speed*) atau *surge* ketika ada gangguan arus laut  $u_c$  yang akan dibahas lebih lanjut pada subbab 3.6.

### 3.5.2 Subsistem Pengendali Kedalaman (*Depth*)

Untuk mengendalikan kedalaman AUV, ada dua hal yang bisa secara aktif dilakukan, yakni dengan dengan mengendalikan *heave* melalui sinyal kontrol  $\tau_3$ , atau dengan mengendalikan inklinasi atau *pitch* dari AUV melalui sinyal kontrol  $\tau_5$ . Berikut merupakan persamaan *state space* yang digunakan dalam subsistem pengendali kedalaman (Mendes, 2017):

$$\begin{bmatrix} m - Z_{\dot{w}} & -Z_{\dot{q}} & 0 \\ -M_{\dot{w}} & (I_y - M_{\dot{q}}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{w} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} -Z_w & (m - X_{\dot{u}})u_0 - Z_q & 0 \\ (m - X_{\dot{u}})u_0 - M_w & -M_q & (z_g W + z_B B) \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} w \\ q \\ \theta \end{bmatrix} = \begin{bmatrix} \tau_3 \\ \tau_5 \\ 0 \end{bmatrix} \quad (3.44)$$

Dengan mensubstitusikan nilai-nilai dari spesifikasi AUV, persamaan 3.44 dapat dituliskan kembali ke dalam bentuk umum *state space* 3.22 sebagai berikut:

$$\begin{bmatrix} \dot{w} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} -0.2604 & 0.4482 & 0.0974 \\ -0.7934 & -0.3070 & -1.0902 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} w \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} 0.0012 & -0.0003 \\ -0.0003 & 0.0034 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \tau_3 \\ \tau_5 \end{bmatrix} \quad (3.45)$$

$$y_{\text{untuk mendapat nilai } w} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} w \\ q \\ \theta \end{bmatrix} \quad (3.46)$$

$$y_{\text{untuk mendapat nilai } \theta} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} w \\ q \\ \theta \end{bmatrix} \quad (3.47)$$

Persamaan 3.45 dan 3.46 kemudian diubah ke dalam persamaan fungsi transfer. Untuk *heave* diperoleh:

$$\frac{w(s)}{\tau_3(s)} = \frac{0.00119s^2 + 0.0002308s + 0.001268}{s^3 + 0.5675s^2 + 1.526s + 0.3612} \quad (3.48)$$

di mana  $w(s)$  adalah nilai *heave* yang dihasilkan oleh masukan berupa sinyal kontrol  $\tau_3$ .

Nilai dari sinyal kontrol  $\tau_3$  sendiri ditentukan oleh kontroler *heave*. Dalam penelitian kali ini, kontroler *heave* menggunakan kontroler PI karena terbukti mampu mencapai kecepatan yang diinginkan (Jalving, 1994) sehingga dapat dirumuskan sebagai berikut:

$$\tau_3 = K_p \tilde{w} + K_i \int_0^t \tilde{w}(\tau) d\tau \quad (3.49)$$

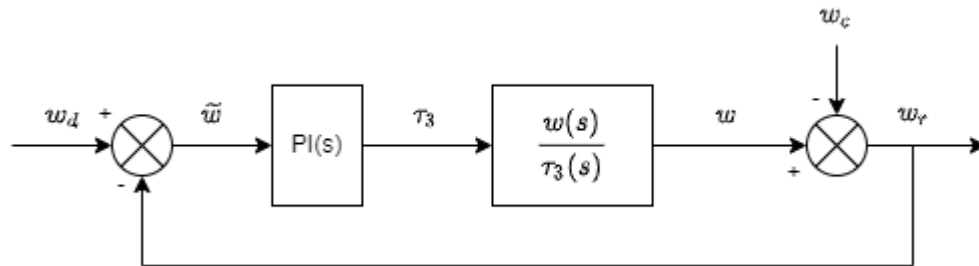
atau bila dinyatakan dalam bentuk Laplace menjadi:

$$\tau_3(s) = K_p \tilde{w}(s) + \frac{K_i}{s} \tilde{w}(s) \quad (3.50)$$

di mana

$$\tilde{w} = w_d - w_r \quad (3.51)$$

dengan  $w_d$  adalah referensi *heave* yang ditentukan oleh algoritma *waypoint following*, sedangkan  $w_r$  adalah nilai *heave* aktual sebagai keluaran dari pembacaan sensor tekanan pada AUV.



**Gambar 3. 5** Diagram Blok Subsistem Pengendali Kedalaman (*Heave*)

Gambar 3.5 menunjukkan diagram blok subsistem pengendali kedalaman (*depth*) melalui variabel kontrol *heave* ketika ada gangguan arus laut  $w_c$  yang akan dibahas lebih lanjut pada subbab 3.6.

Sedangkan untuk *pitch*, diperoleh persamaan fungsi transfer sebagai berikut (Mendes, 2017):

$$\frac{\theta(s)}{\tau_5(s)} = \frac{0.003362s + 0.001114}{s^3 + 0.5675s^2 + 1.526s + 0.3612} \quad (3.52)$$

di mana  $\theta(s)$  adalah nilai *pitch* yang dihasilkan oleh masukan berupa sinyal kontrol  $\tau_5$ .

Nilai dari sinyal kontrol  $\tau_5$  sendiri ditentukan oleh kontroler *pitch*. Dalam penelitian kali ini, kontroler *pitch* menggunakan kontroler PI karena terbukti mampu mencapai kecepatan yang diinginkan (Jalving, 1994) sehingga dapat dirumuskan sebagai berikut:

$$\tau_5 = K_p \tilde{\theta} + K_i \int_0^t \tilde{\theta}(\tau) d\tau + K_d \frac{d\tilde{\theta}}{dt} \quad (3.53)$$

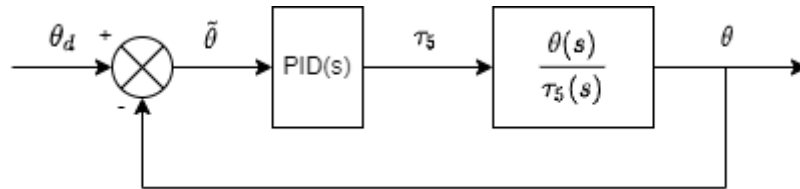
atau bila dinyatakan dalam bentuk Laplace menjadi:

$$\tau_5(s) = K_p \tilde{\theta}(s) + \frac{K_i}{s} \tilde{\theta}(s) + K_d \frac{N}{1+s} \tilde{\theta}(s) \quad (3.54)$$

di mana  $N$  adalah *filter coefficient* pada blok PID di Simulink, dan

$$\tilde{\theta} = \theta_d - \theta \quad (3.55)$$

dengan  $\theta_d$  adalah referensi *pitch* yang ditentukan oleh algoritma *waypoint following*, sedangkan  $\theta$  adalah nilai *pitch* aktual sebagai keluaran dari pengukuran inklinasi (inklinometer) AUV.



**Gambar 3. 6** Diagram Blok Subsistem Pengendali Kedalaman (*Pitch*)

Gambar 3.6 menunjukkan diagram blok subsistem pengendali kedalaman (*depth*) melalui variabel kontrol *heave* ketika ada gangguan arus laut  $w_c$  yang akan dibahas lebih lanjut pada subbab 3.6.

### 3.5.3 Subsistem Pengendali Haluan (*Steering*)

Untuk fungsi pengendalian haluan, ada 3 *state variabel* yang berkaitan, yakni  $v(r), r(t)$ , dan  $\psi(t)$ . Berikut merupakan persamaan *state space* untuk subsistem *steering* menurut Mendes (2017):

$$\begin{bmatrix} m - Y_{\dot{v}} & -Y_{\dot{r}} & 0 \\ -N_{\dot{v}} & (I_z - N_{\dot{r}}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{v} \\ \dot{r} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} -Y_v v & -Y_r + mu_0 & 0 \\ -N_v & -N_r & 0 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} v \\ r \\ \psi \end{bmatrix} = \begin{bmatrix} 0 \\ \tau_6 \\ 0 \end{bmatrix} \quad (3.54)$$

Dengan mensubstitusikan nilai-nilai dari spesifikasi AUV, persamaan 3.51 dapat dituliskan kembali ke dalam bentuk umum *state space* 3.22 sebagai berikut:

$$\begin{bmatrix} \dot{v} \\ \dot{r} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} -0.1653 & -0.4167 & 0 \\ 0.5672 & -0.0503 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} v \\ r \\ \psi \end{bmatrix} + \begin{bmatrix} -0.0003 \\ 0.0037 \\ 0 \end{bmatrix} \tau_6 \quad (3.55)$$

$$y_{\text{untuk mendapat nilai } \psi} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ r \\ \psi \end{bmatrix} \quad (3.56)$$

atau bila dinyatakan dalam fungsi transfer menjadi:

$$\frac{\psi(s)}{\tau_6(s)} = \frac{0.003729s+0.0009888}{s^3+0.3549s^2+0.2147s} \quad (3.57)$$

di mana nilai masukan sinyal kontrol  $\tau_6$  diperoleh dari keluaran kontroler PD yang dirumuskan sebagai berikut:

$$\tau_6 = K_p \tilde{\psi} - K_d \frac{d\tilde{\psi}}{dt} \quad (3.58)$$

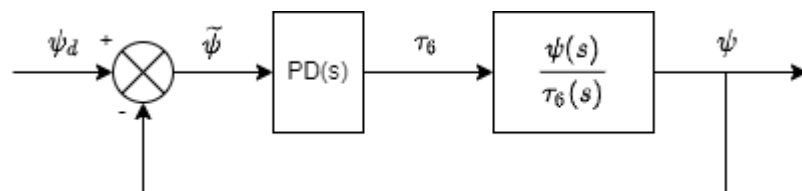
atau bila dinyatakan dalam bentuk Laplace menjadi:

$$\tau_6(s) = K_p \tilde{\psi}(s) + \frac{K_i}{s} \tilde{\psi}(s) + K_d \frac{N}{1+\frac{N}{s}} \tilde{\psi}(s) \quad (3.59)$$

dengan  $N$  adalah *filter coefficient* pada blok PID di Simulink, dan

$$\tilde{\psi} = \psi_d - \psi \quad (3.60)$$

di mana  $\psi_d$  adalah referensi *yaw* yang ditentukan oleh algoritma *waypoint following*, sedangkan  $\psi$  adalah nilai *yaw* aktual sebagai keluaran dari pengukuran kompas AUV.



**Gambar 3. 7** Diagram Blok Subsistem Pengendali Haluan (*Yaw*)

Gambar 3.7 menunjukkan diagram blok subsistem pengendali kedalaman (*depth*) melalui variabel kontrol *heave* ketika ada gangguan arus laut  $w_c$  yang akan dibahas lebih lanjut pada subbab 3.6.

### 3.5.4 Metode *Tuning* Parameter PID

Nilai  $K_p$  sebagai *gain* parameter proporsional, nilai  $K_i$  sebagai *gain* parameter integral, dan  $K_d$  sebagai *gain* parameter derivatif dari kontroler PI, PD, dan PID yang telah dirumuskan di atas akan ditentukan menggunakan *toolbox* PID *Tuner* pada Simulink 2016a untuk memperoleh parameter awal, kemudian

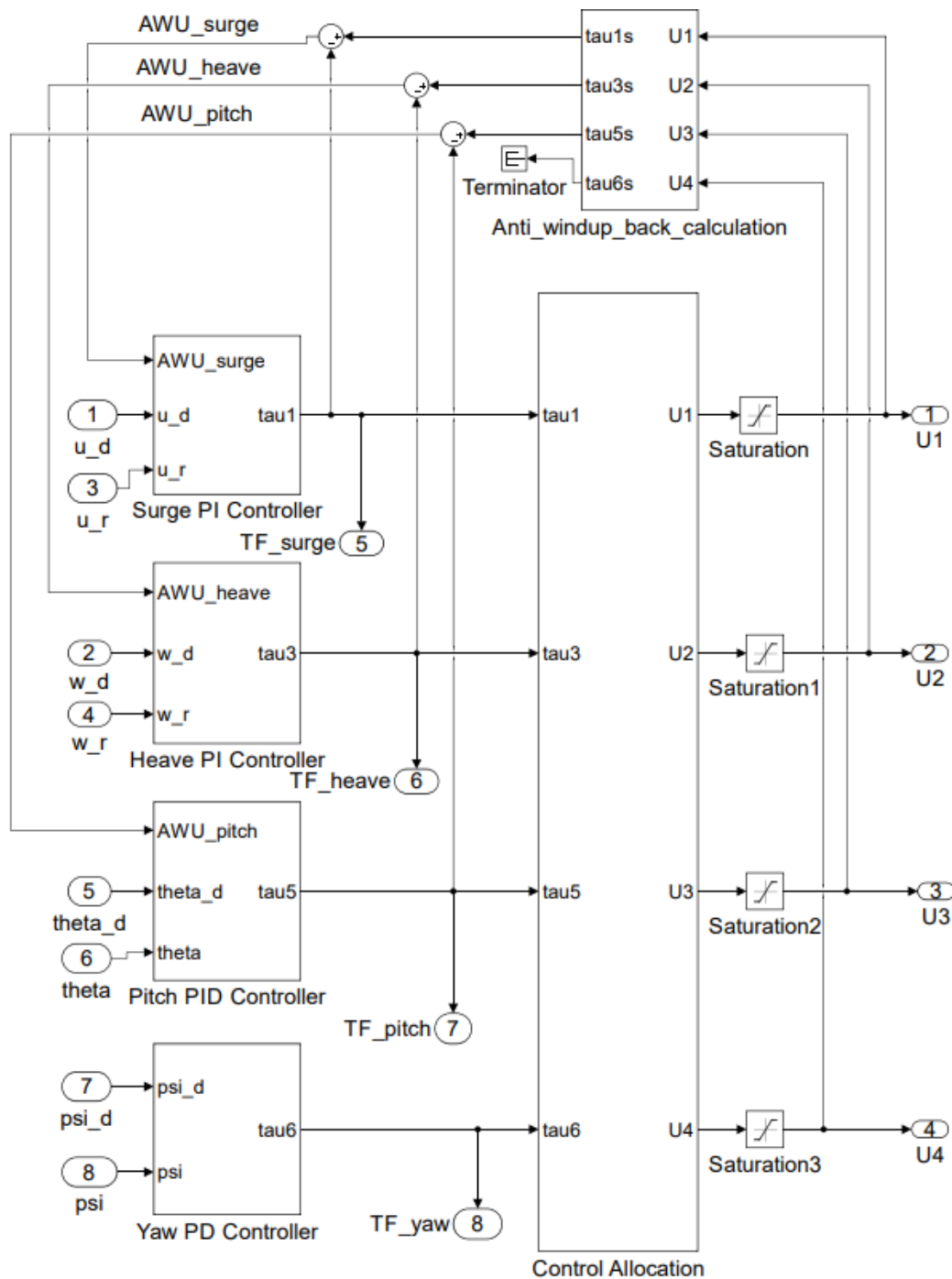
digabungkan dengan sistem nonlinier AUV dan di-*tuning* secara manual untuk mencapai performansi yang diinginkan. Pada penelitian ini diambil performansi yang diinginkan adalah *overshoot* kurang dari 1% dan *settling time* kurang dari 15 detik. Metode *tuning* PID yang serupa telah banyak digunakan dalam perancangan kontroler AUV dan terbukti mampu menghasilkan performa yang diinginkan (Villa, Vallicrosa, Aaltonen, Ridao, & Koskinen, 2021), (Khodayari & Balochian, 2015).

Nilai parameter kontroler PID perlu dianalisa kestabilan hasil responnya. Analisa kestabilan dilakukan dengan menggunakan diagram Bode sesuai dengan teori yang telah disebutkan dalam Dasar Teori, di mana sistem dikatakan stabil apabila: sistem *open-loop* stabil dan semakin besar nilai *phase margin* serta *gain margin* maka sistem akan semakin stabil (Electrical4U, 2021), (Hahn, Edison, & Edgar, 2001).

### 3.5.5 Alokasi Kontrol dan Kompensator *Anti-Windup*

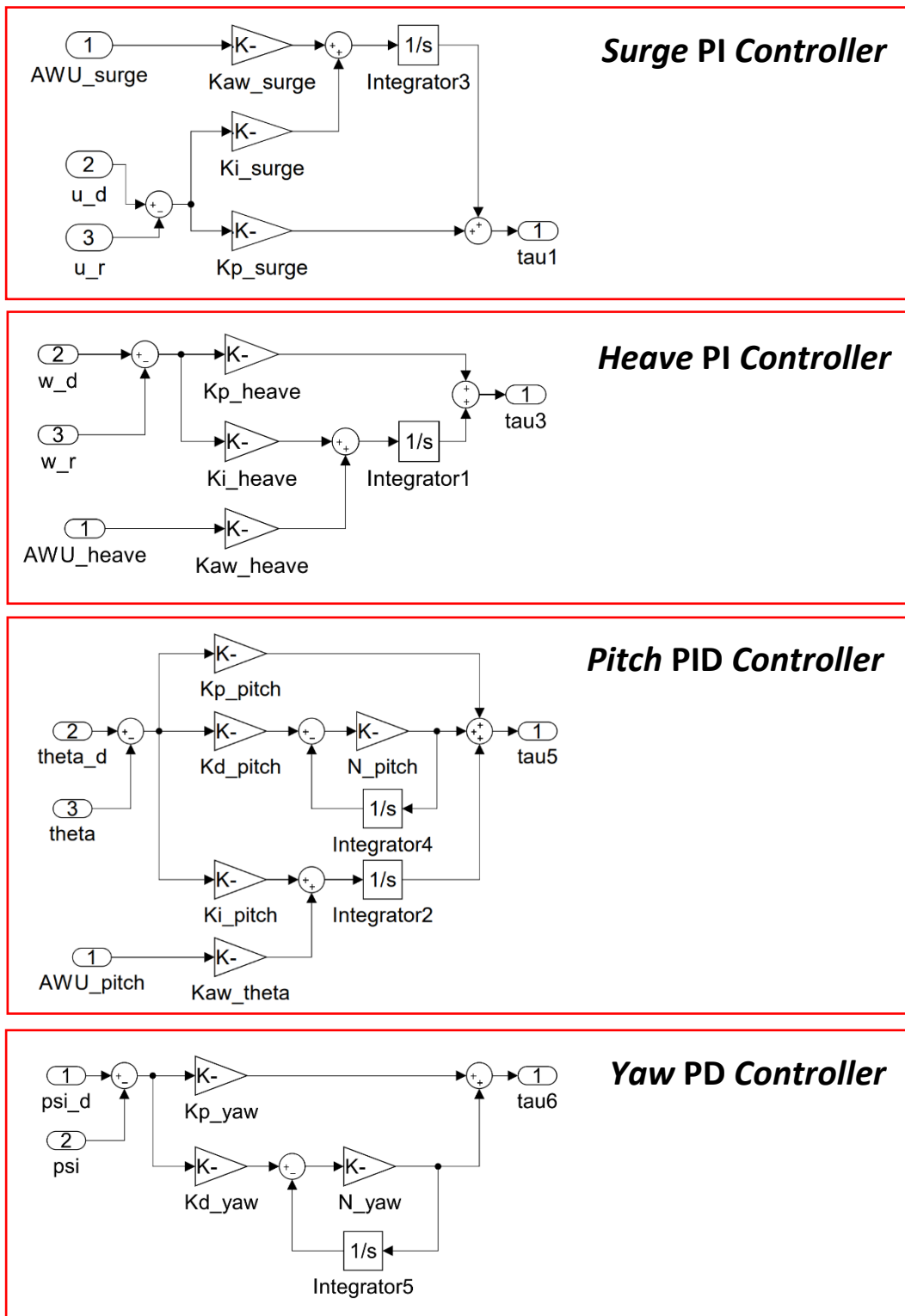
Dalam spesifikasi AUV diperoleh data keluaran maksimum yang dapat diberikan oleh aktuator, yakni 120 N untuk ke depan dan 85 N untuk ke belakang. Untuk keluaran minimum diasumsikan 0 N. Adanya limitasi keluaran pada aktuator perlu dipertimbangkan dalam perancangan sistem pengendali meskipun dinamika aktuator dalam penelitian ini tidak dipertimbangkan, karena dapat menimbulkan fenomena *windup* seperti dijelaskan dalam Dasar Teori.

Gambar 3.8 menunjukkan skema metode kompensator fenomena *windup* atau disebut *anti-windup* yang digunakan dalam penelitian ini, yakni *back-calculation*. Pada metode *back-calculation*, selisih nilai antara sinyal kontrol dengan keluaran kontroler setelah blok saturasi akan dihitung dan hasilnya dikalikan dengan suatu *gain*  $1/T_t$  di mana  $T_t$  adalah *time tracking constant* yang nilainya ditentukan menggunakan persamaan 2.47 dan 2.48, kemudian hasilnya dijumlahkan dengan parameter integral pada kontroler untuk mengatur ulang aksi integral dalam kontroler PI atau PID supaya adanya saturasi keluaran aktuator tidak menyebabkan akumulasi pada integrator.



**Gambar 3. 8** *Anti-Windup Back Calculation*

Gambar 3.9 menunjukkan bagan perhitungan sinyal kontrol dari subsistem pengendali *surge*, *heave*, *pitch*, dan *yaw* pada sistem pengendali PID secara berurutan setelah diintegrasikan dengan *anti-windup back-calculation*:



**Gambar 3. 9** Subsistem Pengendali PID dengan *Anti-Windup Back-Calculation*

Untuk keluaran minimum diasumsikan 0 N. Adanya limitasi keluaran pada aktuator perlu dipertimbangkan dalam perancangan sistem pengendali meskipun



dinamika aktuator dalam penelitian ini tidak dipertimbangkan, karena dapat menimbulkan fenomena *windup* seperti dijelaskan dalam Dasar Teori.

Gambar 3.8 menunjukkan skema metode kompensator fenomena *windup* atau disebut *anti-windup* yang digunakan dalam penelitian ini, yakni *back-calculation*. Pada metode *back-calculation*, selisih nilai antara sinyal kontrol dengan keluaran kontroler setelah blok saturasi akan dihitung dan hasilnya dikalikan dengan suatu *gain*  $1/T_t$  di mana  $T_t$  adalah *time tracking constant* yang nilainya ditentukan menggunakan persamaan 2.47 dan 2.48, kemudian hasilnya dijumlahkan dengan parameter integral pada kontroler untuk mengatur ulang aksi integral dalam kontroler PI atau PID supaya adanya saturasi keluaran aktuator tidak menyebabkan akumulasi pada integrator.

Meskipun dinamika aktuator tidak dimodelkan, dalam penelitian ini kontribusi sinyal kontrol terhadap gaya yang dikeluarkan keempat *thruster* dapat diketahui dari persamaan 3.19 dan 3.20, yang dituliskan kembali sebagai matriks alokasi kontrol sebagai berikut:

$$\begin{bmatrix} \tau_1 \\ 0 \\ \tau_3 \\ 0 \\ \tau_5 \\ \tau_6 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.910 & -0.910 & 0 & 0 \\ 0.300 & -0.300 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ 0 \\ 0 \end{bmatrix} \quad (3.61)$$

Jika sinyal kontrol  $\tau_1$ ,  $\tau_3$ ,  $\tau_5$ , dan  $\tau_6$  sebagai keluaran dari kontroler *surge*, *heave*, *pitch*, dan *yaw* diketahui, maka nilai gaya  $U_1$ ,  $U_2$ ,  $U_3$ , dan  $U_4$  yang dihasilkan oleh *thruster* identik 1, 2, 3, dan 4 dapat diketahui pula dengan persamaan yang diturunkan dari persamaan 3.55, yakni:

$$\begin{aligned} U_1 &= \frac{0.3 \tau_1 + \tau_6}{0.6} \\ U_2 &= \frac{0.3 \tau_1 - \tau_6}{0.6} \\ U_3 &= \frac{0.91 \tau_3 + \tau_5}{1.82} \\ U_4 &= \frac{0.91 \tau_3 - \tau_5}{1.82} \end{aligned} \quad (3.62)$$

Masing-masing nilai keluaran *thruster*  $U_1$ ,  $U_2$ ,  $U_3$ , dan  $U_4$  hasil perhitungan akan dilewatkan pada blok saturasi yang akan memastikan keluaran *thruster* dalam simulasi tidak melebihi batas maksimum dan minimum aktuator sesuai data spesifikasi, yakni 120 N dan 0 N. Secara matematis, perhitungan dalam blok saturasi dapat dituliskan sebagai berikut:

$$U_n' = \begin{cases} 120 \text{ N}, & \text{if } U_n \geq 120 \text{ N} \\ U_n, & \text{if } 0 \text{ N} < U_n < 120 \text{ N} \\ 0 \text{ N}, & \text{if } U_n \leq 0 \text{ N} \end{cases} \quad (3.63)$$

di mana  $U_n$  menunjukkan nilai keluaran dari *thruster* ke- $n$  sebagaimana didapatkan dari persamaan 3.56, dan  $U_n'$  adalah nilai keluaran *thruster* ke- $n$  setelah melalui blok saturasi.

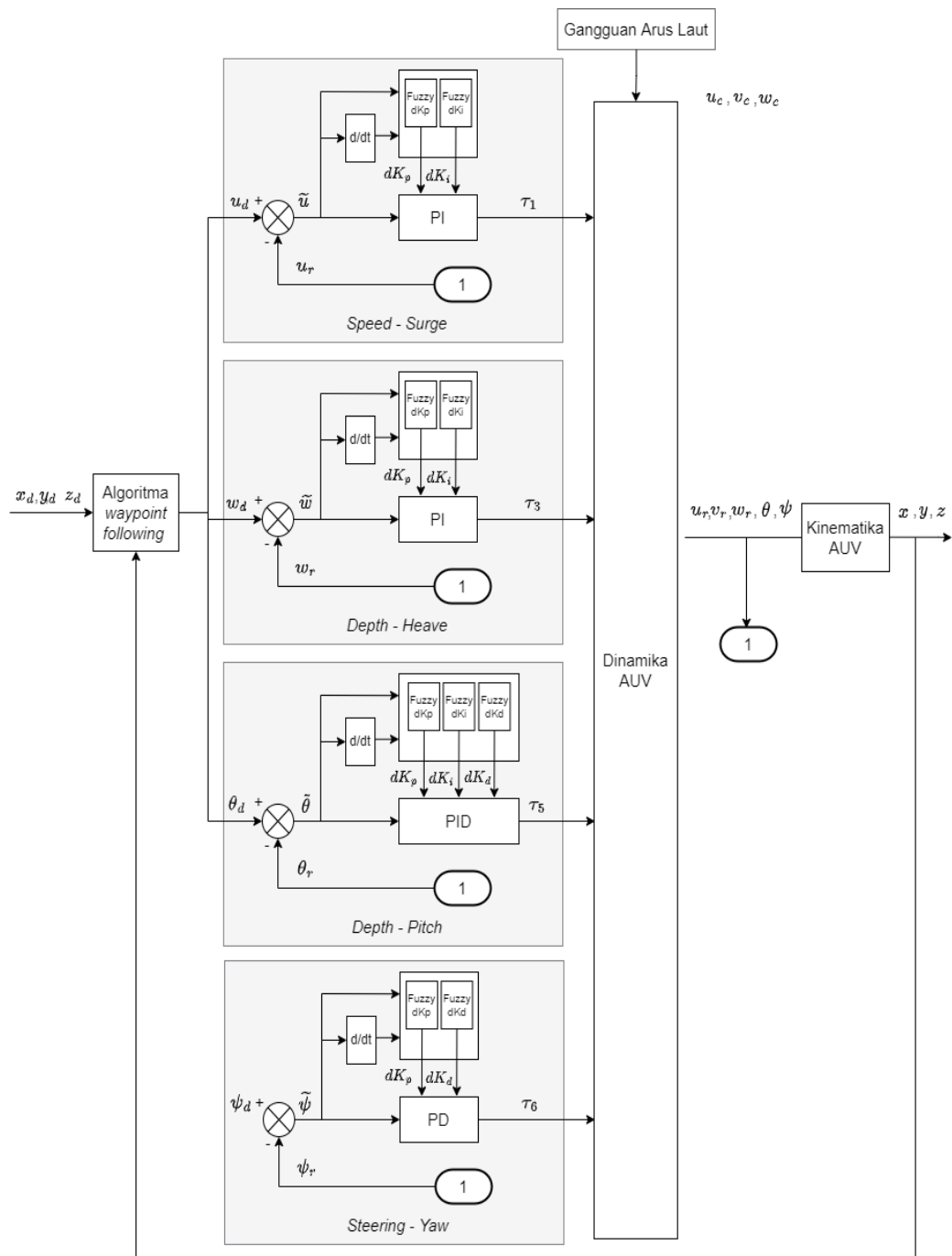
Pada *feedback loop* tambahan, nilai  $U_n'$  akan digunakan untuk menghitung kembali nilai sinyal kontrol yang seharusnya, atau dituliskan  $\tau_n'$ , yang mana nilai dari  $\tau_n'$  dapat diturunkan dari matriks alokasi kontrol pada persamaan 3.55 menjadi:

$$\begin{aligned} \tau_1' &= U_1' + U_2' \\ \tau_3' &= U_3' + U_4' \\ \tau_5' &= 0.91 U_3' - 0.91 U_4' \\ \tau_6' &= 0.3 U_1' - 0.3 U_2' \end{aligned} \quad (3.64)$$

Selisih nilai  $\tau_n'$  dengan nilai  $\tau_n$  kemudian dihitung, dan hasilnya ( $AWU$ ) dikalikan dengan *gain*  $K_{AW}$  yang bernilai  $1/T_t$  dari masing-masing kontroler untuk dijumlahkan dengan *gain* parameter integral pada kontroler PI dan PID. Pada subbab sebelumnya, kontroler *yaw* tidak menggunakan aksi integral, sehingga  $\tau_6'$  tidak perlu dihitung selisihnya dengan  $\tau_6$  dan tidak digunakan.

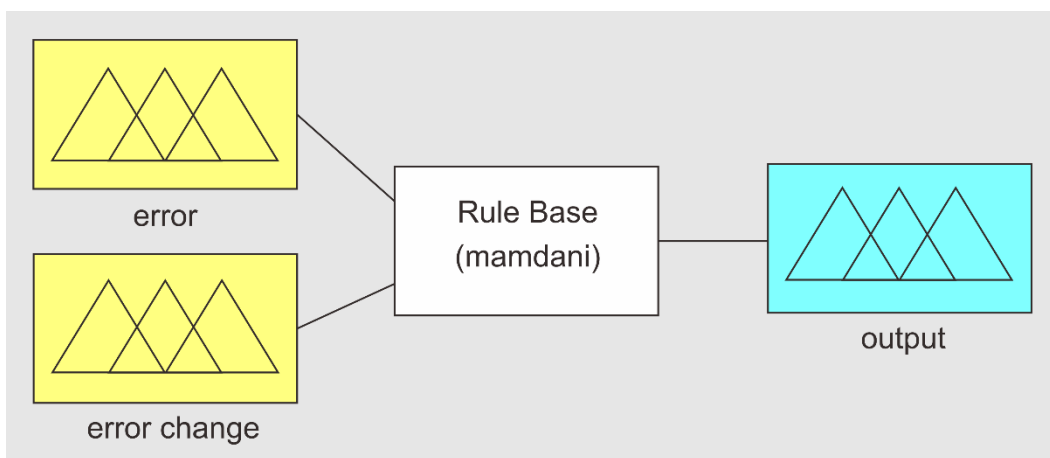
### 3.6 Perancangan Pengendali *Fuzzy*-PID

Perancangan pengendali *fuzzy*-PID dalam penelitian ini mengacu pada perancangan pengendali *fuzzy*-PID yang sebelumnya (Hammad, Elshenawy, & Singaby, 2017) dan diaplikasikan untuk tipe AUV yang berbeda serta diadaptasi dari bentuk diskrit menjadi bentuk kontinyu.



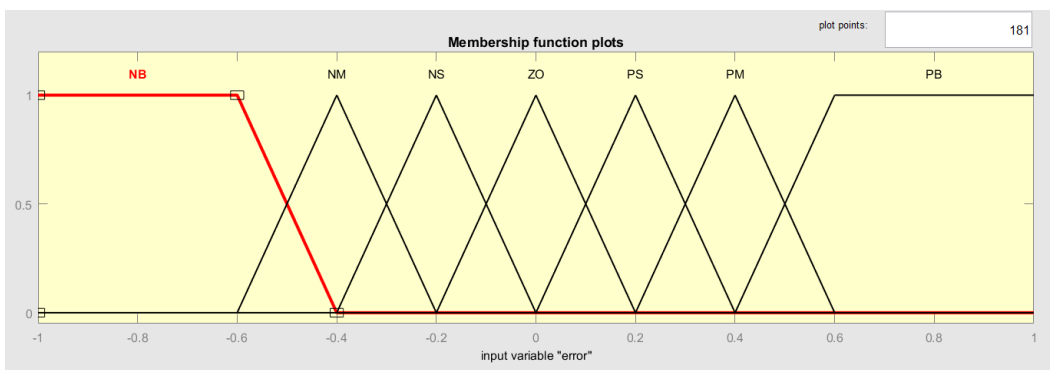
**Gambar 3. 10** Diagram Blok Perancangan Pengendali *Fuzzy*-PID

Gambar 3.10 menunjukkan diagram blok sistem pengendali *fuzzy*-PID. Secara garis besar, diagram kurang lebih sama dengan sistem pengendali PID, namun perbedaannya terletak pada adanya blok *fuzzy* di atas blok pengendali PID dari masing-masing subsistem yang berfungsi menentukan nilai *gain* yang harus ditambahkan pada parameter PID dari proses *tuning* sebelumnya agar pengendali PID mampu melakukan *self-tuning*. Untuk kontroler PI *surge* dan *heave* akan diadaptasi menjadi *fuzzy*-PI; untuk kontroler PID *pitch* akan diadaptasi menjadi *fuzzy*-PID; sedangkan untuk kontroler PD *yaw* akan diadaptasi menjadi *fuzzy*-PD.

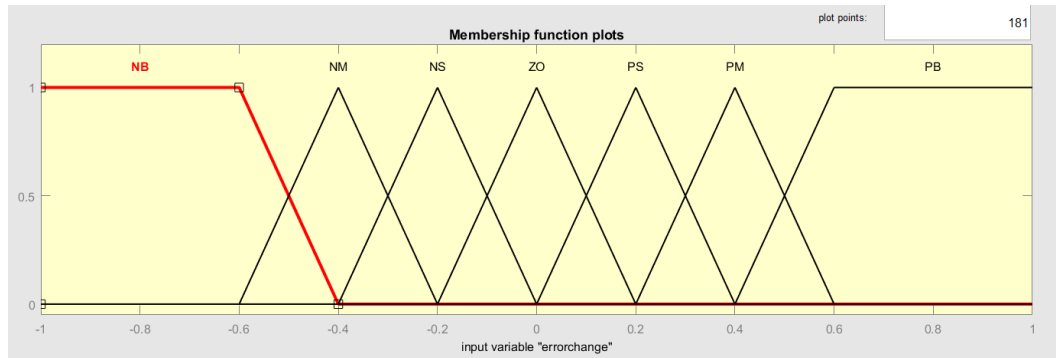


**Gambar 3. 11** Struktur Logika *Fuzzy* untuk *Self-Tuning* PID

Skema pengendali logika *fuzzy*-PID yang digunakan, seperti terlihat pada Gambar 3.11, terdiri atas 2 masukan, yakni *error* dan *error rate*, serta 1 keluaran, yakni nilai tambahan untuk parameter *gain* proporsional ( $dK_p$ ), integral ( $dK_i$ ), atau derivatif ( $dK_d$ ).

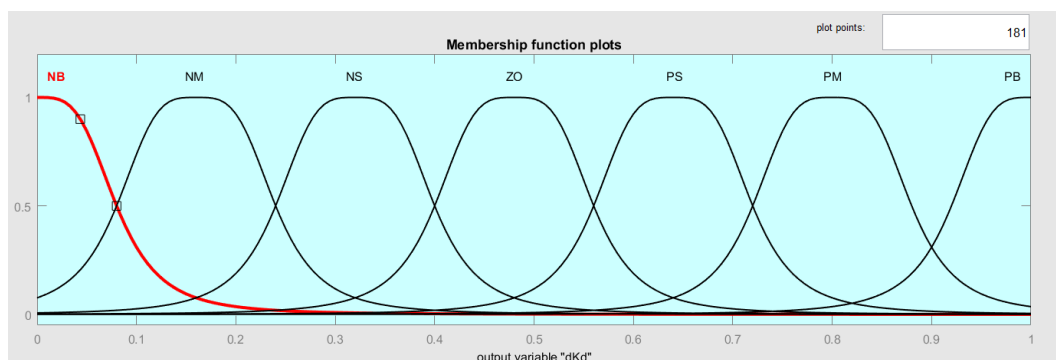


**Gambar 3. 12** Fungsi Keanggotaan Masukan 1 (*Error*)



**Gambar 3.13** Fungsi Keanggotaan Masukan 2 (*Error Rate*)

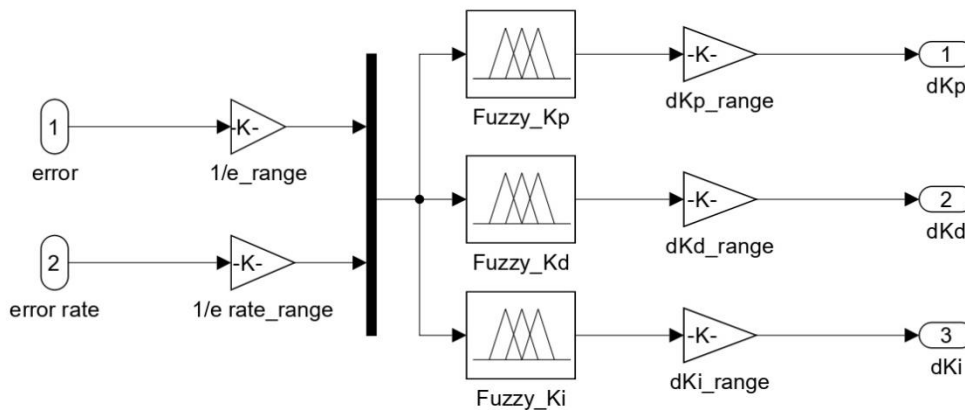
Nilai *error* dan *error rate* akan dinormalisasi terlebih dahulu dengan cara dibagi dengan nilai *range* maksimum-nya, sehingga fungsi keanggotaan-nya memiliki *range* antara -1 hingga 1, seperti ditunjukkan pada Gambar 3.12 dan Gambar 3.13. Nilai *range* maksimum dari *error* diperoleh dari batas maksimum dan minimum dari kecepatan *surge*, *heave*, dan sudut maksimum serta minimum yang dapat diraih oleh *pitch* dan *yaw*, sedangkan nilai *range* maksimum dari *error rate* diperoleh dari simulasi dengan menggunakan blok *Signal Builder* kemudian menetapkan *setpoint* nilai maksimum lalu berubah jadi nilai minimum *error*, lalu sinyal  $e(t)$  diturunkan menggunakan blok *derivatif* pada Simulink. Bentuk fungsi keanggotaan yang dipilih adalah segitiga (Hammad, Elshenawy, & Singaby, 2017) dan terdiri dari tujuh fungsi keanggotaan, yakni *Negative Big* (NB), *Negative Medium* (NM), *Negative Small* (NS), *Zero* (ZO), *Positive Small* (PS), *Positive Medium* (PM), dan *Positive Big* (PB).



**Gambar 3.14** Fungsi Keanggotaan Keluaran  $dK_p$ ,  $dK_i$ , atau  $dK_d$

Gambar 3.14 menunjukkan bahwa keluaran dari struktur *fuzzy*-PID yang digunakan terdiri dari tujuh fungsi keanggotaan berbentuk *gaussian* (Hammad, Elshenawy, & Singaby, 2017), yakni *Negative Big* (NB), *Negative Medium* (NM), *Negative Small* (NS), *Zero* (ZO), *Positive Small* (PS), *Positive Medium* (PM), dan *Positive Big* (PB).

Nilai  $dK_p$ ,  $dK_i$ , dan  $dK_d$  juga dinormalisasi sehingga *range* pada fungsi keanggotaan keluaran logika *fuzzy* hanya 0-1 seperti pada Gambar 3.14, sehingga harus dikalikan dengan *range*  $dK_p$ ,  $dK_i$ , dan  $dK_d$  yang dipilih melalui iterasi.



**Gambar 3. 15** Representasi Simulink dari Struktur Logika *Fuzzy* yang Digunakan

Gambar 3.15 menunjukkan representasi Simulink dari logika *fuzzy* yang digunakan untuk mengkondisikan masukan berupa *error* dan *error rate* sebelum dimasukkan ke logika *fuzzy* dan digunakan untuk menentukan tambahan nilai  $dK_p$ ,  $dK_i$ , dan  $dK_d$ .

**Tabel 3. 6** Tabel Iterasi untuk Memilih *Range*  $dK_p$ ,  $dK_i$ , dan  $dK_d$  Terbaik

Iterasi ke-	<i>Range</i> $dK_p$	<i>Range</i> $dK_i$	<i>Range</i> $dK_d$
1	$1 \times K_p$	$1 \times K_i$	$1 \times K_d$
2	$2 \times K_p$	$2 \times K_i$	$2 \times K_d$
3	$5 \times K_p$	$5 \times K_i$	$5 \times K_d$
4	$10 \times K_p$	$10 \times K_i$	$10 \times K_d$
5	$100 \times K_p$	$100 \times K_i$	$100 \times K_d$

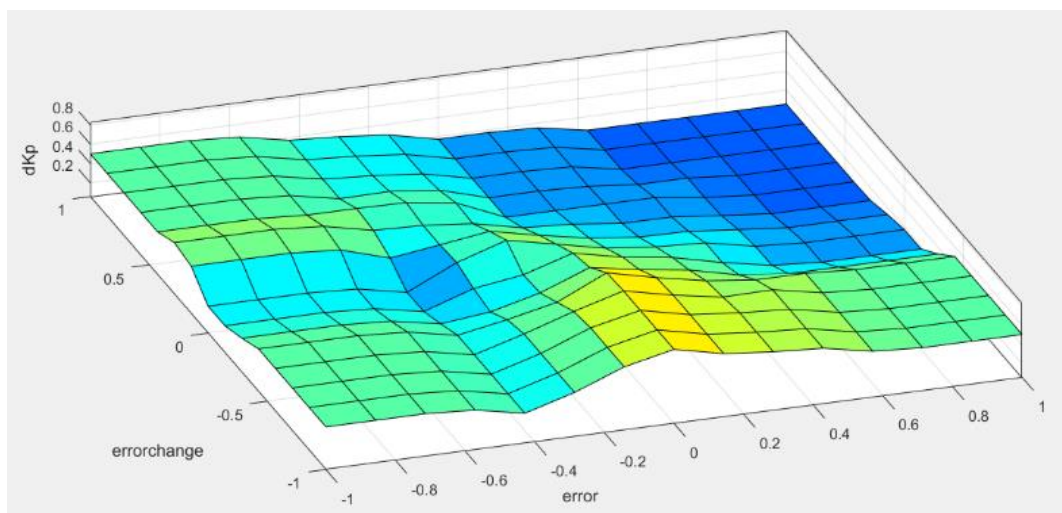
Tabel 3.6 menunjukkan variasi yang dilakukan dalam melakukan iterasi untuk menentukan  $range$   $dK_p$ ,  $dK_i$ , dan  $dK_d$  dengan performansi terbaik di bawah gangguan arus laut.

*Rule base* untuk masing-masing keluaran  $dK_p$ ,  $dK_i$ , dan  $dK_d$  dapat dilihat sebagai berikut (Hammad, Elshenawy, & Singaby, 2017):

**Tabel 3. 7** *Rule Base* untuk  $dK_p$

e\ce	NB	NM	NS	ZO	PS	PM	PB
NB	ZO	ZO	NS	NS	PS	ZO	ZO
NM	NS	NS	NM	NM	NS	ZO	NS
NS	PS	ZO	NS	PS	ZO	ZO	NS
ZO	PM	PM	PS	ZO	NS	NM	NM
PS	PS	PS	ZO	NS	NM	NM	NM
PM	PS	ZO	NS	NS	NM	NM	NB
PB	ZO	ZO	NM	NM	NB	NB	NB

Gambar 3.16 berikut menunjukkan penampang permukaan dari *rule base* yang digunakan untuk menentukan  $dK_p$ :



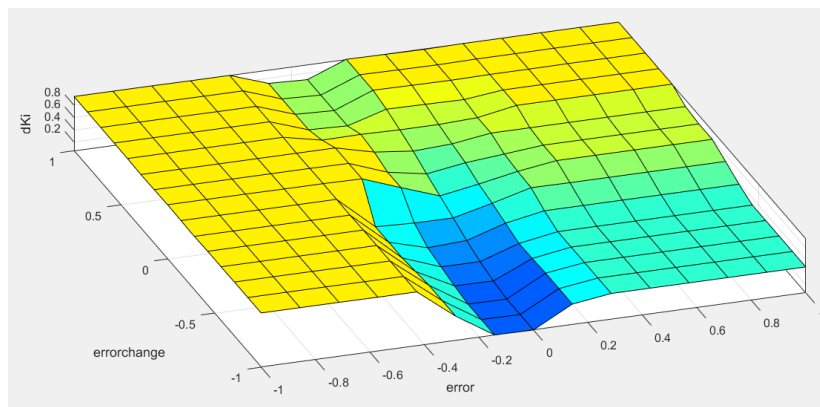
**Gambar 3. 16** Penampang Permukaan *Rule Base*  $dK_p$

Berikut merupakan tabel untuk menentukan keluaran  $dK_i$ :

**Tabel 3. 8** Rule Base untuk  $dK_i$ 

e\ce	NB	NM	NS	ZO	PS	PM	PB
NB	PB	PB	PB	PB	PB	PB	PB
NM	PB	PB	PB	PB	PB	PB	PB
NS	NB	NM	NM	PB	PB	PB	PS
ZO	NB	NM	NS	ZO	PS	PM	PB
PS	ZO	ZO	PS	PS	PM	PM	PB
PM	ZO	ZO	PS	PM	PM	PB	PB
PB	ZO	ZO	PS	PM	PM	PB	PB

Gambar 3.17 berikut menunjukkan penampang permukaan dari aturan fuzzy yang digunakan untuk menentukan  $dK_i$ :

**Gambar 3. 17** Penampang Permukaan Rule Base  $dK_i$ 

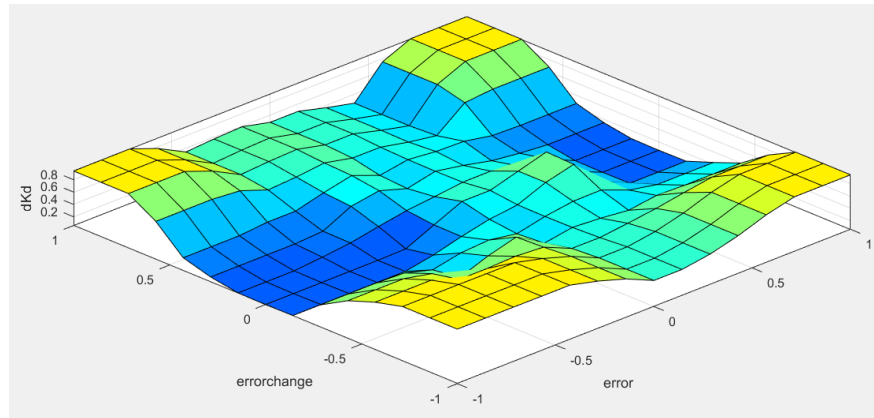
Berikut merupakan tabel untuk menentukan keluaran  $dK_d$ :

**Tabel 3. 9** Rule Base untuk  $dK_d$ 

e\ce	NB	NM	NS	ZO	PS	PM	PB
NB	PB	PS	NB	NB	NB	NM	PB
NM	PB	NM	NB	NB	NM	NM	ZO
NS	PM	PB	NM	NB	NS	PS	PS
ZO	ZO	NS	NS	ZO	NS	NS	ZO
PS	ZO	ZO	ZO	ZO	ZO	ZO	ZO
PM	PS	NB	PB	PB	NM	NS	NB
PB	PB	NS	NB	NB	NB	NM	PB



Gambar 3.18 berikut menunjukkan penampang permukaan dari aturan *fuzzy* yang digunakan untuk menentukan  $dK_d$ :



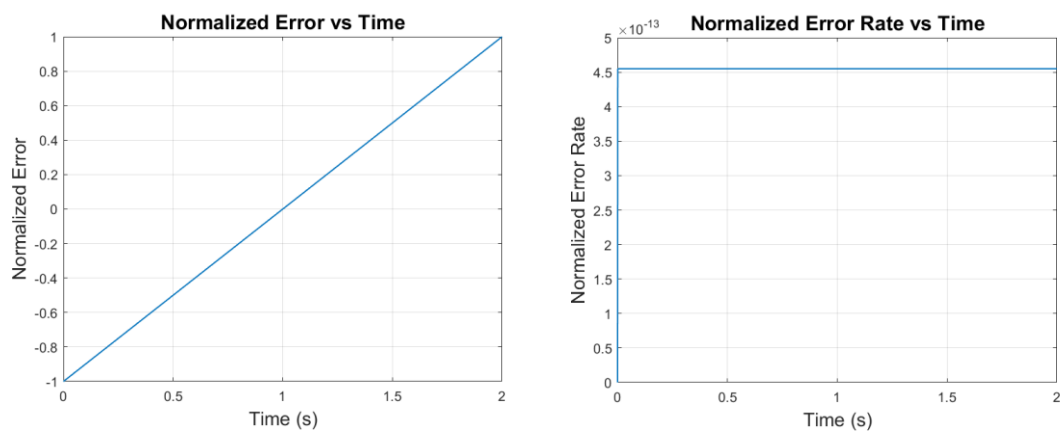
**Gambar 3. 18** Penampang Permukaan *Rule Base*  $dK_d$

Nilai parameter PID setelah di-*tuning* menggunakan logika *fuzzy* menjadi:

$$\begin{aligned} K'_p(t) &= K_p + dK_p(t) \\ K'_i(t) &= K_i + dK_i(t) \\ K'_d(t) &= K_d + dK_d(t) \end{aligned} \quad (3.65)$$

di mana  $K_p$ ,  $K_i$ , dan  $K_d$  adalah nilai parameter *gain* proporsional, integral, dan derivatif yang diperoleh dari *tuning* PID menggunakan MATLAB *Toolbox* pada tahap sebelumnya.

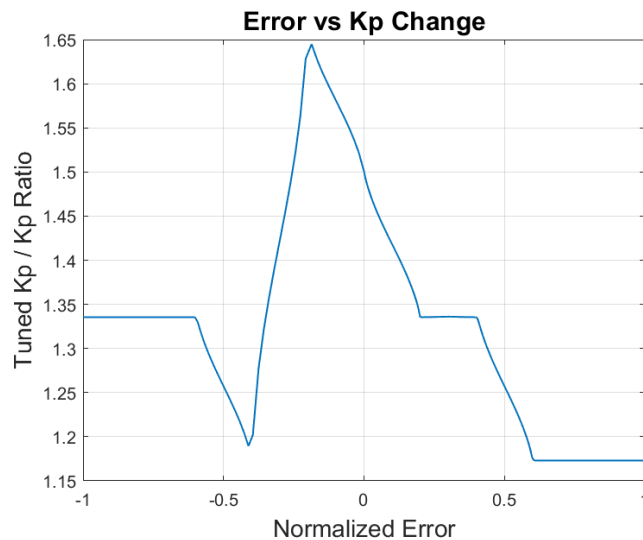
Misal salah satu subsistem diberi masukan sedemikian rupa sehingga diperoleh *error* serta *error rate* yang telah dinormalisasi sebagai berikut:



**Gambar 3. 19** Grafik Contoh *Error* dan *Error Rate* terhadap Waktu

Gambar 3.19 menunjukkan nilai *error* yang telah dinormalisasi terus meningkat dari  $-1$  ke  $1$  terhadap waktu, dengan kecepatan atau *rate* konstan senilai kurang lebih  $4.5E - 13$ . Berikutnya akan dilihat bagaimana perubahan nilai *error* dan *error rate* ini mempengaruhi nilai parameter *gain* proporsional, integral, dan derivatif sesuai *rule base* yang telah dijelaskan sebelumnya.

Berikut merupakan grafik hubungan antara nilai *error* yang telah dinormalisasi dengan rasio antara nilai akhir parameter *gain* proporsional ( $K_p'$ ), setelah *tuning* menggunakan logika *fuzzy* terhadap nilai awal parameter *gain* proporsional ( $K_p$ ):



**Gambar 3. 20** Grafik Perubahan  $K_p$  terhadap *Error* yang Telah Dinormalisasi

Dari Gambar 3.20 dapat dilihat bahwa ketika *error* yang telah dinormalisasi terus meningkat dari  $-1$  ke  $1$  dengan *error rate* konstan senilai  $4.5E - 13$  (mendekati  $0$ ) maka nilai  $K_p'$  tidak selalu meningkat atau tidak linear. Ketika *error*  $-1$  maka nilai  $K_p'$  akan menjadi 1.35 kali lipat dari nilai  $K_p$ . Ketika *error* berkurang menjadi  $-0.25$  maka nilai  $K_p'$  akan bertambah menjadi kurang lebih 1.5 kali lipat dari nilai  $K_p$ . Namun semakin nilai *error* mendekati  $1$  maka nilai  $K_p'$  menjadi 1.2 kali lipat dari nilai  $K_p$ . Hal tersebut disebabkan oleh non-linearitas sistem yang dikendalikan sehingga untuk perubahan nilai *error* tertentu diperlukan penambahan nilai parameter proporsional yang sesuai untuk menjaga sistem tetap stabil. Hal ini sesuai dengan penampang permukaan *rule base* untuk menentukan nilai  $dK_p$  seperti

tampak pada Gambar 3.16, di mana ketika nilai *error rate* positif mendekati 0 dan nilai *error* meningkat dari  $-1$  ke  $1$  maka nilai  $dK_p$  akan berubah dari zona dengan kontur lebih rendah, kemudian meningkat hingga *error* sekitar  $0.2$ , kemudian kembali ke kontur rendah secara bertahap.

Berikut merupakan grafik hubungan antara nilai *error* yang telah dinormalisasi dengan rasio antara nilai akhir parameter *gain* integral ( $K_i'$ ), setelah *tuning* menggunakan logika *fuzzy* terhadap nilai awal parameter *gain* integral ( $K_i$ ):

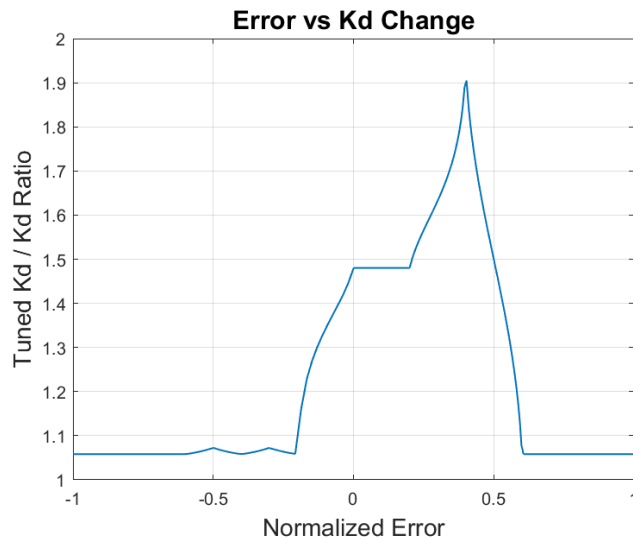


**Gambar 3. 21** Grafik Perubahan  $K_i$  terhadap *Error* yang Telah Dinormalisasi

Dari Gambar 3.21 dapat dilihat bahwa ketika *error* yang telah dinormalisasi terus meningkat dari  $-1$  ke  $1$  dengan *error rate* konstan senilai  $4.5E - 13$  (mendekati 0) maka nilai  $K_i'$  tidak selalu meningkat atau tidak linear. Ketika *error*  $-1$  maka nilai  $K_i'$  akan tinggi, yakni sekitar  $1.94$  kali lipat dari nilai  $K_i$ . Ketika *error* berkurang menjadi  $-0.2$  maka nilai  $K_i'$  akan bertambah menjadi kurang lebih  $1.75$  kali lipatnya saja dari nilai  $K_i$ . Namun semakin nilai *error* mendekati  $0.5$  maka nilai  $K_i'$  akan meningkat lagi menjadi  $1.8$  kali lipat dari nilai  $K_i$ . Hal tersebut disebabkan oleh non-linearitas sistem yang dikendalikan sehingga untuk perubahan nilai *error* tertentu diperlukan penambahan nilai parameter integral yang sesuai untuk menjaga sistem tetap stabil namun tetap mampu mencapai *error steady state* yang minimum. Hal ini sesuai dengan penampang permukaan *rule base* untuk menentukan nilai  $dK_i$  seperti tampak pada Gambar 3.17, di mana ketika nilai *error rate* positif mendekati 0 dan nilai *error* meningkat dari  $-1$  ke  $1$  maka nilai  $dK_i$  mula-mula akan stagnan

di kontur tinggi, kemudian menurun, dan meningkat lagi sebelum akhirnya mencapai kontur stagnan berwarna hijau.

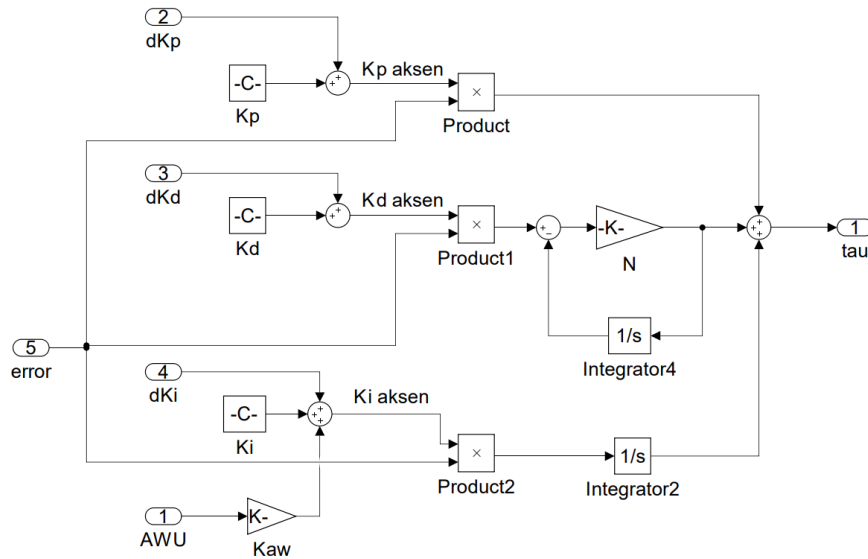
Berikut merupakan grafik hubungan antara nilai *error* yang telah dinormalisasi dengan rasio antara nilai akhir parameter *gain* derivatif ( $K'_d$ ), setelah *tuning* menggunakan logika *fuzzy* terhadap nilai awal parameter *gain* derivatif ( $K_d$ ):



**Gambar 3.22** Grafik Perubahan  $K_d$  terhadap *Error* yang Telah Dinormalisasi

Dari Gambar 3.22 dapat dilihat bahwa ketika *error* yang telah dinormalisasi terus meningkat dari  $-1$  ke  $1$  dengan *error rate* konstan senilai  $4.5E - 13$  (mendekati  $0$ ) maka nilai  $K'_d$  tidak selalu meningkat atau tidak linear. Ketika *error*  $-1$  maka nilai  $K'_d$  akan rendah, yakni sekitar 1.05 kali lipat dari nilai  $K_d$ . Ketika *error* berkurang menjadi  $0.25$  maka nilai  $K'_d$  akan bertambah menjadi kurang lebih 1.6 kali lipatnya dari nilai  $K_d$ . Namun semakin nilai *error* mendekati  $1$  maka nilai  $K'_d$  akan hanya menjadi 1.1 kali lipat dari nilai  $K_d$ . Hal tersebut disebabkan oleh non-linearitas sistem yang dikendalikan sehingga untuk perubahan nilai *error* tertentu diperlukan penambahan nilai parameter integral yang sesuai untuk menjaga sistem tetap stabil. Hal ini sesuai dengan penampang permukaan *rule base* untuk menentukan nilai  $dK_d$  seperti tampak pada Gambar 3.18, di mana ketika nilai *error rate* positif mendekati  $0$  dan nilai *error* meningkat dari  $-1$  ke  $1$  maka nilai  $dK_d$  mula-mula akan stagnan di kontur sangat rendah, kemudian meningkat, dan menurun lagi sebelum akhirnya mencapai kontur stagnan berwarna biru tua.

Gambar 3.22 berikut menunjukkan representasi Simulink pada perhitungan nilai  $K_p'$ ,  $K_i'$ , dan  $K_d'$  setelah *tuning* oleh logika *fuzzy*:



**Gambar 3. 23** Skema *Self-Tuning* PID

Dari Gambar 3.23 dapat dilihat bahwa sinyal kontrol yang dihasilkan dapat diperoleh menggunakan persamaan:

$$u(t) = K_p(t)e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} + dK_p(t)e(t) + \int_0^t dK_i(\tau)e(\tau)d\tau + \frac{d[dK_d(t)e(t)]}{dt} \quad (3.66)$$

di mana  $u(t)$  adalah sinyal kontrol, dan  $e(t)$  adalah *error*. Untuk bagian derivatif perlu diperhatikan bahwa keluarannya akan dikalikan dengan  $N$  sebagai *filter coefficient* yang kemudian hasilnya diintegrasikan untuk ditambahkan kembali pada keluaran berikutnya supaya aksi derivatif dapat dilakukan dalam Simulink. Tanpa adanya sistem *feedback* dan integrasi dari perkalian dengan *filter coefficient*, aksi derivatif dalam bentuk kontinu tidak dapat dilakukan dalam Simulink (Goh, 2019).

Untuk kontroler *surge* dan *heave* akan digunakan *fuzzy*-PI; untuk kontroler *pitch* akan digunakan *fuzzy*-PID; sedangkan untuk kontroler *yaw* akan digunakan *fuzzy*-PD. Bagian dari PID yang tidak digunakan pada persamaan di atas akan dianggap 0.

### 3.7 Simulasi dengan Gangguan Arus Laut

Simulasi dengan gangguan arus laut dilakukan untuk menguji performansi dari setiap variasi pengendali *fuzzy*-PID, serta membandingkan performansi dari variasi terbaik dengan pengendali PID biasa. Gangguan arus laut dimodelkan menggunakan persamaan 2.28 – 2.34, dan digabungkan dengan persamaan 2.3, dengan catatan: variabel yang tidak dikontrol secara aktif diabaikan atau dianggap 0, sehingga diperoleh persamaan baru:

$$\begin{bmatrix} u_c \\ v_c \\ w_c \end{bmatrix} = \begin{bmatrix} c \psi c \theta & s \psi c \theta & -s \theta \\ c \psi s \theta - s \psi & s \theta s \psi + c \psi & c \theta \\ s \psi + c \psi s \theta & -c \psi + s \theta s \psi & c \theta \end{bmatrix} \begin{bmatrix} V_c \cos \beta_c \\ V_c \sin \beta_c \\ 0 \end{bmatrix} \quad (3.67)$$

Dalam penelitian ini, *side slip angle*  $\beta_c$  disimulasikan  $120^\circ$  sedangkan nilai kecepatan arus  $V_c$  akan divariasikan untuk mengetahui seberapa *robust* sistem kontrol dan *waypoint following guidance* yang dibangun. Berikut merupakan variasi kecepatan arus  $V_c$  yang disimulasikan, beserta pemodelannya berdasarkan persamaan 2.33:

Iterasi 1: Kecepatan arus 0.2 m/s

$$V_c(t) = 0.2 + t(\omega(t)) \quad (3.68)$$

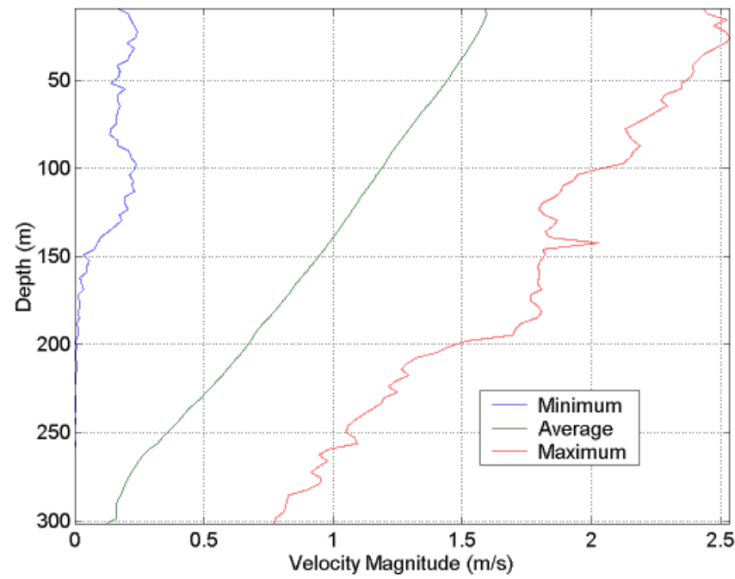
Iterasi 2: Kecepatan arus 40 m/s

$$V_c(t) = 40 + t(\omega(t)) \quad (3.69)$$

Iterasi 3: Kecepatan arus 41 m/s

$$V_c(t) = 41 + t(\omega(t)) \quad (3.70)$$

Pemilihan variasi kecepatan arus laut pada persamaan 3.68 didasarkan pada penelitian oleh Driscoll, et al. (2008) pada Gambar 3.24 terkait klasifikasi kecepatan arus laut pada kedalaman 50 m di lepas pantai Ft. Lauderdale selama 2 tahun pengamatan. Gambar 3.24 menunjukkan bahwa pada kedalaman 50 m, nilai kecepatan arus laut minimum adalah sebesar 0,2 m/s sehingga nilai tersebut dipilih sebagai salah satu variasi. Nilai 40 m/s dan 41 m/s dipilih setelah dilakukan iterasi pada simulasi untuk mencari batas maksimum kecepatan gangguan arus laut ekstrim yang dapat ditolerir oleh salah satu sistem pengendali yang telah dirancang.



**Gambar 3. 24** Kecepatan Arus Laut Minimum, Maksimum dan Rata-rata di Lepas Pantai (Driscoll, et al., 2008)

Nilai posisi  $x$ ,  $y$ , dan  $z$  dalam satuan meter dapat diperoleh dengan mengintegrasikan persamaan berikut:

$$\begin{aligned}\dot{x} &= u_r c \psi c \theta - v_r s \psi + w_r c \psi s \theta \\ \dot{y} &= u_r s \psi c \theta + v_r c \psi + w_r s \psi s \theta \\ \dot{z} &= -u_r s \theta + w_r c \theta\end{aligned}\tag{3.71}$$

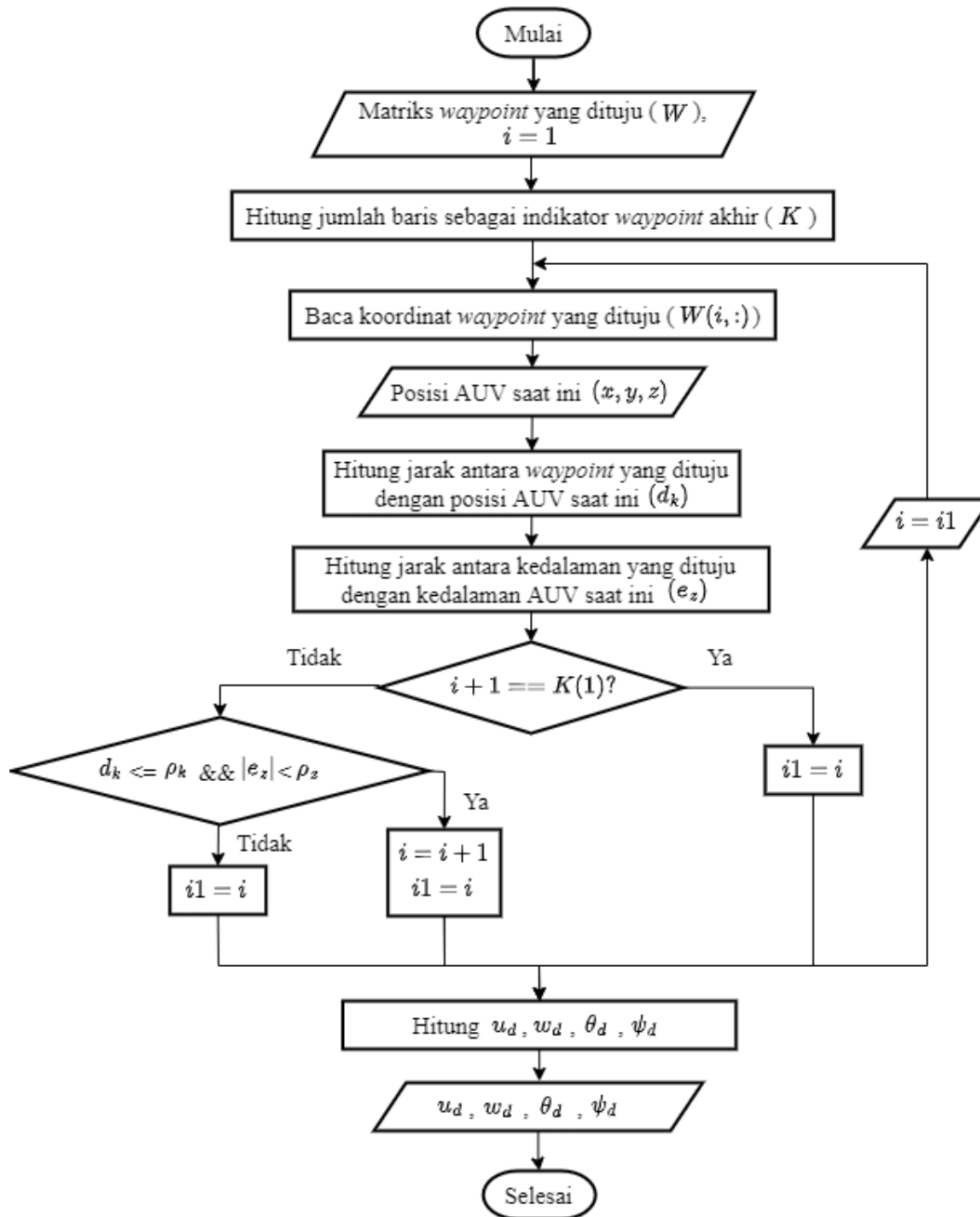
dengan

$$\begin{aligned}u_r &= u - u_c \\ v_r &= v - v_c \\ w_r &= w - w_c\end{aligned}\tag{3.72}$$

di mana  $u$  dan  $w$  diperoleh dari keluaran sistem kontrol *surge* dan *heave*, sedangkan  $v$  diasumsikan 0 karena tidak dikendalikan secara aktif.

### 3.8 Simulasi dengan *Waypoint Following Guidance*

Tahap ini dilakukan dengan mengintegrasikan sistem pengendali yang telah dirancang dengan algoritma *waypoint following guidance* sebagai berikut:



**Gambar 3. 25** Diagram Alir *Waypoint Following Guidance*

Gambar 3.25 menunjukkan bahwa dari algoritma ini ditentukan nilai *setpoint* dari variabel-variabel yang dikendalikan secara aktif, yakni  $u$ ,  $w$ ,  $\theta$ , dan  $\psi$  agar AUV mampu bergerak menuju titik  $x, y, z$  yang diinginkan.

Anggap ada sebuah matriks *waypoint*  $W$  berdimensi  $i \times 3$ , di mana  $i$  menunjukkan jumlah titik yang dituju dalam koordinat  $x, y, z$  yang dimasukkan



oleh *programmer* sebelum AUV dioperasikan. Kolom pertama matriks  $W$  menunjukkan koordinat sumbu  $x$ , kolom kedua menunjukkan koordinat sumbu  $y$ , dan kolom ketiga menunjukkan koordinat sumbu  $z$ . AUV akan bergerak dari titik awal  $[x, y, z]$  menuju *waypoint*  $[x_k, y_k, z_k]$ . Maka berapa derajat AUV harus berbelok pada bidang X-Y dihitung dengan persamaan:

$$\psi_d = \arctan\left(\frac{y_k - y}{x_k - x}\right) \in [-\pi/2, \pi/2] \quad (3.73)$$

Jika AUV sudah memasuki *Circle of Acceptance* (COA) yang merupakan lingkaran maya dengan radius  $\rho_k$  dengan titik pusat  $(x_k, y_k)$ , maka AUV sudah dianggap mencapai target pada bidang X-Y. Jarak antara AUV dengan titik pusat ( $d_k$ ) dihitung dengan persamaan:

$$d_k = \sqrt{(x_k - x)^2 + (y_k - y)^2} \leq \rho_k \quad (3.74)$$

Namun, hanya mengatur arah haluan belum cukup untuk menggerakkan AUV menuju titik yang dituju. AUV juga harus diberi masukan berupa kecepatan pada sumbu  $x$  pada *body frame* atau disebut *surge*. Dalam penelitian ini, *surge* tidak diatur konstan, namun nilai *setpoint*-nya akan berubah secara dinamis sebagai fungsi jarak antara posisi AUV saat ini dengan *waypoint* yang dituju. Semakin mendekati titik yang dituju maka kecepatan AUV akan dibuat melambat. Hal ini bertujuan untuk menghindari penggunaan baterai yang berlebihan (Mendes, 2017). Kecepatan *surge* AUV dapat ditentukan menggunakan persamaan:

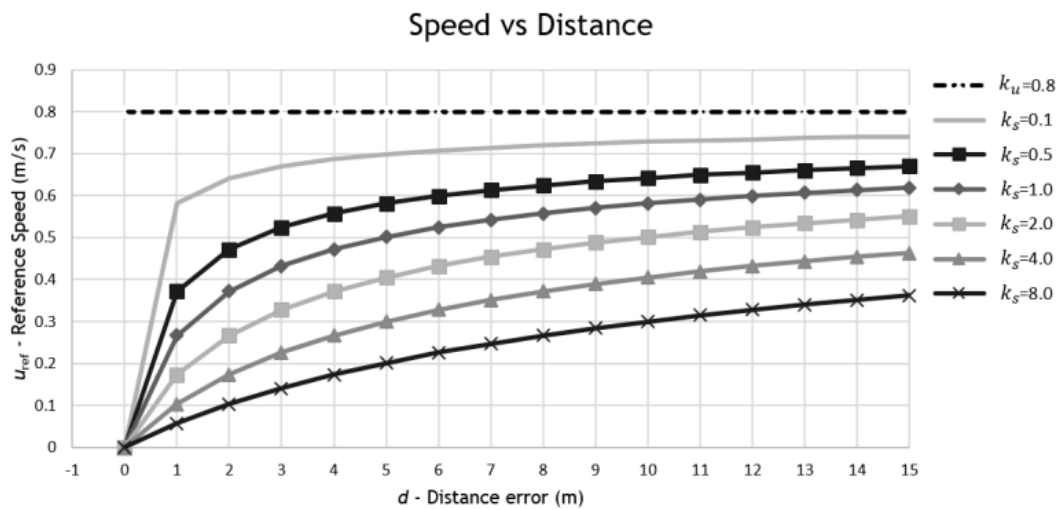
$$u_d = k_u \sin^{-1}\left[\left(\frac{d_u}{|d_u| + k_s}\right) \frac{2}{\pi}\right] \quad (3.75)$$

di mana  $k_u$  adalah batas maksimum kecepatan AUV,  $d_u$  adalah *Circle of Acceptance* (COA) kedua sebagai penentu pada radius berapa AUV harus mulai melambat, di mana nilainya ditentukan menggunakan persamaan:

$$d_u = d_k - \rho_u \quad (3.76)$$

dengan  $\rho_u$  adalah radius dari COA kedua untuk kecepatan. Nilai  $\rho_u$  akan selalu lebih kecil daripada  $\rho_k$ , sehingga nilai  $d_u$  tidak akan 0 dan kecepatan AUV juga tidak akan mencapai 0 kecuali *waypoint* tersebut merupakan *waypoint* terakhir.

Nilai  $k_s$  pada persamaan 3.75 merupakan parameter *tuning* yang nilainya lebih besar dari 0. Semakin besar nilai  $k_s$  maka akan kecepatan AUV akan semakin dijaga untuk menjauhi batas maksimum, seperti ditunjukkan pada Gambar 3.26:



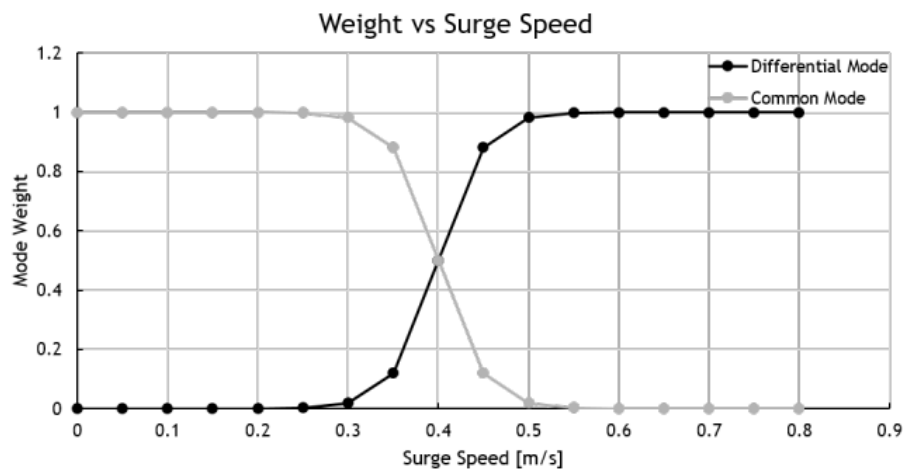
**Gambar 3. 26** Hubungan antara Parameter  $k_s$  dengan Referensi Kecepatan dan Jarak (Mendes, 2017)

Untuk mengatur kedalaman AUV atau posisi AUV pada bidang X-Z, ada dua cara yang dapat dilakukan, yaitu dengan mengatur sudut inklinasi (*pitch*) AUV kemudian menggerakkan AUV dengan kecepatan *surge* pada sumbu  $x$  dari *body frame*, atau mengatur *heave* atau kecepatan pada sumbu  $y$ . Apabila AUV sedang bergerak dengan kecepatan *surge* yang cukup besar maka akan lebih efektif untuk mengatur sudut inklinasi atau *pitch* dari AUV, namun ketika AUV sedang bergerak dalam kecepatan *surge* yang sangat rendah, atau dengan kata lain sudah sangat mendekati COA pada bidang X-Y, maka akan lebih efektif untuk mengatur *heave* dalam mencapai kedalaman yang diinginkan (Mendes, 2017). Untuk menentukan kapan menggunakan *pitch* dan kapan menggunakan *heave* dalam mencapai kedalaman yang diinginkan, dirumuskan parameter pembebanan yakni  $W_{cm}$  dan  $W_{dm}$ , yang ditentukan dengan persamaan berikut:

$$W_{cm} = 1 - \frac{1}{2} \left( \tanh \left( \frac{u - \frac{(u_{cmS} + u_{cmI})}{2}}{\sigma_{cm}^*} \right) + 1 \right) \quad (3.77)$$

$$W_{dm} = s \frac{1}{2} \left( \tanh \left( \frac{u - \frac{(u_{dmS} + u_{dmI})}{2}}{\sigma_{dm}^*} \right) + 1 \right) \quad (3.78)$$

di mana  $u_{cmS}$ ,  $u_{cmI}$ ,  $u_{dmS}$ , dan  $u_{dmI}$  adalah batas atas dan batas bawah kecepatan transisional dari mode menggunakan *heave* ke mode menggunakan *pitch*, sedangkan  $\sigma_{cm}^*$  dan  $\sigma_{dm}^*$  menentukan kecuraman kurva seperti gambar berikut:



**Gambar 3. 27** Fungsi Mode Pembebanan dengan  $u_{cmS} = u_{dmS} = 0.5 \frac{m}{s}$ ;  $u_{cmI} = u_{dmI} = 0.3$ ; dan  $\sigma_{cm}^* = \sigma_{dm}^* = 0.05$  (Mendes, 2017)

Nilai *pitch* sebagai *setpoint* ditentukan dengan persamaan:

$$\theta_d = \arctan \left( \frac{z_k - z}{x_k - x} \right) \quad (3.79)$$

Sedangkan untuk *heave* ditentukan menggunakan persamaan:

$$w_d = k_w \sin^{-1} \left[ \left( \frac{e_z}{|e_z| + k_s} \right) \frac{2}{\pi} \right] \quad (3.80)$$

di mana  $k_w$  merupakan batas maksimum kecepatan vertikal (*heave*),  $k_s$  adalah parameter *tuning*, dan  $e_z$  adalah selisih antara kedalaman *waypoint* yang dituju dengan kedalaman AUV saat ini, atau dituliskan dengan persamaan:

$$e_z = z_k - z \quad (3.81)$$

Akhirnya, apabila COA pada bidang X-Y dan X-Z digabungkan, maka syarat yang harus dipenuhi pada algoritma *waypoint following* untuk beralih ke titik berikutnya adalah:

$$|z_k - z| < \varepsilon_r \wedge d_k = \sqrt{(x_k - x)^2 + (y_k - y)^2} \leq \rho_k \quad (3.82)$$

di mana  $\varepsilon_r$  adalah *error* kedalaman maksimum yang diterima (Mendes, 2017).

Koordinat *waypoint* yang dituju dalam simulasi dapat dilihat pada Tabel 3.10 berikut:

**Tabel 3. 10** Koordinat *Waypoint* yang Dituju dalam Simulasi

<i>Waypoint ke-i</i>	<i>x</i> (meter)	<i>y</i> (meter)	<i>z</i> (meter)
1	10	20	-20
2	30	20	-20
3	50	40	-20

Berikut parameter simulasi yang digunakan:

**Tabel 3. 11** Parameter Simulasi *Waypoint Following Guidance* (Mendes, 2017)

Notasi	Nilai	Satuan
$\rho_u$	1	m
$\rho_k$	1.5	m
$\rho_z$	0.5	m
$k_u$	0.8	m/s
$k_w$	0.4	m/s
$\eta_1 = (x_0, y_0, z_0)$	(0,0,0)	m

### 3.9 Analisa Perbandingan Hasil Simulasi PID dan *Fuzzy*-PID

Performansi sistem pengendali PID dan *fuzzy*-PID yang telah dirancang akan dibandingkan berdasarkan uji *closed loop* dengan gangguan sebelum sistem pengendali diintegrasikan dengan algoritma *waypoint following guidance* sehingga *setpoint* dapat ditentukan secara manual terlebih dahulu, dan uji simulasi *waypoint*

*following* di mana sistem pengendali diintegrasikan dengan algoritma *waypoint following guidance* supaya mampu menentukan *setpoint* secara otomatis baik di bawah gangguan arus laut maupun tanpa gangguan arus laut.

Untuk uji *closed loop* dengan gangguan arus laut, performansi PID dan *fuzzy* PID akan dinilai berdasarkan *steady state error*, *settling time*, dan *overshoot*, sedangkan untuk simulasi menggunakan *waypoint following* akan dianalisa secara kualitatif menggunakan kurva yang diperoleh dan berhasil atau tidaknya AUV mencapai *circle of acceptance* yang telah ditentukan.

*Halaman ini sengaja dikosongkan*

## BAB IV

### HASIL DAN PEMBAHASAN

#### 4.1 Analisa Hasil Perancangan Sistem Pengendali PID

Perancangan sistem pengendali PID dibagi ke dalam tiga tahap, yakni: uji *open loop*, *tuning* parameter PID, dan uji stabilitas menggunakan diagram Bode. Uji *open loop* bertujuan untuk mengetahui respon sistem sebelum diberikan sistem pengendali. *Tuning* parameter PID dilakukan dengan menggunakan *toolbox* PID *Tuner* pada Simulink untuk mendapatkan nilai awal, kemudian parameter disesuaikan secara manual untuk mendapatkan respons waktu, *transient behavior*, *overshoot*, dan *settling time* yang diinginkan. Berikut merupakan daftar nilai akhir parameter PID dan *gain anti-windup* yang diperoleh dari proses *tuning* parameter masing-masing subsistem:

**Tabel 4. 1** Parameter PID dan *Gain Anti-Windup*

Variabel	Tipe	$K_p$	$K_i$	$K_d$	$N$	$1/T_t$
<i>Surge</i>	PI	4380	738.6141	-	-	0.21
<i>Heave</i>	PI	12729.55	3380.068	-	-	0.3319
<i>Pitch</i>	PID	15568.77	10,000	12019.09	4624.512	0.331
<i>Yaw</i>	PD	6580.602	-	10000	2209.731	-

Keterangan:

$K_p$  = Parameter *gain* proporsional

$K_i$  = Parameter *gain* integral

$K_d$  = Parameter *gain* derivatif

$N$  = *Filter coefficient*

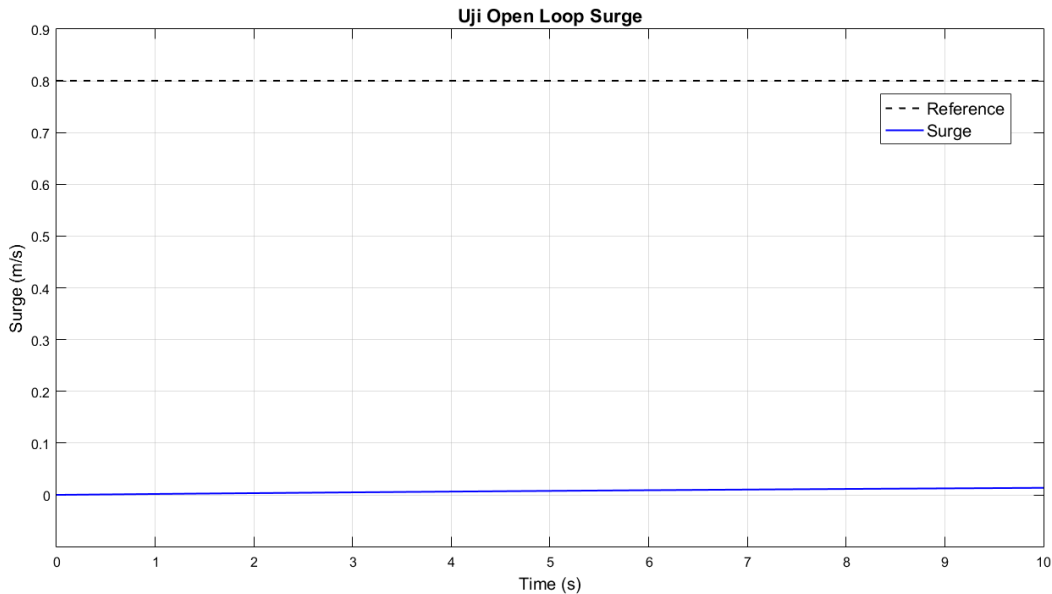
$T_t$  = *Time constant* untuk menghitung *gain anti-windup*

Akhirnya, uji stabilitas dilakukan untuk menguji kestabilan masing-masing sistem pengendali apabila digunakan parameter *gain* PID yang telah diperoleh pada tahap *tuning*.

Berikut merupakan hasil dan analisa dari perancangan sistem pengendali PID untuk ketiga subsistem dan keempat kontroler:

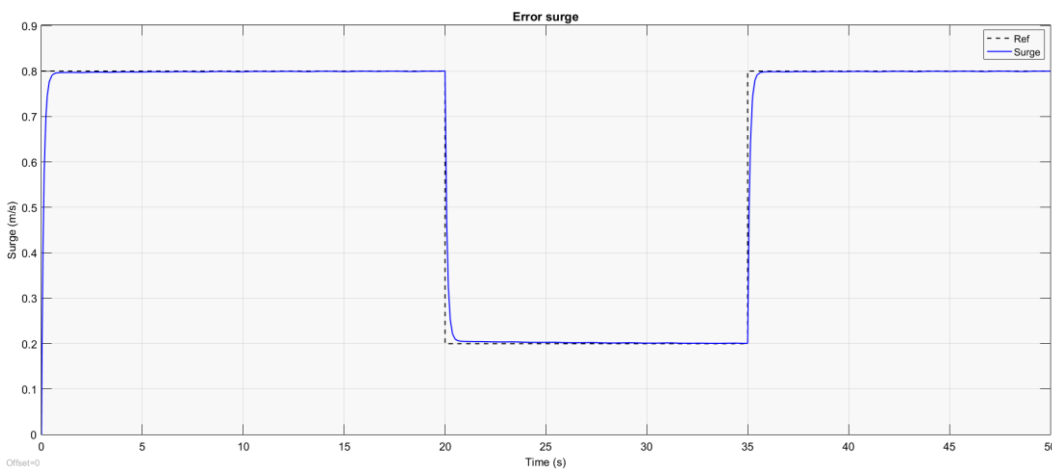
#### 4.1.1 Subsistem Pengendali Kecepatan (*Speed*)

Gambar 4.1 menunjukkan respon subsistem pengendali variabel *surge* sebelum diberikan kontroler PI.



**Gambar 4. 1** Respon Uji *Open Loop* pada Subsistem Pengendali *Surge*

Dari Gambar 4.1 dapat dilihat bahwa tanpa adanya kontroler, nilai keluaran hanya akan meningkat dalam jumlah yang kecil dan lambat, bahkan tidak dapat mendekati nilai *setpoint* yang ditentukan, yakni 0.8 m/s, dalam waktu 10 detik.

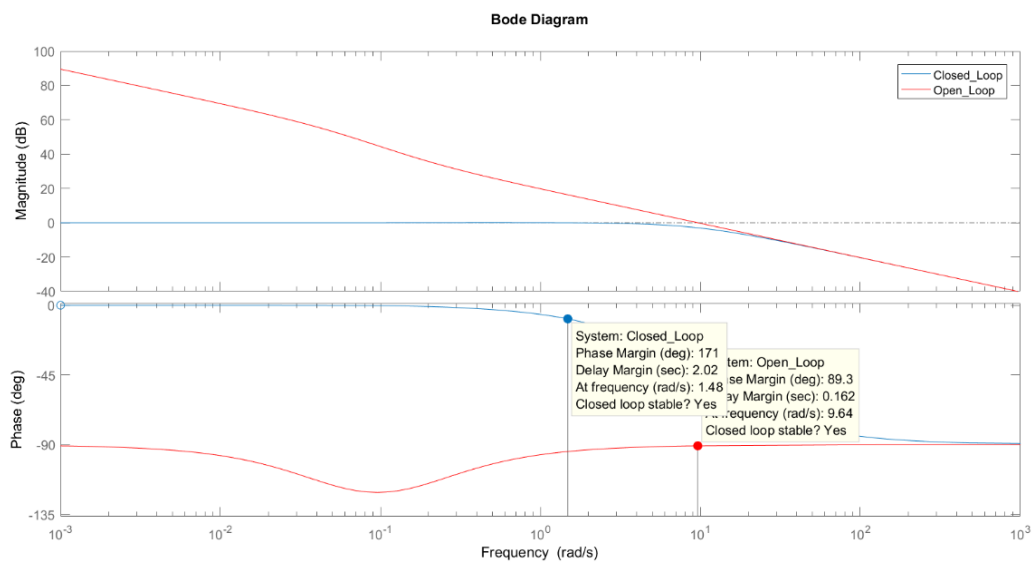


**Gambar 4. 2** Uji *Closed Loop* pada Subsistem Pengendali *Surge*



Gambar 4.2 menunjukkan respon sistem setelah diberi kontroler PI dengan nilai parameter *gain* proporsional dan integral pada Tabel 4.1. Dari Gambar 4.2 dapat dilihat bahwa subsistem pengendali *surge* dengan kontroler PI yang telah dirancang mampu untuk mencapai nilai *setpoint* yang ditentukan, bahkan dalam waktu kurang dari 10 detik.

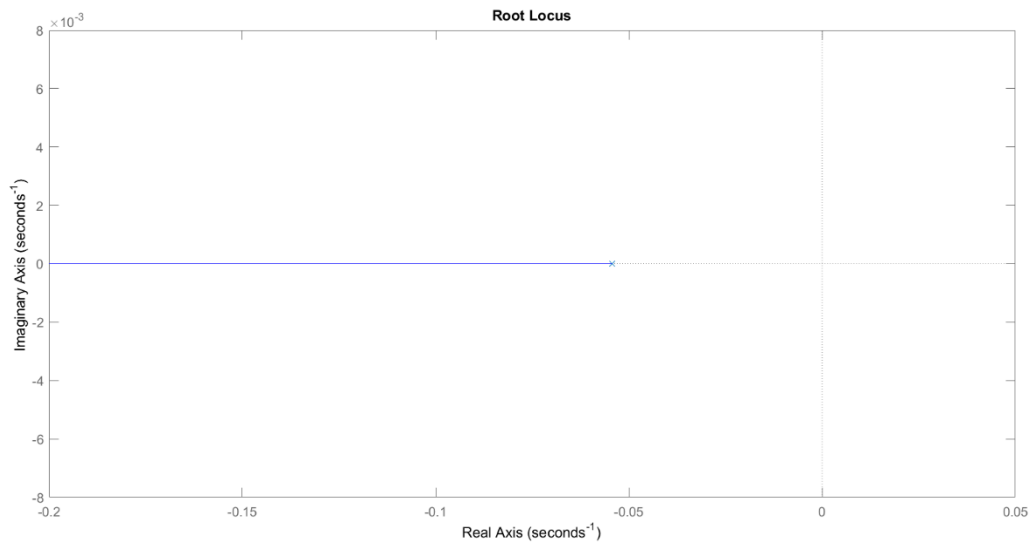
Untuk membuktikan kestabilan dari parameter kontroler PI yang telah ditetapkan seperti pada Tabel 4.1, dilakukan uji diagram Bode. Berikut merupakan diagram Bode yang dihasilkan:



**Gambar 4.3** Diagram Bode Uji Stabilitas Subsistem Pengendali *Surge*

Gambar 4.3 menunjukkan bahwa parameter *gain* kontroler PI yang dipilih memenuhi kriteria kestabilan diagram Bode seperti yang dicantumkan dalam Dasar Teori, yakni memiliki sistem *open loop* yang stabil dan nilai *phase margin* 89.3 derajat dan *gain margin* yang bernilai *infinity*. Nilai *gain margin* sebesar *infinity* juga diperkuat dengan analisa *root locus* dari fungsi transfer *surge* yang telah dirumuskan pada persamaan 3.40.

Gambar 4.4 berikut merupakan gambar kurva *root locus* dari fungsi transfer *surge* yang telah dirumuskan pada persamaan 3.40:



**Gambar 4. 4** Diagram *Root Locus* Subsistem Pengendali *Surge*

Gambar 4.4 menunjukkan bahwa kurva *root locus* dari fungsi transfer *surge* tidak ada yang berpotongan dengan sumbu *real axis* = 0 sehingga nilai *gain margin* dari subsistem *surge* adalah *infinity*, yang mana sistem akan stabil untuk berapapun nilai *gain* yang digunakan.

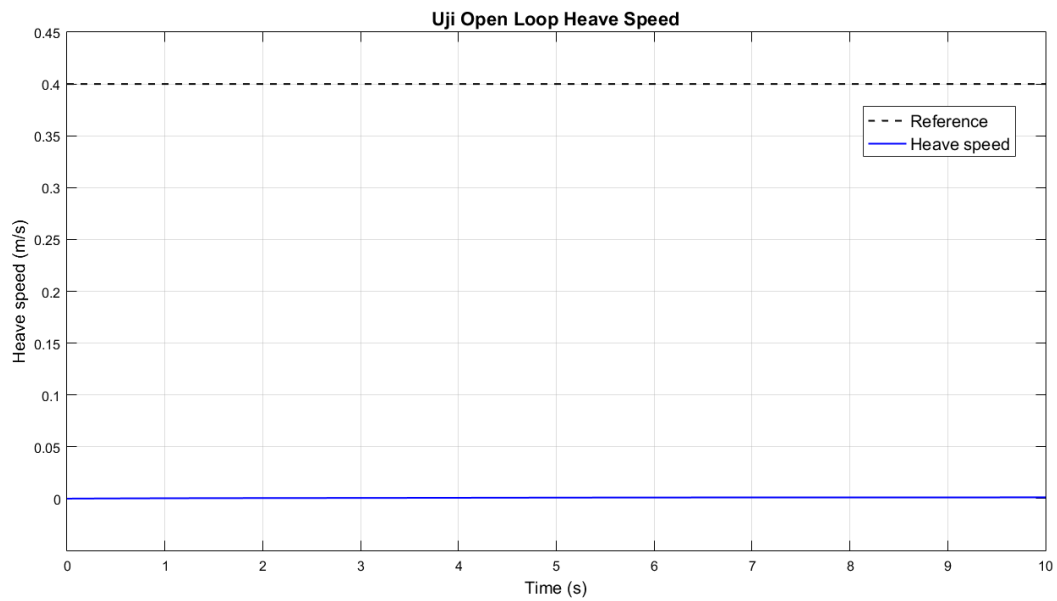
#### 4.1.2 Subsistem Pengendali Kedalaman (*Depth*)

Sesuai perancangan subsistem pengendali kedalaman yang telah dijelaskan pada Metodologi, subsistem pengendali kedalaman dipecah menjadi 2, yakni dengan mengendalikan *pitch* dan *heave*.

##### 4.1.2.1 Pengendali *Heave*

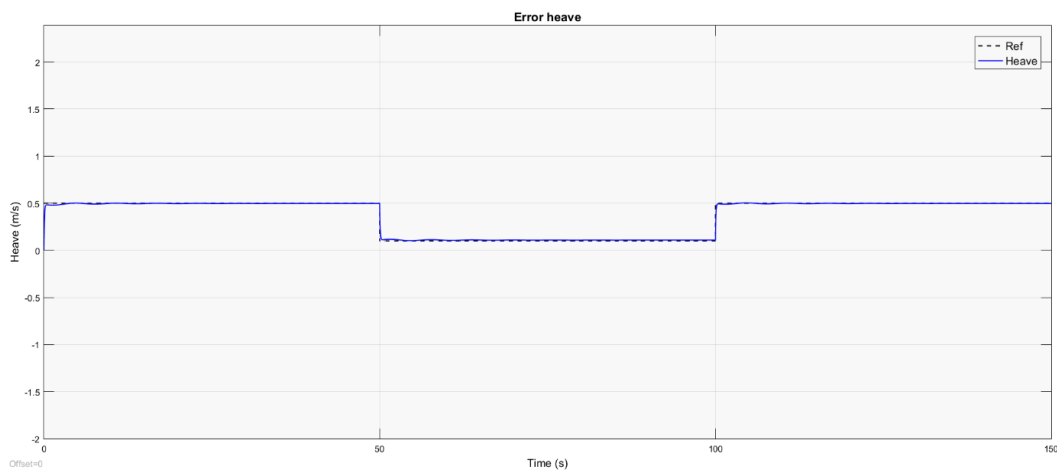
Gambar 4.5 menunjukkan uji *open loop* pada subsistem pengendali variabel *heave* untuk meninjau respon subsistem pengendali variabel *heave* sebelum diberikan kontroler PI.

Dari Gambar 4.5 dapat dilihat bahwa tanpa adanya kontroler, nilai keluaran hanya akan meningkat dalam jumlah yang kecil dan lambat, bahkan tidak dapat mendekati nilai *setpoint* yang ditentukan, yakni 0.4 m/s, dalam waktu 10 detik.



**Gambar 4. 5** Respon Uji *Open Loop* pada Subsisitem Pengendali *Heave*

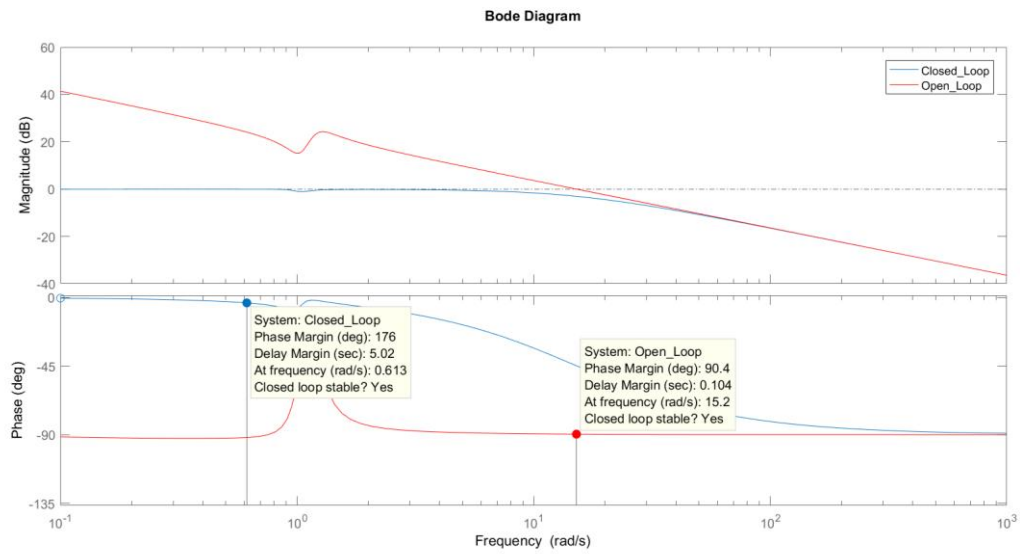
Setelah diberi kontroler PI dengan nilai parameter *gain* proporsional dan integral pada Tabel 4.1, diperoleh respon sistem sebagai berikut:



**Gambar 4. 6** Uji *Closed Loop* pada Subsisitem Pengendali *Heave*

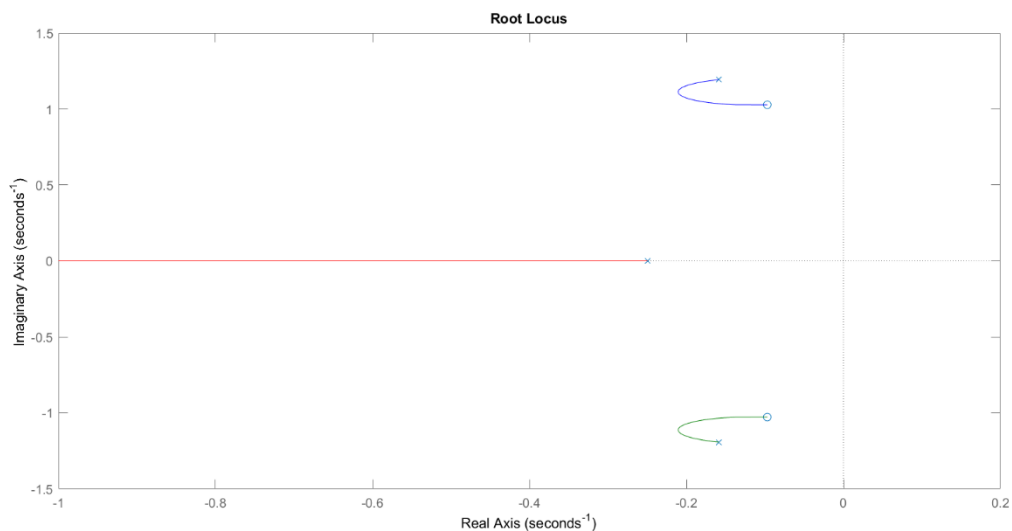
Dari Gambar 4.6 dapat dilihat bahwa subsistem pengendali *heave* dengan kontroler PI yang telah dirancang mampu untuk mencapai nilai *setpoint* yang ditentukan, bahkan dalam waktu kurang dari 10 detik.

Untuk membuktikan kestabilan dari parameter kontroler PI yang telah ditetapkan seperti pada Tabel 4.1, dilakukan uji diagram Bode. Berikut merupakan diagram Bode yang dihasilkan:



**Gambar 4. 7** Diagram Bode Uji Stabilitas Subsystem Pengendali *Heave*

Gambar 4.7 menunjukkan bahwa parameter *gain* kontroler PI yang dipilih memenuhi kriteria kestabilan diagram Bode seperti yang dicantumkan dalam Dasar Teori, yakni memiliki sistem *open loop* yang stabil dan nilai *phase margin* 90.4 derajat dan *gain margin* yang bernilai *infinity*. Nilai *gain margin* sebesar *infinity* juga diperkuat dengan analisa *root locus* dari fungsi transfer *heave* yang telah dirumuskan pada persamaan 3.48.

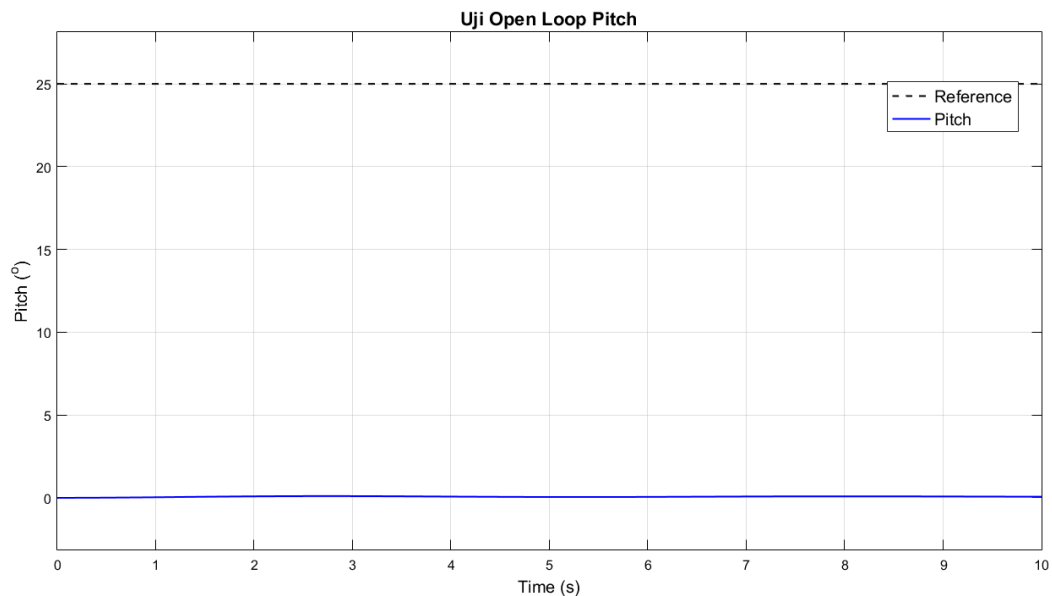


**Gambar 4. 8** Diagram *Root Locus* Subsystem Pengendali *Heave*

Gambar 4.8 menunjukkan bahwa kurva *root locus* dari fungsi transfer *heave* tidak ada yang berpotongan dengan sumbu *real axis* = 0 sehingga nilai *gain margin* dari subsistem *heave* adalah *infinity*, yang mana sistem akan stabil untuk berapapun nilai *gain* yang digunakan.

#### 4.1.2.2 Pengendali *Pitch*

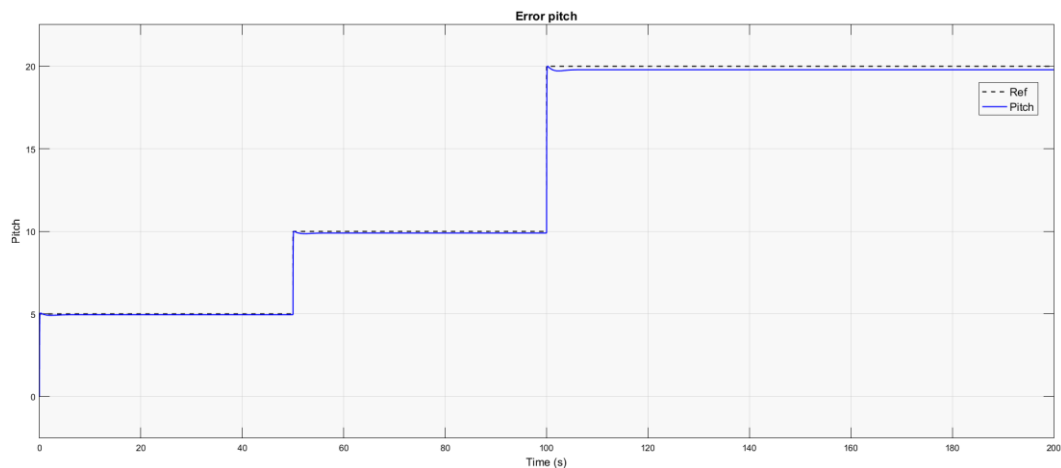
Gambar 4.9 menunjukkan respon subsistem pengendali variabel *pitch* sebelum diberikan kontroler PID.



**Gambar 4.9** Respon Uji *Open Loop* pada Subsistem Pengendali *Pitch*

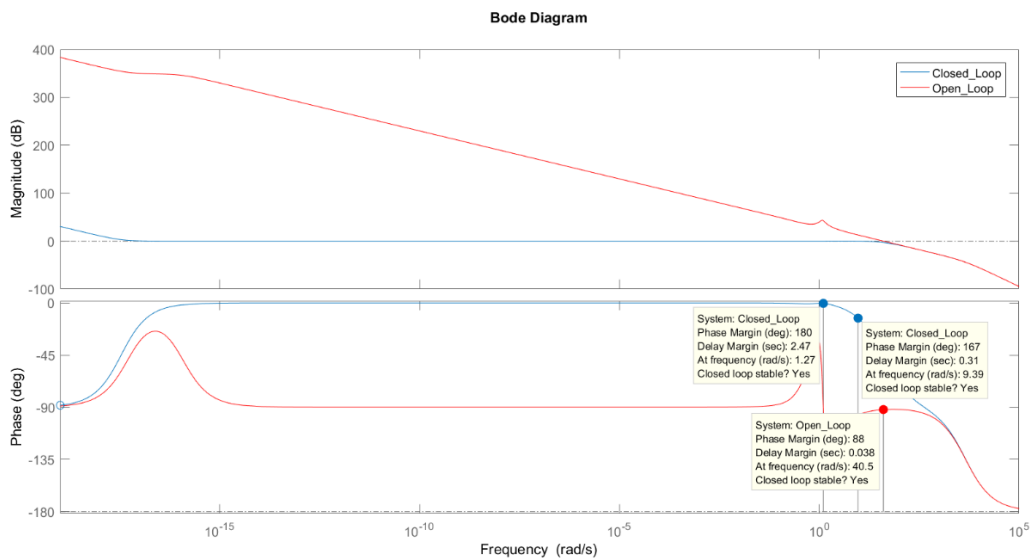
Dari Gambar 4.9 dapat dilihat bahwa tanpa adanya kontroler, nilai keluaran hanya akan meningkat dalam jumlah yang kecil dan lambat, bahkan tidak dapat mendekati nilai *setpoint* yang ditentukan, yakni  $25^{\circ}$ , dalam waktu 10 detik.

Gambar 4.10 di bawah ini menunjukkan respons sistem setelah diberi kontroler PID dengan nilai parameter *gain* proporsional, integral, dan derivatif pada Tabel 4.1. Dari Gambar 4.10 dapat dilihat bahwa subsistem pengendali *pitch* dengan kontroler PID yang telah dirancang mampu untuk mencapai nilai *setpoint* yang ditentukan, bahkan dalam waktu kurang dari 10 detik, meskipun seiring bertambahnya waktu *error steady state* bertambah karena adanya nonlinearitas sistem (Bolton W. , 2021).



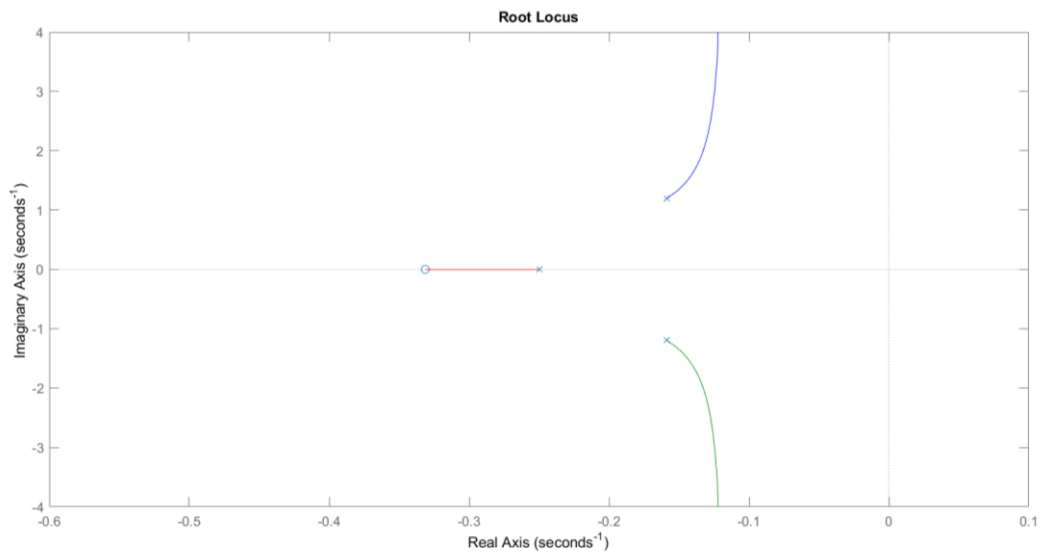
**Gambar 4. 10** Uji *Closed Loop* pada Subsistem Pengendali *Pitch*

Untuk membuktikan kestabilan dari parameter kontroler PID yang telah ditetapkan seperti pada Tabel 4.1, dilakukan uji diagram Bode. Berikut merupakan diagram Bode yang dihasilkan:



**Gambar 4. 11** Diagram Bode Uji Stabilitas Subsistem Pengendali *Pitch*

Gambar 4.11 menunjukkan bahwa parameter *gain* kontroler PID yang dipilih memenuhi kriteria kestabilan diagram Bode seperti yang dicantumkan dalam Dasar Teori, yakni memiliki sistem *open loop* yang stabil dan nilai *phase margin* 88 derajat dan *gain margin* yang bernilai *infinity*. Nilai *gain margin* sebesar *infinity* juga diperkuat dengan analisa *root locus* dari fungsi transfer *pitch* yang telah dirumuskan pada persamaan 3.52.

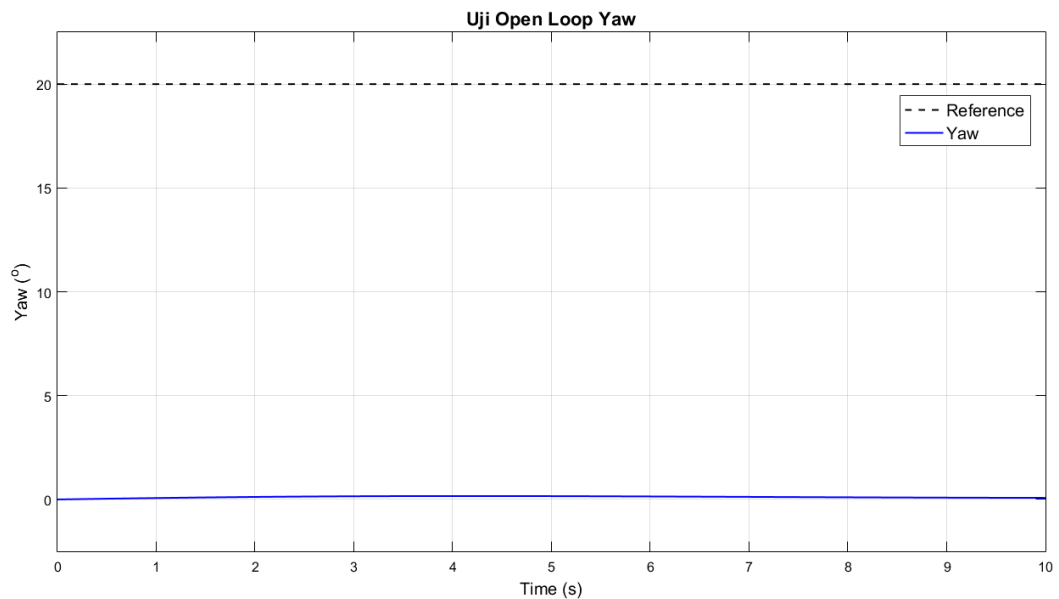


**Gambar 4. 12** Diagram *Root Locus* Subsystem Pengendali *Pitch*

Gambar 4.12 menunjukkan bahwa kurva *root locus* dari fungsi transfer *pitch* tidak ada yang berpotongan dengan sumbu *real axis* = 0 sehingga nilai *gain margin* dari subsystem *pitch* adalah *infinity*, yang mana sistem akan stabil untuk berapapun nilai *gain* yang digunakan.

#### 4.1.3 Subsystem Pengendali Haluan (*Steering*)

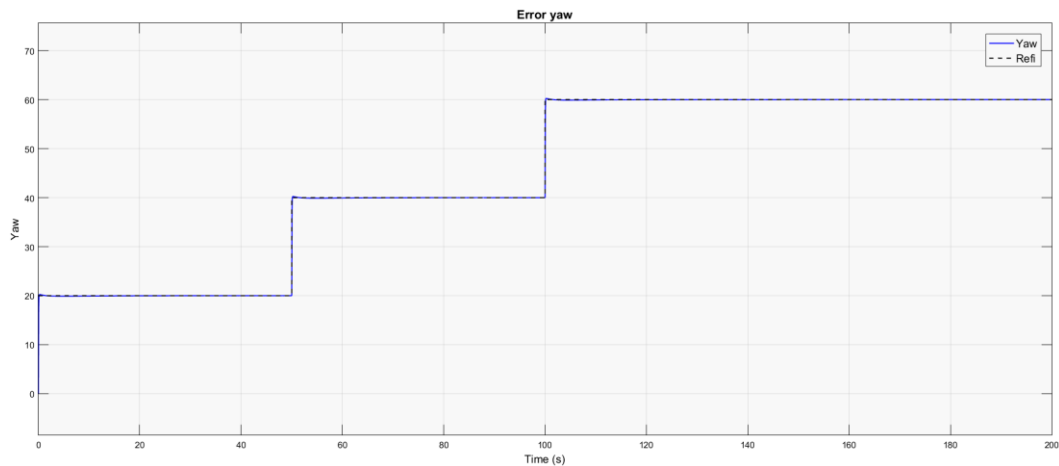
Gambar 4.13 menunjukkan respon subsystem pengendali variabel *yaw* sebelum diberikan kontroler PD.



**Gambar 4. 13** Respon Uji *Open Loop* pada Subsystem Pengendali *Yaw*

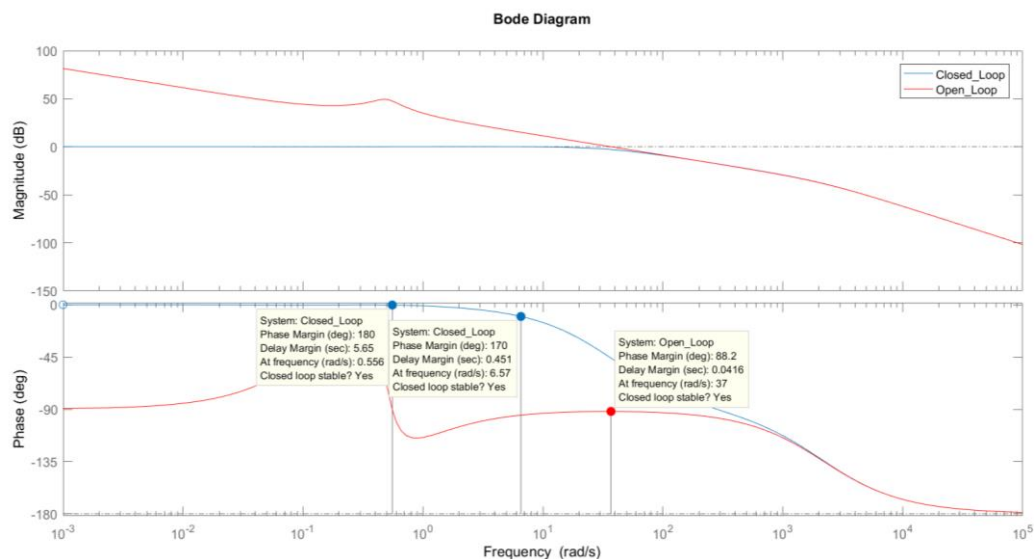
Dari Gambar 4.13 dapat dilihat bahwa tanpa adanya kontroler, nilai keluaran hanya akan meningkat dalam jumlah yang kecil dan lambat, bahkan tidak dapat mendekati nilai *setpoint* yang ditentukan, yakni  $20^\circ$ , dalam waktu 10 detik.

Setelah diberi kontroler PD dengan nilai parameter *gain* proporsional dan derivatif pada Tabel 4.1, diperoleh respon sistem sebagai berikut:



**Gambar 4. 14** Uji *Closed Loop* pada Subsistem Pengendali *Yaw*

Dari Gambar 4.14 dapat dilihat bahwa subsistem pengendali *yaw* dengan kontroler PD yang telah dirancang mampu untuk mencapai nilai *setpoint* yang ditentukan, bahkan dalam waktu kurang dari 10 detik.

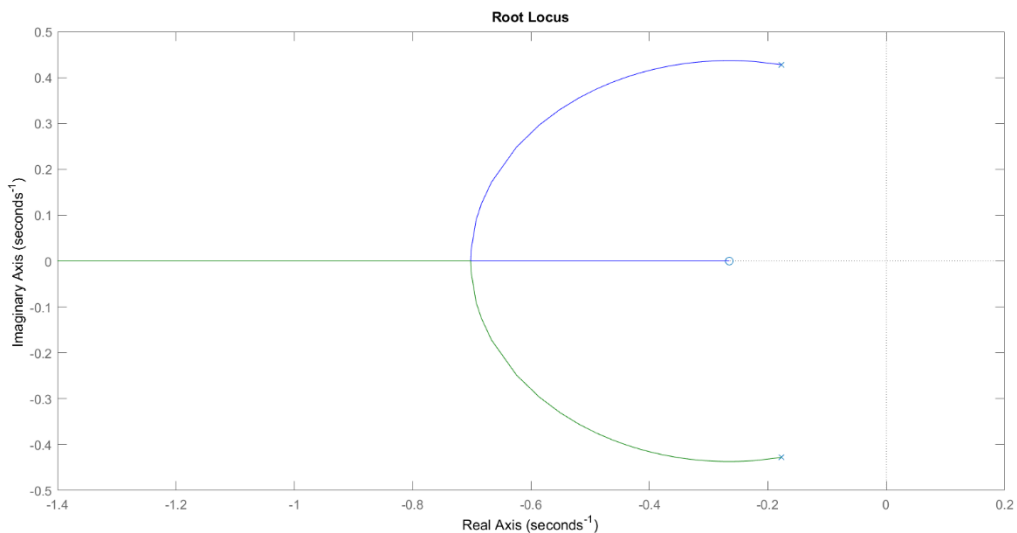


**Gambar 4. 15** Diagram Bode Uji Stabilitas Subsistem Pengendali *Yaw*



Untuk membuktikan kestabilan dari parameter kontroler PD yang telah ditetapkan seperti pada Tabel 4.1, dilakukan uji diagram Bode, seperti tampak dalam Gambar 4.15 di atas.

Gambar 4.15 menunjukkan bahwa parameter *gain* kontroler PD yang dipilih memenuhi kriteria kestabilan diagram Bode seperti yang dicantumkan dalam Dasar Teori, yakni memiliki sistem *open loop* yang stabil dan nilai *phase margin* 88.2 derajat dan *gain margin* yang bernilai *infinity*. Nilai *gain margin* sebesar *infinity* juga diperkuat dengan analisa *root locus* dari fungsi transfer *yaw* yang telah dirumuskan pada persamaan 3.57.



**Gambar 4. 16** Diagram *Root Locus* Subsistem Pengendali *Yaw*

Gambar 4.16 menunjukkan bahwa kurva *root locus* dari fungsi transfer *yaw* tidak ada yang berpotongan dengan sumbu *real axis* = 0 sehingga nilai *gain margin* dari subsistem *yaw* adalah *infinity*, yang mana sistem akan stabil untuk berapapun nilai *gain* yang digunakan.

## 4.2 Analisa Hasil Perancangan Sistem Pengendali *Fuzzy-PID*

Perancangan sistem pengendali *fuzzy-PID* terbagi ke dalam dua tahap, yakni: menentukan *range* nilai *error* dan *error rate* untuk menormalisasi masukan pada masing-masing subsistem supaya masukan pada logika *fuzzy* untuk tiap kontroler bisa disamakan rentangnya menjadi antara -1 dan 1; serta iterasi dengan variasi

*range tuning* parameter *gain* proporsional ( $dK_p$ ), integral ( $dK_i$ ), atau derivatif ( $dK_d$ ) sesuai dengan Tabel 3.6 untuk mendapatkan *range tuning* dengan performansi terbaik di bawah gangguan.

Berikut merupakan hasil perhitungan *range* nilai *error* dan *error rate* yang diperoleh berdasarkan kapasitas AUV:

**Tabel 4. 2** *Range Error* dan *Error Rate* untuk Menormalisasi Masukan Logika Fuzzy

	<i>Range Error</i>	<i>Range Error Rate</i>
<i>Surge</i>	[-0.8 0.8] → 0.8	[-3.518e+12 3.518e+12] → 3.518e+12
<i>Heave</i>	[-0.4 0.4] → 0.4	[-8.796e+11 8.796e+11] → 8.796e+11
<i>Pitch</i>	[-25 25] → 25	[-5.498e+13 5.498e+13] → 5.498e+13
<i>Yaw</i>	[-180 180] → 180	[-3.958e+14 3.958e+14] → 3.958e+14

Berikut merupakan data dan analisa data yang diperoleh dari iterasi variasi *range tuning* parameter *gain* proporsional ( $dK_p$ ), integral ( $dK_i$ ), atau derivatif ( $dK_d$ ) dari masing-masing pengendali:

#### 4.2.1 Subsistem Pengendali Kecepatan (*Speed*)

Data nilai parameter *gain* proporsional dan integral pada kontroler PI *surge* sebelum dimodifikasi dengan *fuzzy*, variasi nilai *range*, dan performansi dari setiap iterasi dapat dilihat pada tabel di bawah ini:

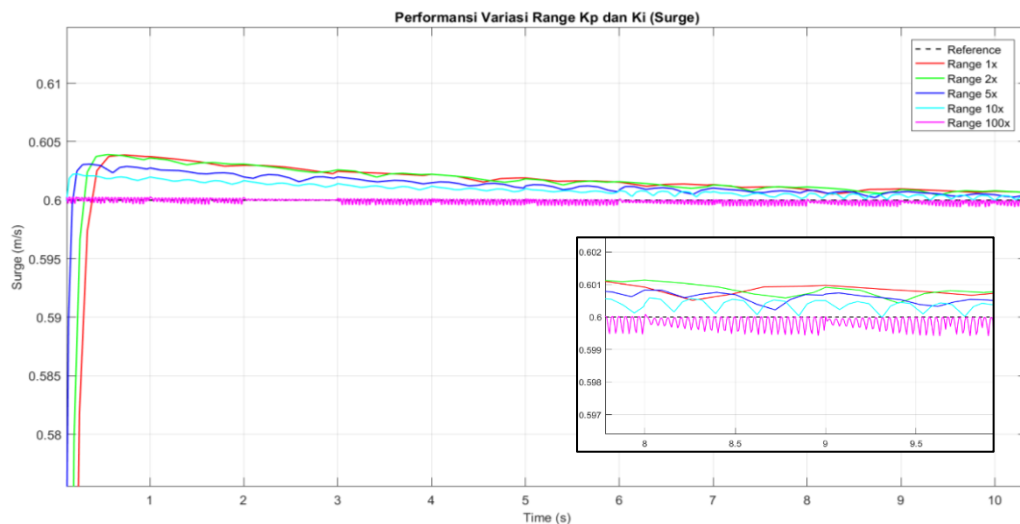
**Tabel 4. 3** Data Iterasi Variasi *Range*  $dK_p$  dan  $dK_i$  Kontroler Fuzzy-PI *Surge*

Iterasi ke-	<i>Range</i> $dK_p$	<i>Range</i> $dK_i$	$T_r$	$T_s$	$T_{s\ min}$	$T_{s\ max}$	%OS	%US	$P$	$T_p$	$e_{ss}$
1	4379.52	738.61	0.17	0.28	0.58	0.60	0.65	0.00	0.60	0.72	4.45E-05
2	8759.04	1477.23	0.14	0.22	0.58	0.60	0.65	0.00	0.60	0.55	1.93E-05
3	21897.60	3693.07	0.08	0.13	0.55	0.60	0.52	0.00	0.60	0.36	6.45E-05
4	43795.20	7386.14	0.05	0.08	0.57	0.60	0.41	0.00	0.60	0.23	2.02E-04
5	437951.96	73861.41	0.01	0.01	0.58	0.60	0.07	0.00	0.60	0.02	9.14E-05

Dari Tabel 4.3 tampak bahwa nilai  $e_{ss}$  terkecil dimiliki oleh iterasi ke-2 dengan *range tuning* 2x parameter *gain* PI, namun meskipun perhitungan  $e_{ss}$  menggunakan MATLAB dapat memberikan gambaran terkait performansi kontroler saat *steady*, kurang tepat apabila hanya mempertimbangkan nilai  $e_{ss}$

untuk memilih *range* parameter terbaik. Hal ini juga dikarenakan metode perhitungan  $e_{ss}$  pada MATLAB yang hanya menghitung selisih antara nilai keluaran dengan *setpoint* di akhir waktu simulasi, sehingga kurang tepat untuk respons sistem yang beresilasi terhadap waktu. Sebaliknya, %OS terminimum dimiliki oleh iterasi ke-5.

Maka dari itu dibuat perbandingan plot respons sistem terhadap waktu dari masing-masing iterasi variasi *range tuning* parameter *gain* proporsional ( $dK_p$ ), integral ( $dK_i$ ), atau derivatif ( $dK_d$ ) untuk kontroler PI *surge* di bawah gangguan untuk pertimbangan tambahan, sebagai berikut:



**Gambar 4. 17** Respons Sistem terhadap Waktu – Iterasi Variasi *Range*  $dK_p$  dan  $dK_i$  Kontroler *Fuzzy-PI Surge*

Gambar 4.17 menunjukkan bahwa, meskipun beresilasi, iterasi ke-5 dengan nilai *range tuning* parameter *gain* proporsional ( $dK_p$ ) dan integral ( $dK_i$ ) sebesar 100 kali dari parameter *gain* proporsional dan integral dari kontroler PI *surge* sebelum di-*tuning* dengan logika *fuzzy* mampu mencapai *setpoint* yang ditentukan berkali-kali dalam kurun waktu yang sama apabila dibandingkan dengan iterasi lainnya. Selain itu, pada Tabel 4.3 juga dapat dilihat bahwa iterasi ke-5 memiliki %OS terminimum sehingga dapat disimpulkan bahwa iterasi ke-5 merupakan *range* terbaik.

## 4.2.2 Subsistem Pengendali Kedalaman (*Depth*)

Subsistem pengendali kedalaman dibagi lagi menjadi dua yakni pengendali *heave* dan pengendali *pitch*.

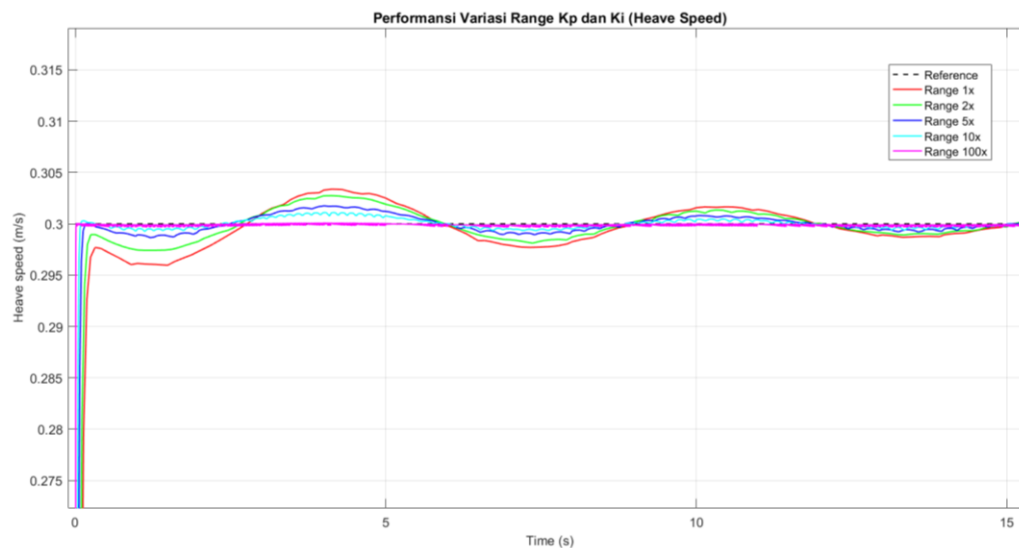
### 4.2.2.1 Pengendali *Heave*

Data nilai parameter *gain* proporsional dan integral pada kontroler PI *heave* sebelum dimodifikasi dengan *fuzzy*, variasi nilai *range*, dan performansi dari setiap iterasi dapat dilihat pada tabel di bawah ini:

**Tabel 4. 4** Data Iterasi Variasi *Range*  $dK_p$  dan  $dK_i$  Kontroler *Fuzzy-PI Heave*

Iterasi ke-	<i>Range</i> $dK_p$	<i>Range</i> $dK_i$	$T_r$	$T_s$	$T_s \text{ min}$	$T_s \text{ max}$	%OS	%US	$P$	$T_p$	$e_{ss}$
1	12729.55	3380.07	0.11	0.20	0.28	0.30	1.14	0.00	0.30	4.13	5.43E-05
2	25459.10	6760.14	0.09	0.15	0.28	0.30	0.94	0.00	0.30	4.12	7.86E-05
3	63647.74	16900.34	0.05	0.09	0.29	0.30	0.59	0.00	0.30	4.00	2.88E-05
4	127295.48	33800.68	0.03	0.06	0.29	0.30	0.38	0.00	0.30	4.00	2.61E-05
5	1272954.83	338006.75	0.00	0.01	0.27	0.30	0.05	0.00	0.30	4.00	2.05E-05

Dari Tabel 4.4 tampak bahwa nilai  $e_{ss}$  dan %OS terkecil dimiliki oleh iterasi ke-5. Untuk memastikan performansi iterasi ke-5 adalah yang terbaik, ditinjau plot perbandingan respons sistem terhadap waktu sebagai berikut:



**Gambar 4. 18** Respons Sistem terhadap Waktu – Iterasi Variasi *Range*  $dK_p$  dan  $dK_i$  Kontroler *Fuzzy-PI Heave*

Gambar 4.18 menunjukkan bahwa respons sistem terbaik dimiliki oleh iterasi ke-5 dengan nilai *range tuning* parameter *gain* proporsional ( $dK_p$ ) dan integral ( $dK_i$ ) sebesar 100 kali dari parameter *gain* proporsional dan integral dari kontroler PI *heave* sebelum di-*tuning* dengan logika *fuzzy*. Didukung dengan nilai  $e_{ss}$  dan %OS dari iterasi ke-5 pada Tabel 4.4, dapat disimpulkan bahwa *range* terbaik untuk kontroler *fuzzy* PI adalah 100 kali parameter *gain* PI *heave*.

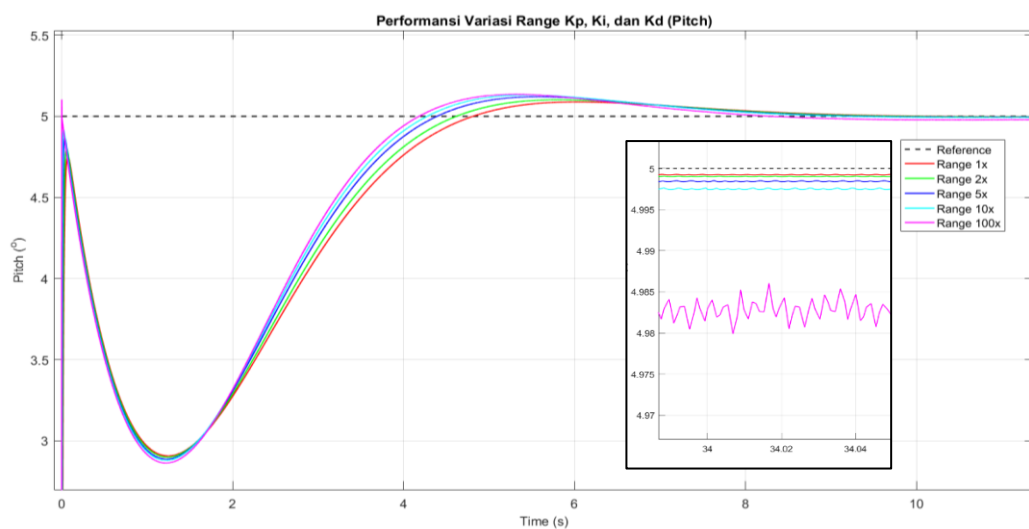
#### 4.2.2.2 Pengendali Pitch

Data nilai parameter *gain* proporsional, integral, dan derivatif pada kontroler PID *pitch* sebelum dimodifikasi dengan *fuzzy*, variasi nilai *range*, dan performansi dari setiap iterasi dapat dilihat pada tabel di bawah ini:

**Tabel 4. 5** Data Iterasi Variasi  $dK_p$  ,  $dK_i$  , dan  $dK_d$  Kontroler *Fuzzy*-PID *Pitch*

Iterasi ke-	Range $dK_p$	Range $dK_i$	Range $dK_d$	$T_r$	$T_s$	$T_{s\ min}$	$T_{s\ max}$	%OS	%US	$P$	$T_p$	$e_{ss}$
1	15568.77	10000.00	12019.09	0.04	4.38	2.91	5.09	1.79	0.00	5.09	6.04	7.61E-04
2	31137.55	20000.00	24038.18	0.03	6.09	2.90	5.10	2.06	0.00	5.10	5.82	9.94E-04
3	77843.87	50000.00	60095.45	0.02	6.36	2.89	5.12	2.44	0.00	5.12	5.57	1.60E-03
4	155687.74	100000.00	120190.90	0.01	6.42	2.88	5.13	2.68	0.00	5.12	5.44	2.60E-03
5	1556877.45	1000000.00	1201909.00	0.00	6.51	2.86	5.14	3.09	0.00	5.14	5.32	1.56E-02

Dari Tabel 4.5 tampak bahwa nilai  $e_{ss}$  dan %OS terkecil dimiliki oleh iterasi ke-1.



**Gambar 4. 19** Respons Sistem terhadap Waktu – Iterasi Variasi  $dK_p$  ,  $dK_i$  , dan  $dK_d$  Kontroler *Fuzzy* PID *Pitch*

Untuk memastikan performansi iterasi ke-1 adalah yang terbaik, dibuatlah plot perbandingan respons sistem terhadap waktu seperti tampak dalam Gambar 4.19. Pada Gambar 4.19 dapat dilihat bahwa respons sistem terbaik dimiliki oleh iterasi ke-1 dengan nilai *range tuning* parameter *gain* proporsional ( $dK_p$ ), integral ( $dK_i$ ), dan derivatif ( $dK_d$ ) sebesar 1 kali dari parameter *gain* proporsional, integral, dan derivatif dari kontroler PID *pitch* sebelum di-*tuning* dengan logika *fuzzy*. Didukung dengan nilai  $e_{ss}$  dan %OS dari iterasi ke-1 pada Tabel 4.5, dapat disimpulkan bahwa *range* terbaik untuk kontroler *fuzzy* PID adalah 1 kali parameter *gain* PID *pitch*.

#### 4.2.3 Subsistem Pengendali Kecepatan (*Steering*)

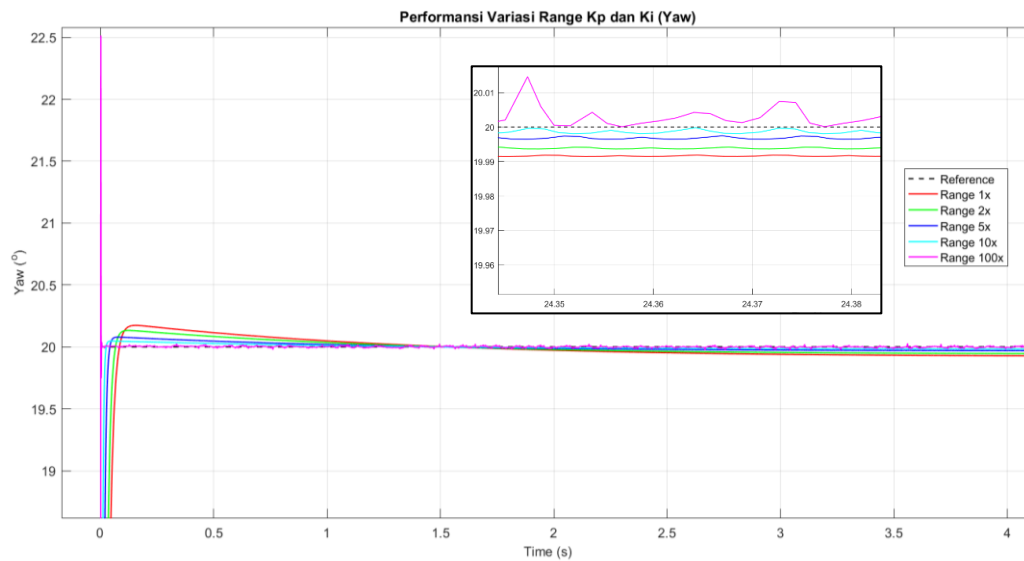
Data nilai parameter *gain* proporsional dan derivatif pada kontroler PD *yaw* sebelum dimodifikasi dengan *fuzzy*, variasi nilai *range*, dan performansi dari setiap iterasi dapat dilihat pada tabel di bawah ini:

**Tabel 4. 6** Data Iterasi Variasi *Range*  $dK_p$  dan  $dK_d$  Kontroler *Fuzzy*-PD *Yaw*

Iterasi ke-	<i>Range</i> $dK_p$	<i>Range</i> $dK_d$	$T_r$	$T_s$	$T_{s\ min}$	$T_{s\ max}$	%OS	%US	$P$	$T_p$	$e_{ss}$
1	6580.60	10000.00	0.04	0.06	18.11	20.17	0.87	0.00	20.17	0.16	3.30E-04
2	13161.20	20000.00	0.03	0.05	18.21	20.13	0.66	0.00	20.13	0.13	-1.34E-04
3	32903.01	50000.00	0.02	0.03	18.29	20.08	0.40	0.00	20.08	0.08	1.42E-04
4	65806.02	100000.00	0.01	0.02	18.05	20.05	0.24	0.00	20.05	0.05	-1.30E-04
5	658060.17	1000000.00	0.00	0.00	19.75	22.51	12.57	0.00	22.51	0.00	-1.05E-04

Dari Tabel 4.6 tampak bahwa nilai  $e_{ss}$  terkecil dimiliki oleh iterasi ke-5 dengan *range tuning* 100x parameter *gain* PD, namun meskipun perhitungan  $e_{ss}$  menggunakan MATLAB dapat memberikan gambaran terkait performansi kontroler saat *steady*, kurang tepat apabila hanya mempertimbangkan nilai  $e_{ss}$  untuk memilih *range* parameter terbaik. Hal ini juga dikarenakan metode perhitungan  $e_{ss}$  pada MATLAB yang hanya menghitung selisih antara nilai keluaran dengan *setpoint* di akhir waktu simulasi, sehingga kurang tepat untuk respons sistem yang beresilasi terhadap waktu. Sebaliknya, %OS terminimum dimiliki oleh iterasi ke-4 dengan nilai  $e_{ss}$  yang tidak berbeda jauh dari iterasi ke-5.

Perbandingan plot respons sistem terhadap waktu dari masing-masing iterasi ditinjau untuk pertimbangan tambahan, sebagai berikut:



**Gambar 4. 20** Respons Sistem terhadap Waktu – Iterasi Variasi  $Range\ dK_p$  dan  $dK_d$  Kontroler *Fuzzy PD Yaw*

Gambar 4.20 menunjukkan bahwa, meskipun beresilasi, iterasi ke-4 dengan nilai *range tuning* parameter *gain* proporsional ( $dK_p$ ) dan derivatif ( $dK_d$ ) sebesar 10 kali dari parameter *gain* proporsional dan integral dari kontroler PD *yaw* sebelum di-*tuning* dengan logika *fuzzy* paling mampu mendekati *setpoint* apabila dibandingkan dengan iterasi lainnya. Selain itu, pada Tabel 4.6 juga dapat dilihat bahwa iterasi ke-4 memiliki %OS terminimum sehingga dapat disimpulkan bahwa iterasi ke-4 merupakan *range* terbaik.

**Tabel 4. 7** Parameter *Fuzzy-PID*

	<i>Surge</i>	<i>Heave Speed</i>	<i>Pitch</i>	<i>Yaw</i>
Type	<i>Fuzzy PI</i>	<i>Fuzzy PI</i>	<i>Fuzzy PID</i>	<i>Fuzzy PD</i>
<i>Range Error</i>	0.8	0.4	25	180
<i>Range Error Rate</i>	3.52E+12	8.80E+11	5.50E+13	3.96E+14
$K_p$	4,379.52	12,729.55	15,568.77	6,580.60
$K_i$	738.61	3,380.07	10,000.00	-
$K_d$	-	-	12,019.09	10,000.00
$N$	-	-	4,624.51	2,209.73
$1/T_t$	0.21	0.33	0.33	-
<i>Range dK<sub>p</sub></i>	437951.96	1272954.83	15568.77	65806.02
<i>Range dK<sub>i</sub></i>	73861.41	338006.75	10000.00	-
<i>Range dK<sub>d</sub></i>	-	-	12019.09	100000.00

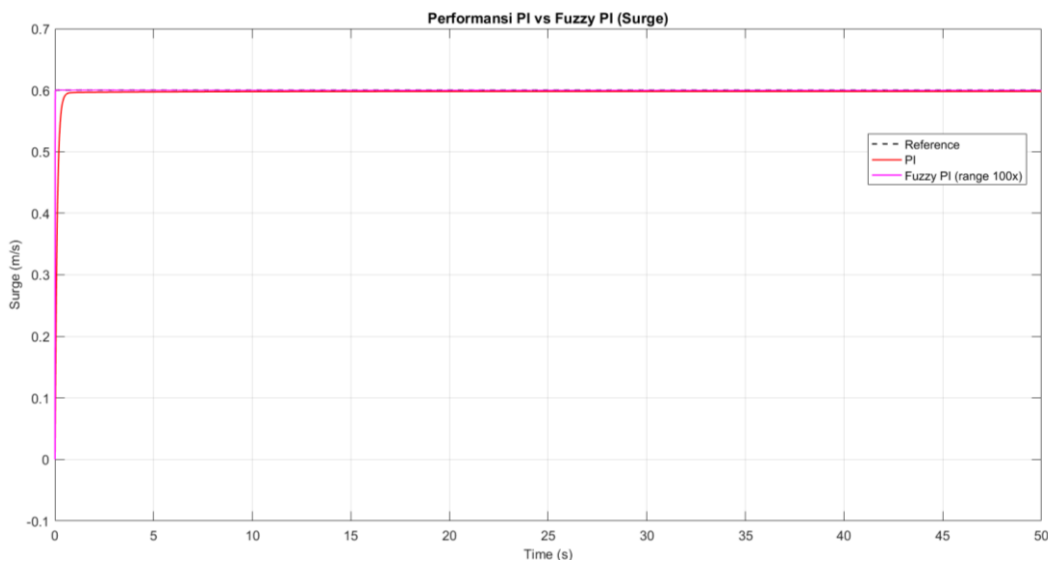
Tabel 4.7 menunjukkan rangkuman parameter yang digunakan dalam *fuzzy*-PID setelah iterasi dilakukan. Dari uji iterasi *range*  $dK_p$ ,  $dK_i$ , dan  $dK_d$  ini juga dapat disimpulkan bahwa semakin besar *range tuning* tidak menjamin performansi yang dihasilkan adalah yang terbaik, sehingga memang diperlukan uji coba untuk melihat secara langsung performansi *fuzzy*-PID dalam membantu *self-tuning* PID agar mampu mencapai *setpoint* dengan lebih cepat dan stabil meskipun ada pengaruh dari gangguan.

### 4.3 Perbandingan Performansi Uji *Closed Loop*

Setelah memperoleh parameter terbaik untuk *fuzzy*-PID, langkah berikutnya adalah memasukkan data parameter tersebut ke sistem pengendali dan membandingkan performansi dari masing-masing kontroler *fuzzy*-PID dengan PID.

#### 4.3.1 Subsistem Pengendali Kecepatan (*Speed*)

Berikut merupakan plot perbandingan respons sistem pengendali *surge* menggunakan *fuzzy*-PI dengan PI:



**Gambar 4. 21** Perbandingan *Fuzzy*-PI dengan PI untuk Pengendali *Surge*

Dari Gambar 4.21 dapat dilihat bahwa kontroler *fuzzy*-PI lebih mampu mengurangi *error steady state* dan mampu mencapai *setpoint* dalam waktu yang lebih cepat bila dibandingkan dengan kontroler PI.



Berikut merupakan data kuantitatif perbandingan performansi kontroler *fuzzy*-PI dengan PI dalam mengendalikan *surge*:

**Tabel 4. 8** Perbandingan Performansi *Fuzzy*-PI dengan PI untuk *Surge*

Kontroler	$T_r$	$T_s$	$T_{s\ min}$	$T_{s\ max}$	%OS	%US	$P$	$T_p$	$e_{ss}$
PI	0.23	0.43	0.54	0.60	0.00	0.00	0.60	50.00	2.10E-03
<i>Fuzzy</i> -PI	0.01	0.01	0.58	0.60	0.07	0.00	0.60	0.02	9.14E-05

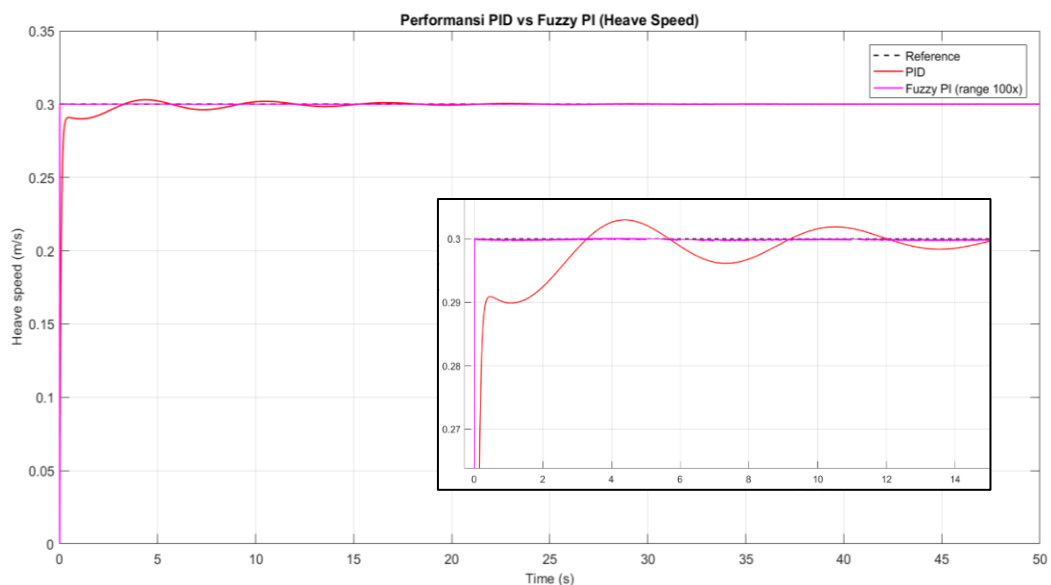
Tabel 4.8 menunjukkan bahwa performansi *fuzzy*-PI lebih unggul dibandingkan PI dalam mengendalikan *surge* karena memiliki nilai *error steady state*, *percent overshoot*, *settling time*, dan *rise time* yang lebih rendah meskipun dikenai gangguan.

### 4.3.2 Subsistem Pengendali Kedalaman (*Depth*)

Subsistem pengendali kedalaman dibagi lagi menjadi dua yakni pengendali *heave* dan pengendali *pitch*.

#### 4.3.2.1 Pengendali *Heave*

Berikut merupakan plot perbandingan respons sistem pengendali *heave* menggunakan *fuzzy*-PI dengan PI:



**Gambar 4. 22** Perbandingan *Fuzzy*-PI dengan PI untuk Pengendali *Heave*

Dari Gambar 4.22 dapat dilihat bahwa kontroler *fuzzy*-PI lebih cepat mencapai kondisi *steady*, mampu mengurangi *error steady state*, dan mampu mencapai *setpoint* dalam waktu yang lebih cepat bila dibandingkan dengan kontroler PI. Berikut merupakan data kuantitatif perbandingan performansi kontroler *fuzzy*-PI dengan PI dalam mengendalikan *heave*:

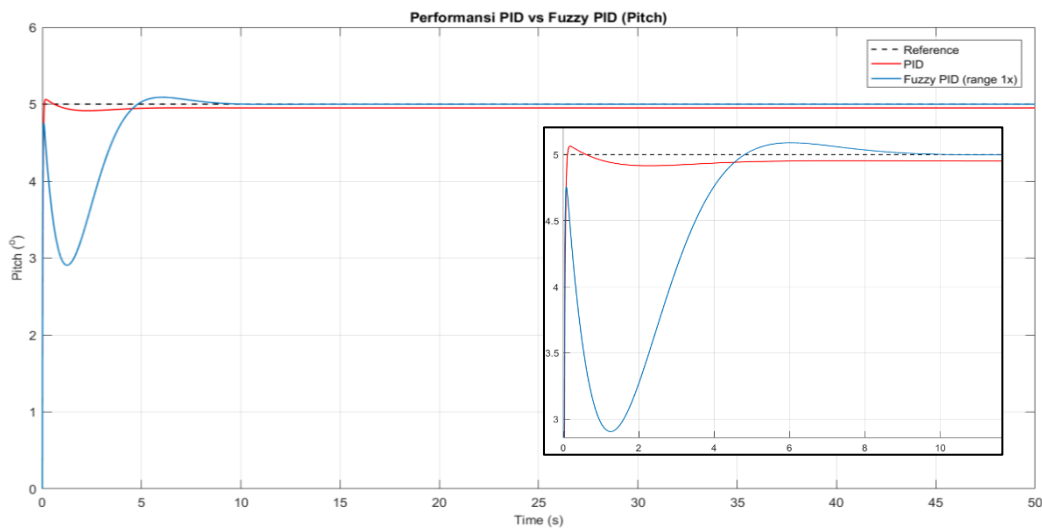
**Tabel 4. 9** Perbandingan Performansi *Fuzzy*-PI dengan PI untuk *Heave*

Kontroler	$T_r$	$T_s$	$T_{s\ min}$	$T_{s\ max}$	%OS	%US	$P$	$T_p$	$e_{ss}$
PI	0.16	2.28	0.28	0.30	0.99	0.00	0.30	0.44	-1.76E-07
<i>Fuzzy</i> -PI	0.00	0.01	0.27	0.30	0.05	0.00	0.30	4.00	2.05E-05

Tabel 4.9 menunjukkan bahwa, meskipun memiliki *error steady state* yang lebih besar namun performansi *fuzzy*-PI lebih unggul dibandingkan PI dalam mengendalikan *heave* karena memiliki nilai *percent overshoot*, *settling time*, dan *rise time* yang lebih rendah meskipun dikenai gangguan. Hal tersebut juga mempertimbangkan metode penghitungan *error steady state* pada MATLAB, yakni menghitung selisih antara nilai keluaran dengan *setpoint* pada akhir waktu simulasi sehingga kurang cukup untuk menggambarkan keunggulan sistem yang beresilasi meskipun osilasinya tidak besar.

#### 4.3.2.2 Pengendali *Pitch*

Berikut merupakan plot perbandingan respons sistem pengendali *pitch* menggunakan *fuzzy*-PID dengan PID:



**Gambar 4. 23** Perbandingan *Fuzzy*-PID dengan PID untuk Pengendali *Pitch*

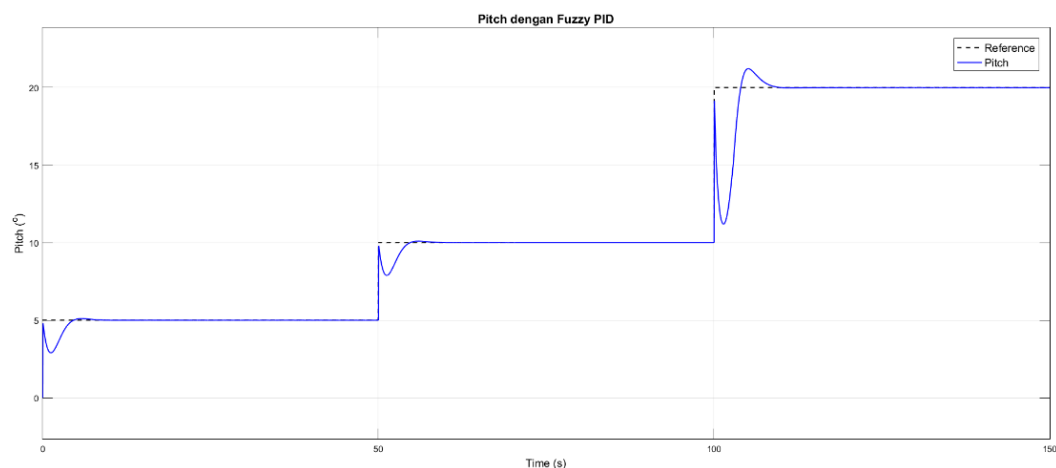
Dari Gambar 4.23 dapat dilihat bahwa kontroler *fuzzy*-PID mampu mengurangi *error steady state* bila dibandingkan dengan kontroler PID.

Berikut merupakan data kuantitatif perbandingan performansi kontroler *fuzzy*-PID dengan PID dalam mengendalikan *pitch*:

**Tabel 4. 10** Perbandingan Performansi *Fuzzy*-PID dengan PID untuk *Pitch*

Kontroler	$T_r$	$T_s$	$T_{s\ min}$	$T_{s\ max}$	%OS	%US	$P$	$T_p$	$e_{ss}$
PID	0.05	0.27	4.48	5.06	2.26	0.00	5.06	0.17	4.95E-02
<i>Fuzzy</i> -PID	0.04	4.38	2.91	5.09	1.79	0.00	5.09	6.04	7.61E-04

Tabel 4.10 menunjukkan bahwa performansi *fuzzy*-PID lebih unggul dibandingkan PID dalam mengendalikan *pitch* karena memiliki nilai *error steady state*, *percent overshoot*, *settling time*, dan *rise time* yang lebih rendah meskipun dikenai gangguan.

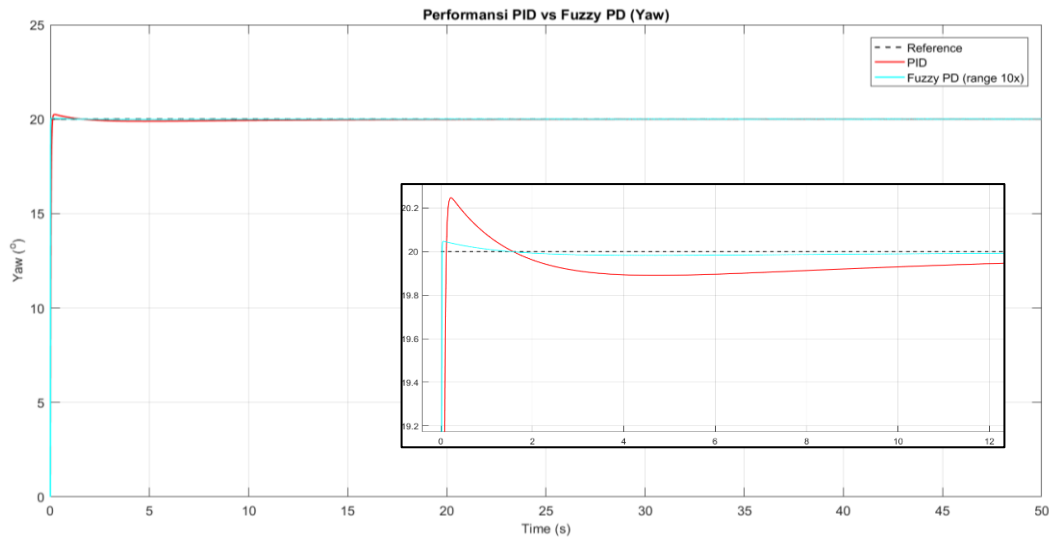


**Gambar 4. 24** Uji *Closed Loop Fuzzy*-PID *Pitch*

Kelebihan lainnya dari kontroler *fuzzy*-PID tampak pada Gambar 4.23, yakni mampu mengatasi nonlinearitas sistem yang dikendalikan sehingga perubahan *setpoint* seiring berjalannya waktu tidak membuat respons sistem kehilangan kemampuannya untuk mencapai *setpoint* seperti pada kontroler PI pada Gambar 4.10. Untuk *overshoot* yang bertambah, dapat dikompensasi dengan menambah *gain* proporsional  $K_p$ , namun dalam penelitian ini *gain* parameter proporsional, integral, dan derivatif *fuzzy*-PID dipertahankan sama dengan PID.

### 4.3.3 Subsistem Pengendali Haluan (*Steering*)

Berikut merupakan plot perbandingan respons sistem pengendali *yaw* menggunakan *fuzzy*-PD dengan PD:



**Gambar 4. 25** Perbandingan *Fuzzy*-PD dengan PD untuk Pengendali *Yaw*

Dari Gambar 4.25 dapat dilihat bahwa kontroler *fuzzy*-PD lebih cepat mencapai kondisi *steady*, mampu mengurangi *error steady state*, dan mampu mencapai *setpoint* dalam waktu yang lebih cepat bila dibandingkan dengan kontroler PD.

Berikut merupakan data kuantitatif perbandingan performansi kontroler *fuzzy*-PD dengan PD dalam mengendalikan *yaw*:

**Tabel 4. 11** Perbandingan Performansi *Fuzzy*-PD dengan PD untuk *Yaw*

Kontroler	$T_r$	$T_s$	$T_{s\ min}$	$T_{s\ max}$	%OS	%US	$P$	$T_p$	$e_{ss}$
PD	0.06	0.09	18.49	20.25	1.24	0.00	20.25	0.23	6.36E-04
<i>Fuzzy</i> -PD	0.01	0.02	18.05	20.05	0.24	0.00	20.05	0.05	-1.30E-04

Tabel 4.11 menunjukkan bahwa performansi *fuzzy*-PD lebih unggul dibandingkan PD dalam mengendalikan *yaw* karena memiliki nilai *error steady*

*state*, *percent overshoot*, *settling time*, dan *rise time* yang lebih rendah meskipun dikenai gangguan.

#### **4.4 Perbandingan Hasil Simulasi *Waypoint Following***

Simulasi *waypoint following* bertujuan untuk menguji apakah sistem pengendalian yang telah dirancang mampu menyelesaikan misi untuk mencapai titik-titik koordinat 3D yang harus dilalui oleh AUV. Sesuai Metodologi, AUV dikatakan berhasil mencapai suatu titik koordinat apabila telah berhasil memasuki *circle of acceptance* (COA) atau daerah berbentuk lingkaran dengan radius tertentu yang telah disepakati. Dalam simulasi ini, AUV dianggap berhasil mencapai titik yang dituju apabila telah memasuki COA baik COA bidang X-Y maupun bidang X-Z. Pada umumnya, nilai maksimum radius COA sama dengan 2 kali panjang badan AUV. Sesuai spesifikasi pada Tabel 3.1, panjang badan AUV tipe *double-hull* di CEiiA adalah 2,8 meter, sehingga nilai maksimum radius COA adalah 5,6 meter. Namun, radius COA pada bidang X-Y ( $\rho_k$ ) dalam simulasi ini ditetapkan sebagai 1,5 meter sedangkan radius COA pada bidang X-Z ( $\rho_z$ ) ditetapkan sebagai 0,5 meter.

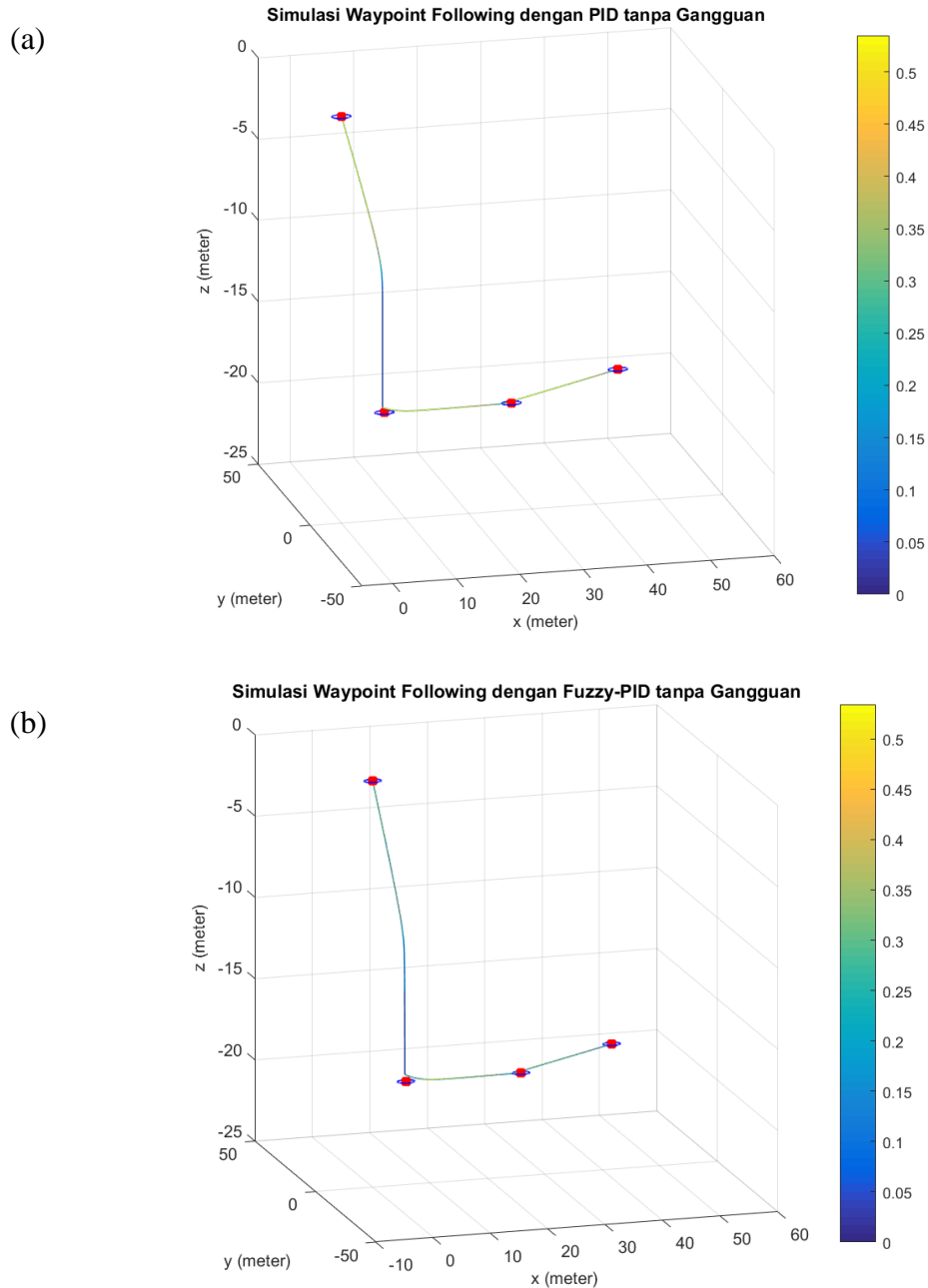
Simulasi *waypoint following* dibagi ke dalam dua tahap, yakni simulasi *waypoint following* tanpa gangguan arus laut dan simulasi *waypoint following* dengan adanya gangguan arus laut. Perbandingan antara *fuzzy*-PID dengan PID yang telah dirancang akan dinilai secara kualitatif berdasarkan bentuk akhir lintasan yang dilalui secara 3D dan respons sistem dari pengendali *surge*, *heave*, *pitch*, dan *yaw*.

##### **4.4.1 Simulasi *Waypoint Following* tanpa Gangguan Arus Laut**

Pada bagian ini akan dijelaskan hasil simulasi *waypoint following* menggunakan sistem pengendali PID dan *fuzzy*-PID yang telah dirancang, tanpa adanya gangguan arus laut. Untuk mempermudah analisa performansi antara kedua sistem kendali, maka bagian ini akan langsung dibagi berdasarkan plot yang dibandingkan, yakni: lintasan, performa *surge*, performa *heave*, performa *pitch*, dan performa *yaw*.

#### 4.4.1.1. Perbandingan Lintasan

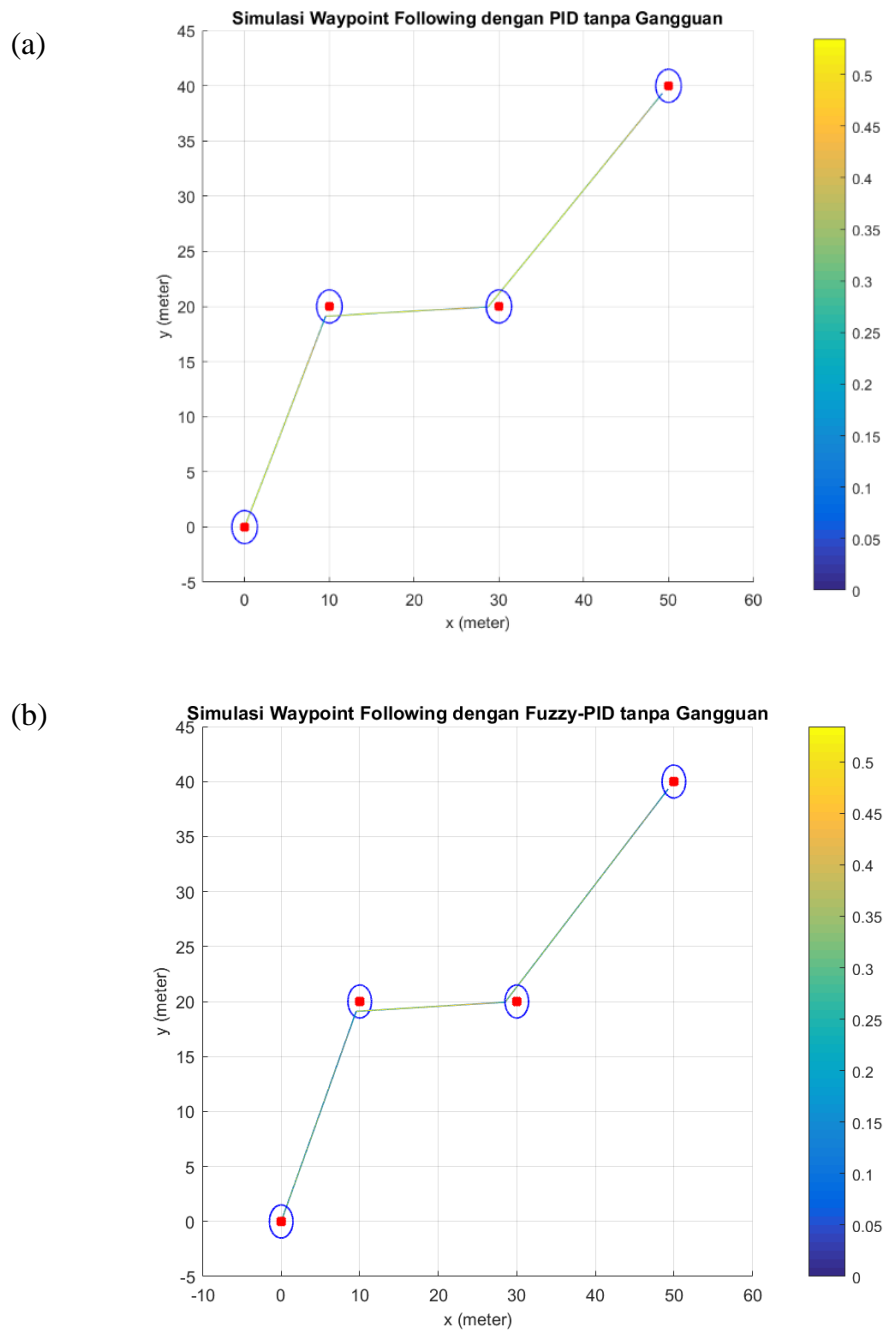
Berikut merupakan perbandingan lintasan hasil simulasi *waypoint following* menggunakan sistem pengendali PID dan *fuzzy*-PID tanpa adanya gangguan arus laut:



**Gambar 4. 26** Lintasan X-Y-Z Simulasi tanpa Gangguan Arus Laut dengan

(a)PID; (b) *Fuzzy*-PID

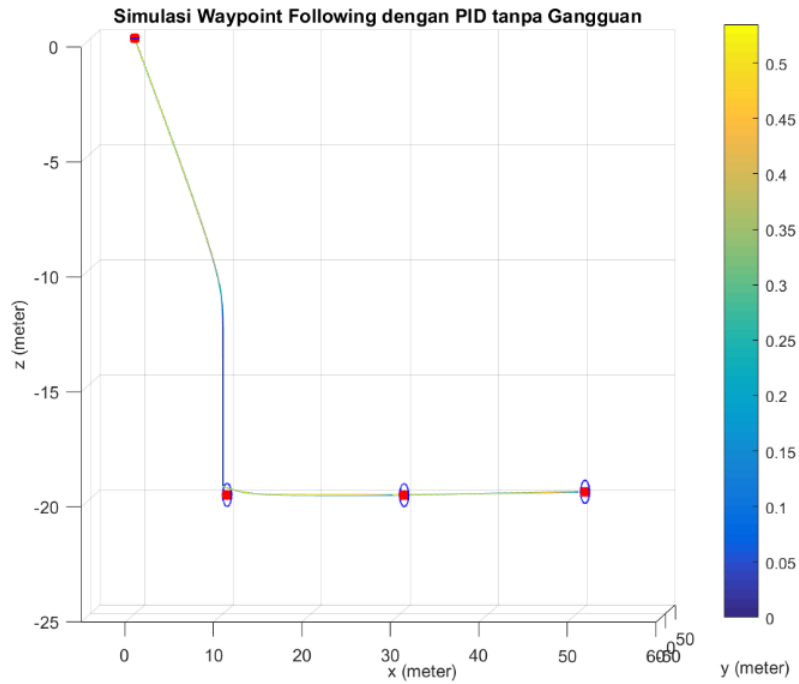
Gambar 4.26 menunjukkan bahwa kedua sistem pengendali mampu mencapai titik-titik koordinat yang dituju bila ditinjau secara 3D.



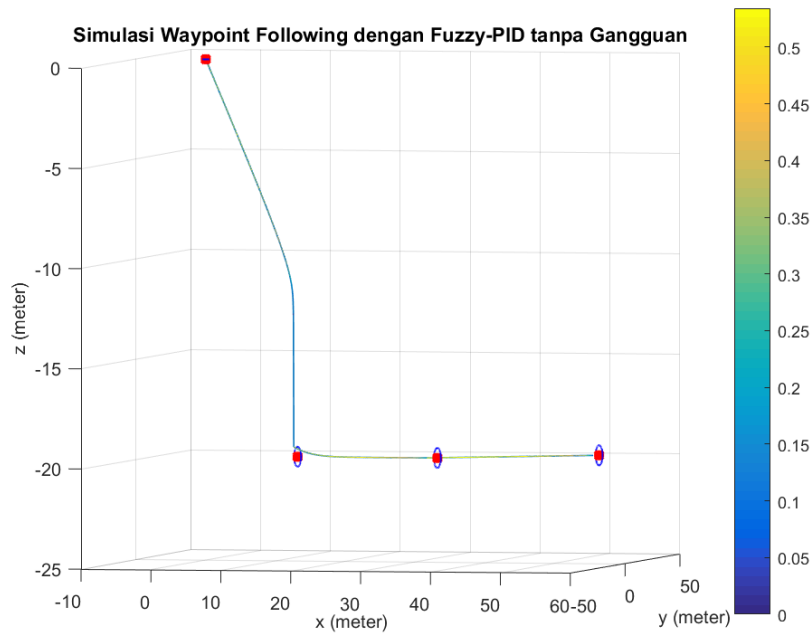
**Gambar 4. 27** Lintasan X-Y Simulasi tanpa Gangguan Arus Laut dengan (a) PID; (b) *Fuzzy*-PID

Gambar 4.27 menunjukkan bahwa kedua sistem pengendali mampu mencapai menggerakkan AUV hingga memasuki COA bidang X-Y.

(a)



(b)



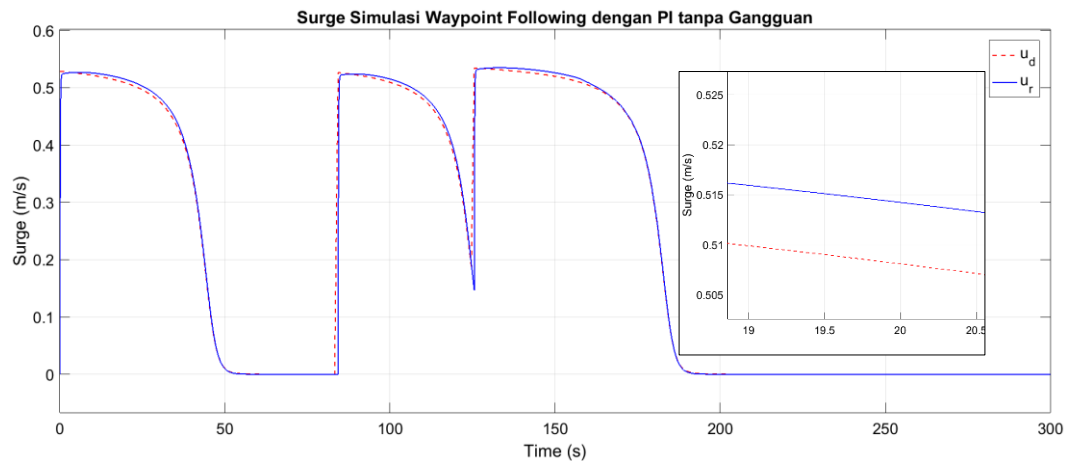
**Gambar 4. 28** Lintasan X-Z Simulasi tanpa Gangguan Arus Laut dengan (a) PID; (b) *Fuzzy*-PID

Gambar 4.28 menunjukkan bahwa kedua sistem pengendali mampu mencapai menggerakkan AUV hingga memasuki COA bidang X-Z.

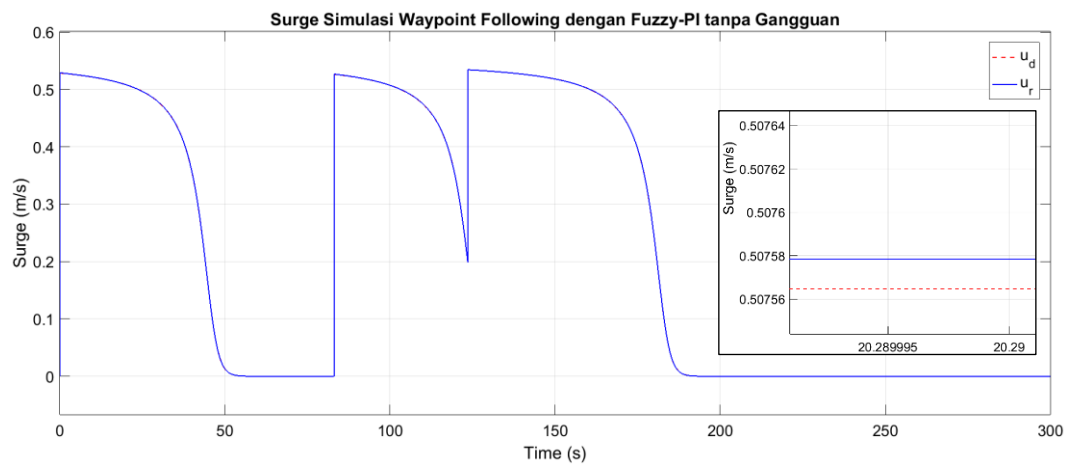


#### 4.4.1.2 Perbandingan *Surge*

Berikut merupakan perbandingan plot respon sistem pengendali PI dan *fuzzy*-PI dalam mengendalikan *surge* pada simulasi *waypoint following* tanpa gangguan arus laut:



(a)



(b)

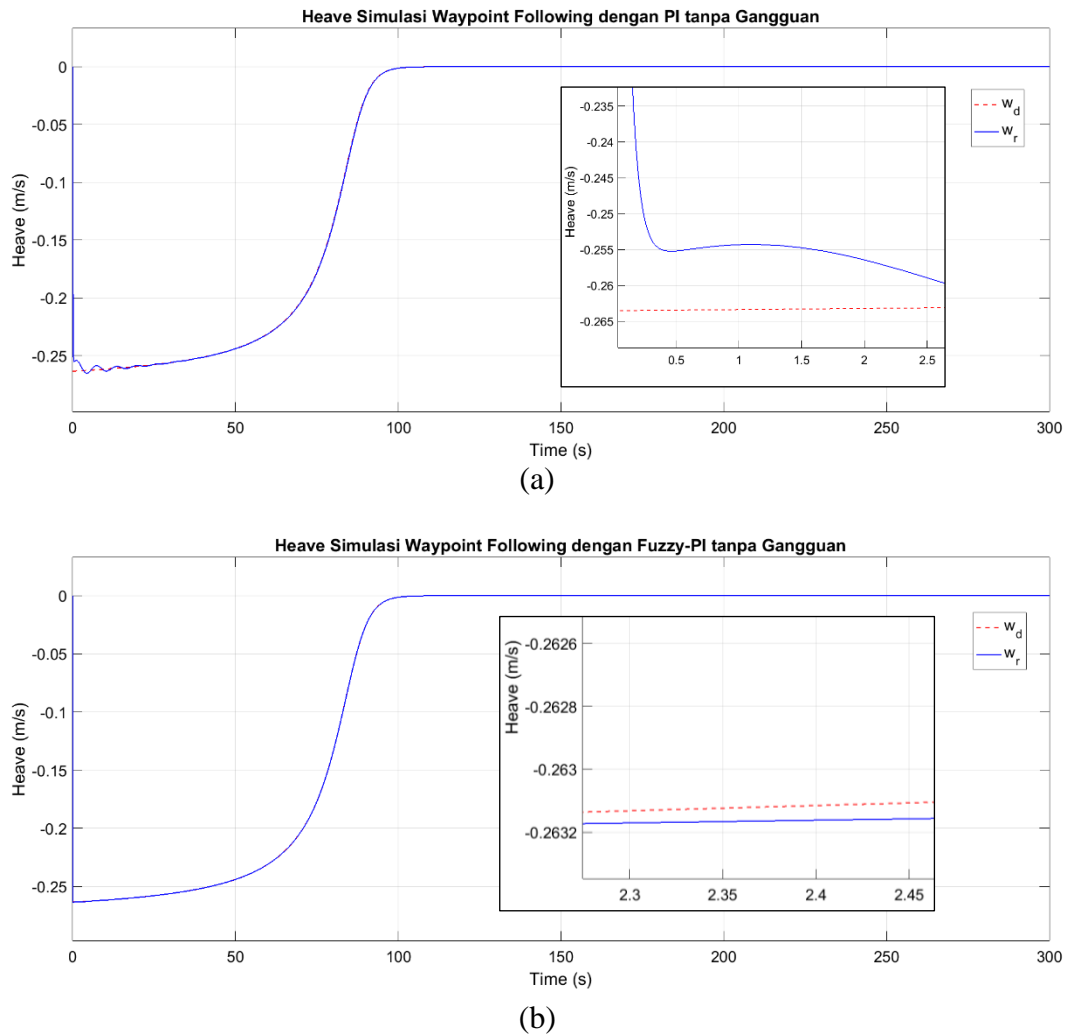
**Gambar 4. 29** Respon *Surge* pada Simulasi *Waypoint Following* tanpa Gangguan dengan: (a) PI; (b) *Fuzzy*-PI

Dari Gambar 4.29 dapat dilihat bahwa algoritma *waypoint following* yang dibangun mampu menentukan nilai *surge* yang diinginkan untuk mencapai titik koordinat yang dituju oleh AUV. Kurva nilai *surge* yang diinginkan ditandai dengan garis putus-putus berwarna merah, dan berkelok-kelok karena nilai *surge* bervariasi terhadap jarak antara AUV dengan titik koordinat yang sedang dituju. Apabila jarak antara AUV dengan titik yang dituju masih jauh maka algoritma akan meningkatkan nilai referensi *surge* sedangkan jika sudah dekat maka algoritma

akan menurunkan nilai referensi *surge* supaya hemat tenaga. Selain itu, dapat dilihat bahwa pada waktu antara 20 hingga 20,2 sekon, selisih antara *surge* yang diinginkan dengan *surge* AUV yang terukur pada sistem pengendali PI mencapai 0,005 sedangkan pada pengendali *fuzzy*-PI hanya 0,00002 sehingga dapat dibuktikan bahwa pengendali *fuzzy*-PI memiliki performa yang lebih baik bila dibandingkan pengendali PI dalam mengendalikan *surge* ketika tidak ada gangguan.

#### 4.4.1.3 Perbandingan *Heave*

Berikut merupakan perbandingan plot respon sistem pengendali PI dan *fuzzy*-PI dalam mengendalikan *heave* pada simulasi *waypoint following* tanpa gangguan arus laut:

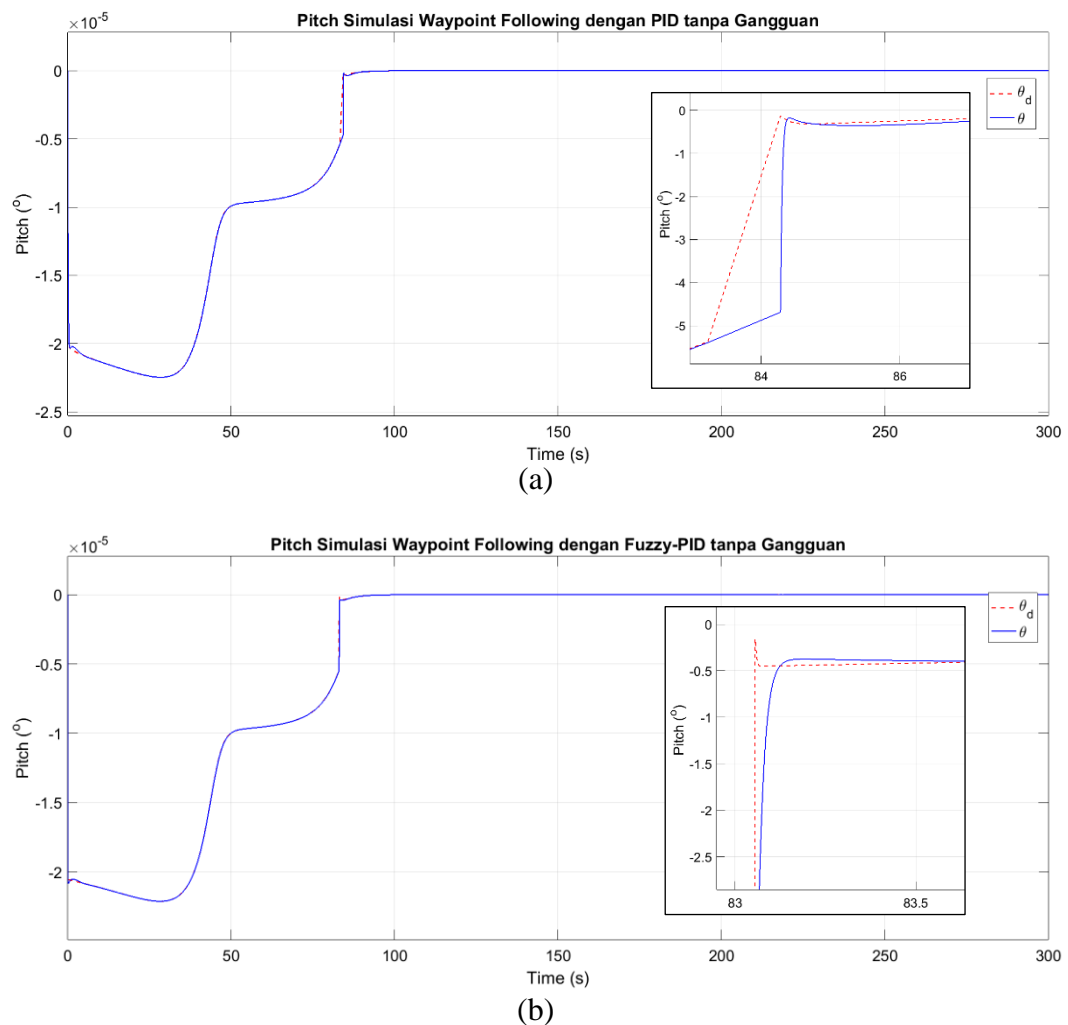


**Gambar 4. 30** Respon *Heave* pada Simulasi *Waypoint Following* tanpa Gangguan dengan: (a) PI; (b) *Fuzzy*-PI

Dari Gambar 4.30 dapat dilihat bahwa pada waktu antara 2 hingga 2,5 sekon, selisih antara *heave* yang diinginkan dengan *heave* AUV yang terukur pada sistem pengendali PI mencapai 0,05 sedangkan pada pengendali *fuzzy*-PI hanya 0,0005 sehingga dapat dibuktikan bahwa pengendali *fuzzy*-PI memiliki performa yang lebih baik bila dibandingkan pengendali PI dalam mengendalikan *heave* ketika tidak ada gangguan.

#### 4.4.1.4 Perbandingan *Pitch*

Berikut merupakan perbandingan plot respon sistem pengendali PID dan *fuzzy*-PID dalam mengendalikan *pitch* pada simulasi *waypoint following* tanpa gangguan arus laut:

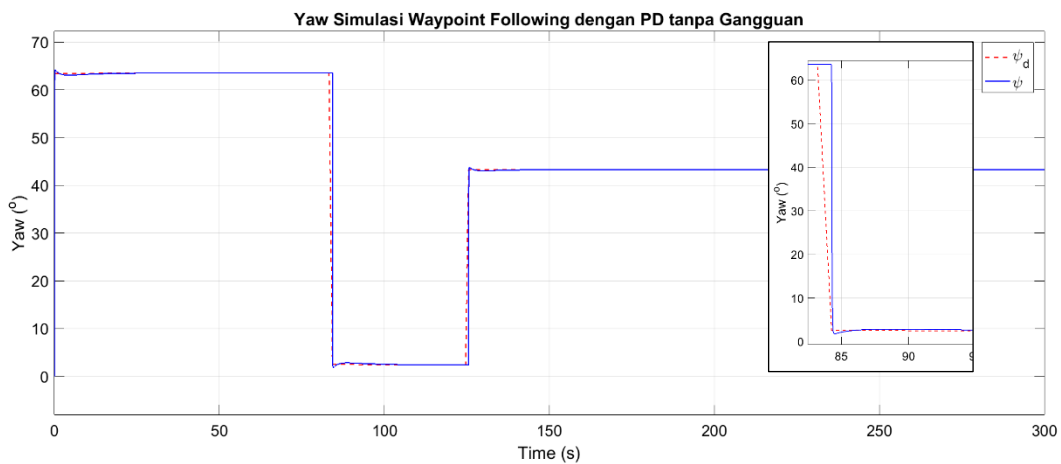


**Gambar 4. 31** Respon *Pitch* pada Simulasi *Waypoint Following* tanpa Gangguan dengan: (a) PID; (b) *Fuzzy*-PID

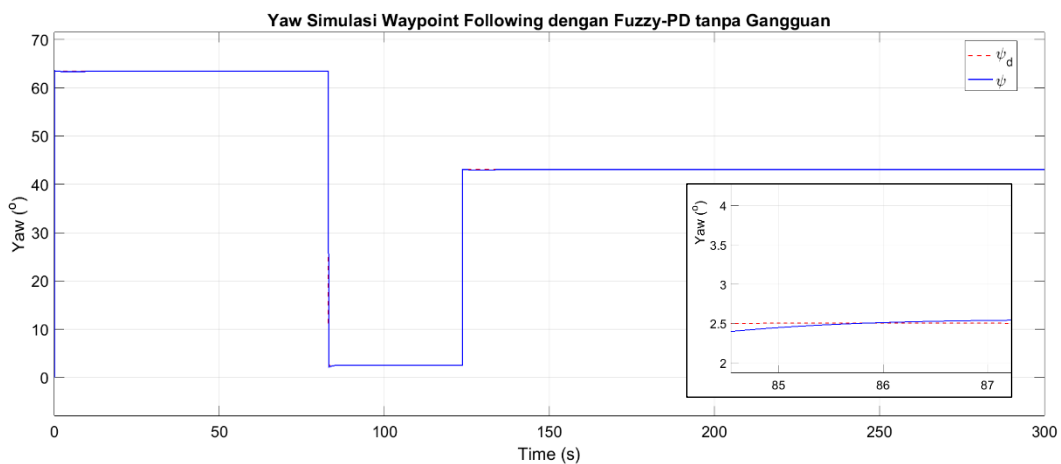
Dari Gambar 4.31 dapat dilihat bahwa pada waktu antara 83 hingga 85 sekon, respon *pitch* pada *fuzzy*-PID terhadap perubahan nilai *pitch* yang diinginkan lebih cepat bila dibandingkan pengendali PID sehingga dapat dibuktikan bahwa pengendali *fuzzy*-PI memiliki performa yang lebih baik bila dibandingkan pengendali PI dalam mengendalikan *heave* ketika tidak ada gangguan.

#### 4.4.1.5 Perbandingan Yaw

Berikut merupakan perbandingan plot respon sistem pengendali PD dan *fuzzy*-PD dalam mengendalikan *yaw* pada simulasi *waypoint following* tanpa gangguan arus laut:



(a)



(b)

**Gambar 4. 32** Respon *Yaw* pada Simulasi *Waypoint Following* tanpa Gangguan dengan: (a) PD; (b) *Fuzzy*-PD

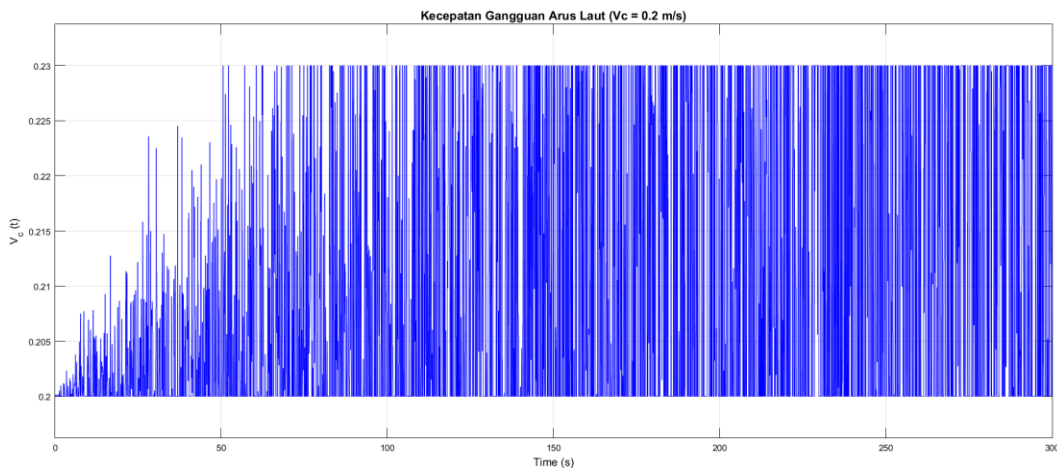
Dari Gambar 4.32 dapat dilihat bahwa pada waktu antara 85 hingga 87 sekon, respon *yaw* pada *fuzzy*-PD terhadap perubahan nilai *setpoint* lebih cepat bila dibandingkan pengendali PD sehingga dapat dibuktikan bahwa pengendali *fuzzy*-PD memiliki performa yang lebih baik bila dibandingkan pengendali PD dalam mengendalikan *yaw* ketika tidak ada gangguan.

#### 4.4.2 Simulasi *Waypoint Following* dengan Gangguan Arus Laut

Pada bagian ini akan dijelaskan hasil simulasi *waypoint following* menggunakan sistem pengendali PID dan *fuzzy*-PID yang telah dirancang, dengan adanya gangguan arus laut. Ada 3 variasi kecepatan arus laut yang disimulasikan, yakni 0,2 m/s; 40 m/s; dan 41 m/s. Pemilihan variasi didasarkan pada tujuan untuk menganalisa performansi pengendali di bawah gangguan ringan dan gangguan berat, serta mengetahui batas kecepatan arus laut yang mampu ditangani.

##### 4.4.2.1 Gangguan Arus 0,2 m/s

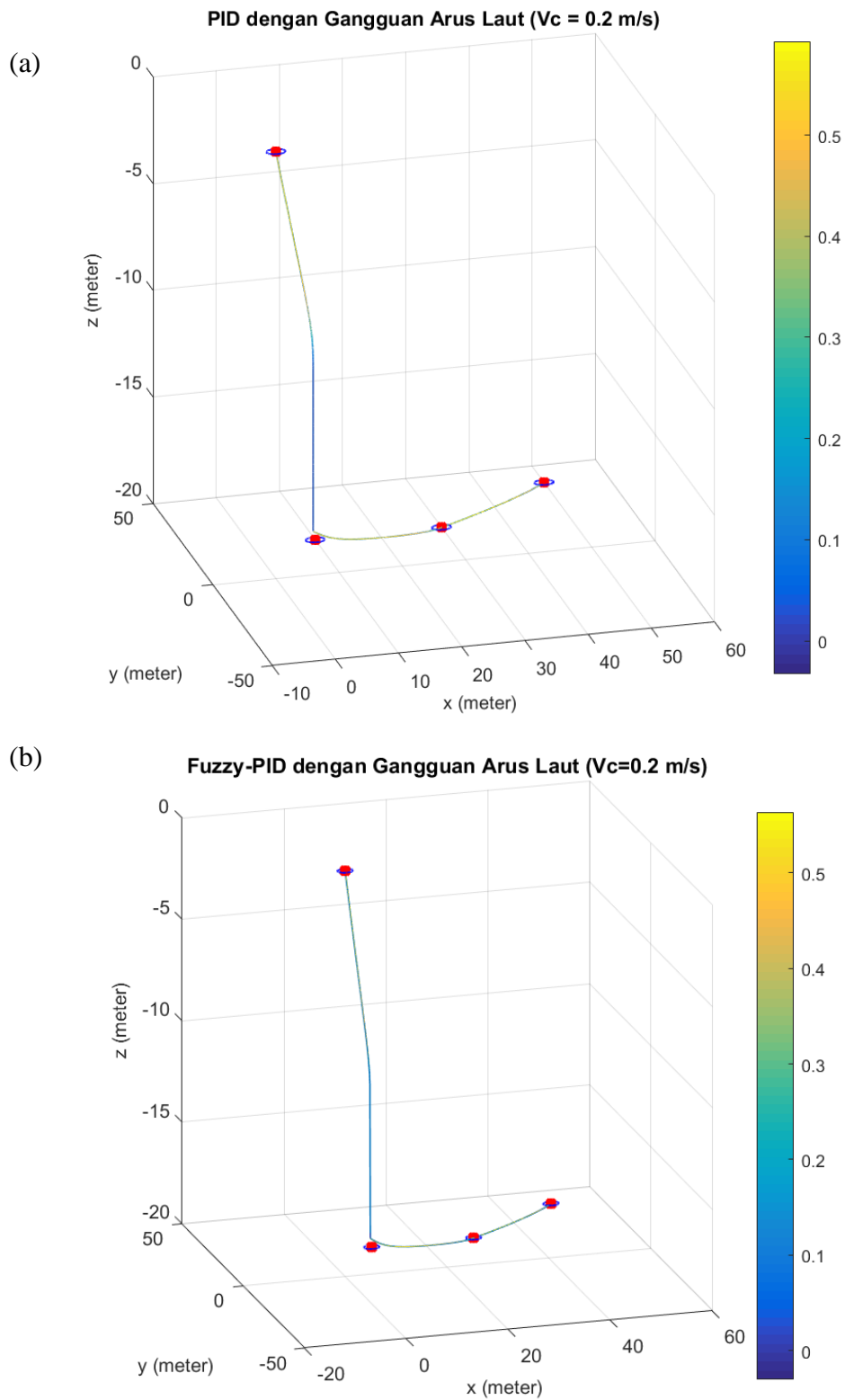
Gambar di bawah ini menunjukkan dinamika kecepatan arus laut terhadap waktu yang digunakan dalam simulasi. Adapun batas atas kecepatan arus laut ditetapkan 0,23 m/s sedangkan batas bawahnya adalah 0,2 m/s.



**Gambar 4. 33** Kecepatan Arus Laut terhadap Waktu (Variasi 0,2 m/s)

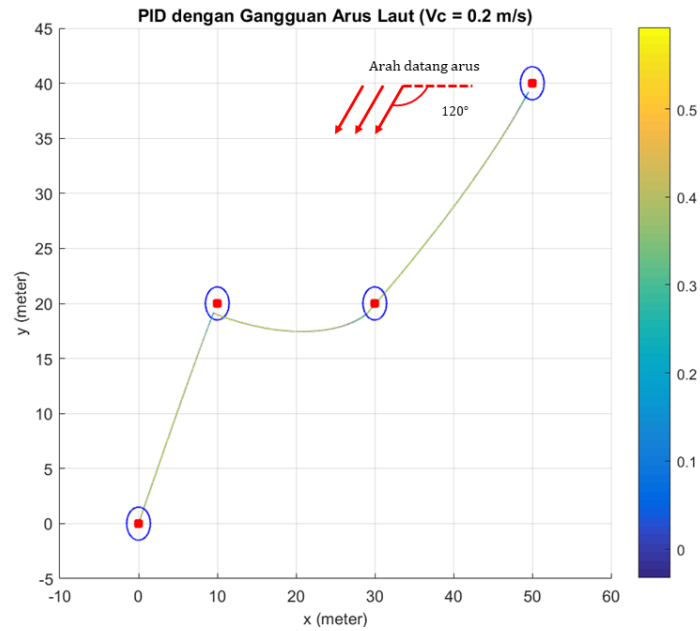
##### 4.4.2.1.1 Perbandingan Lintasan

Berikut merupakan perbandingan lintasan hasil simulasi *waypoint following* menggunakan sistem pengendali PID dan *fuzzy*-PID dengan adanya gangguan arus laut sebesar 0,2 m/s dan *side slip angle* 120°:

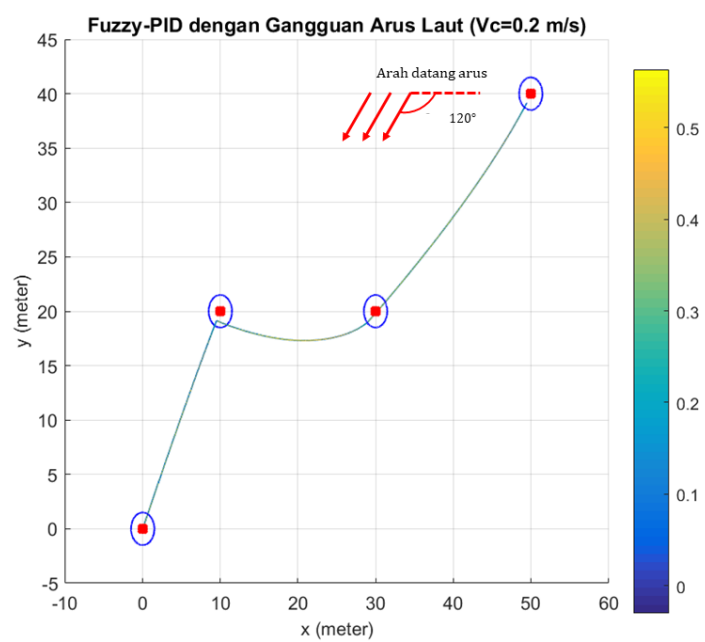


**Gambar 4. 34** Lintasan X-Y-Z Simulasi dengan Gangguan Arus Laut ( $V_c = 0,2 \text{ m/s}$ ) Menggunakan: (a) PID; (b) *Fuzzy*-PID

Gambar 4.34 menunjukkan bahwa kedua sistem pengendali mampu mencapai titik-titik koordinat yang dituju bila ditinjau secara 3D.



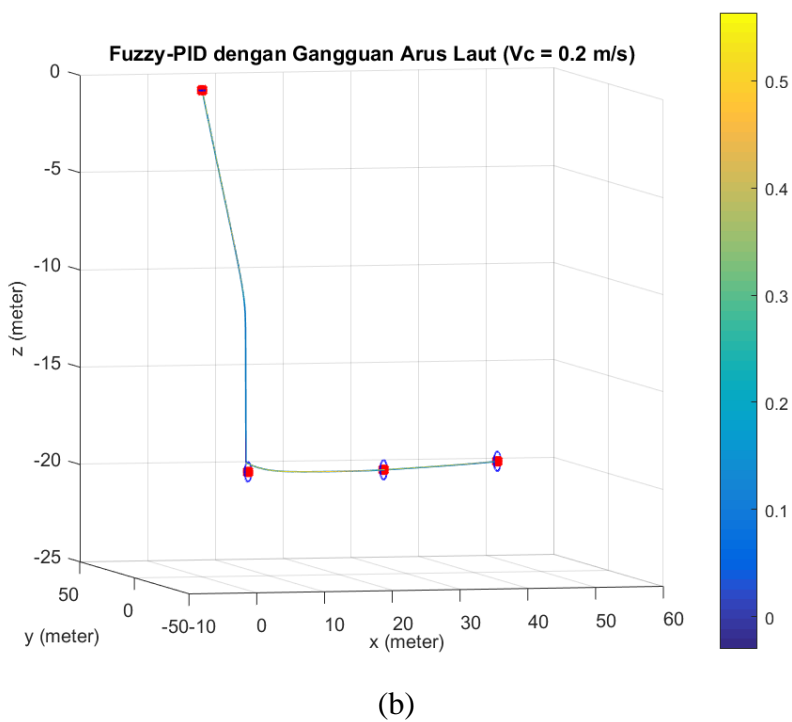
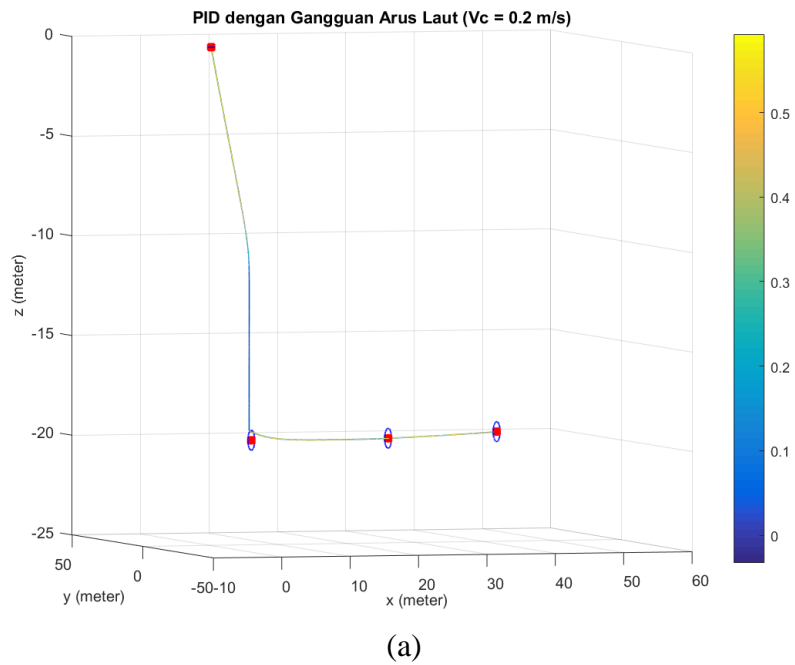
(a)



(b)

**Gambar 4. 35** Lintasan X-Y Simulasi dengan Gangguan Arus Laut ( $V_c = 0,2 \text{ m/s}$ ) Menggunakan: (a) PID; (b) *Fuzzy*-PID

Gambar 4.35 menunjukkan bahwa kedua sistem pengendali mampu mencapai menggerakkan AUV hingga memasuki COA bidang X-Y.



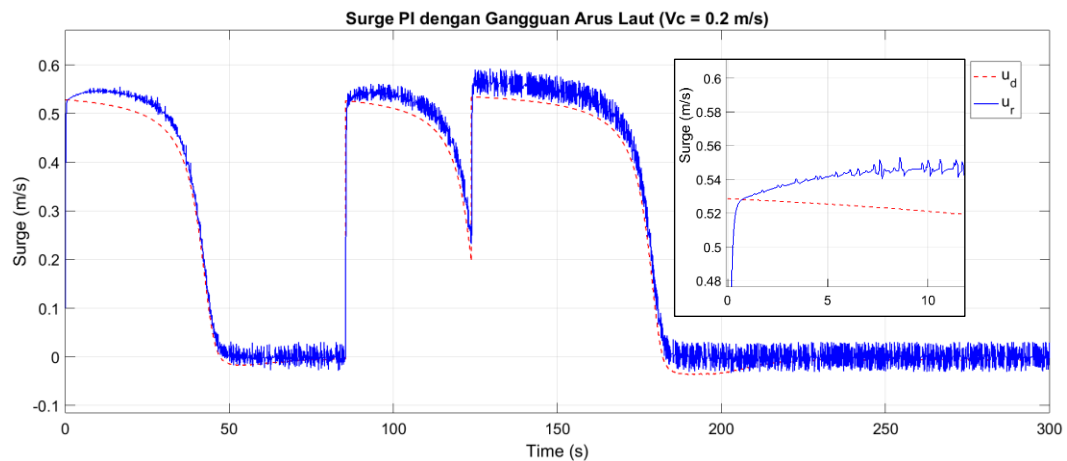
**Gambar 4. 36** Lintasan X-Z Simulasi dengan Gangguan Arus Laut ( $V_c = 0,2 \text{ m/s}$ ) Menggunakan: (a) PID; (b) *Fuzzy*-PID



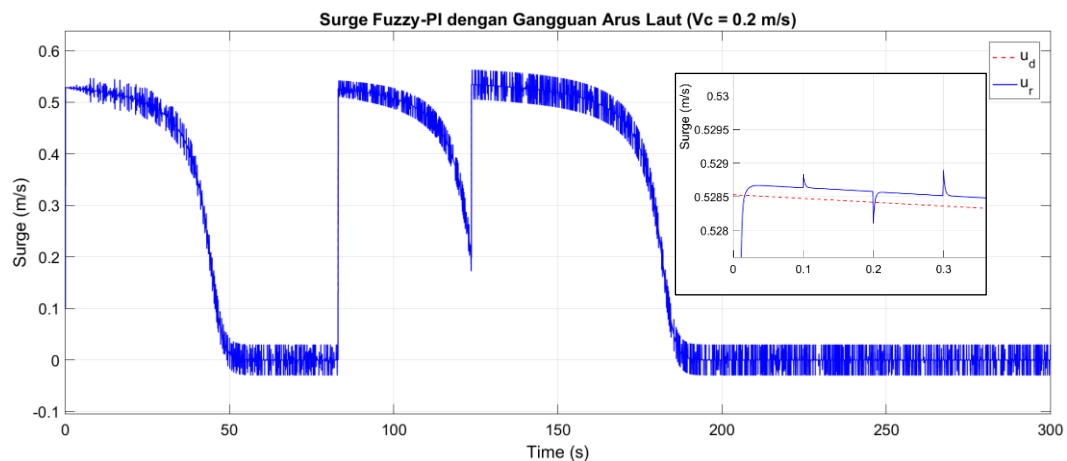
Gambar 4.36 menunjukkan bahwa kedua sistem pengendali mampu mencapai menggerakkan AUV hingga memasuki COA bidang X-Z.

#### 4.4.2.1.2 Perbandingan *Surge*

Berikut merupakan perbandingan plot respon sistem pengendali PI dan *fuzzy*-PI dalam mengendalikan *surge* pada simulasi *waypoint following* dengan gangguan arus laut ( $V_c = 0,2 \text{ m/s}$ ):



(a)



(b)

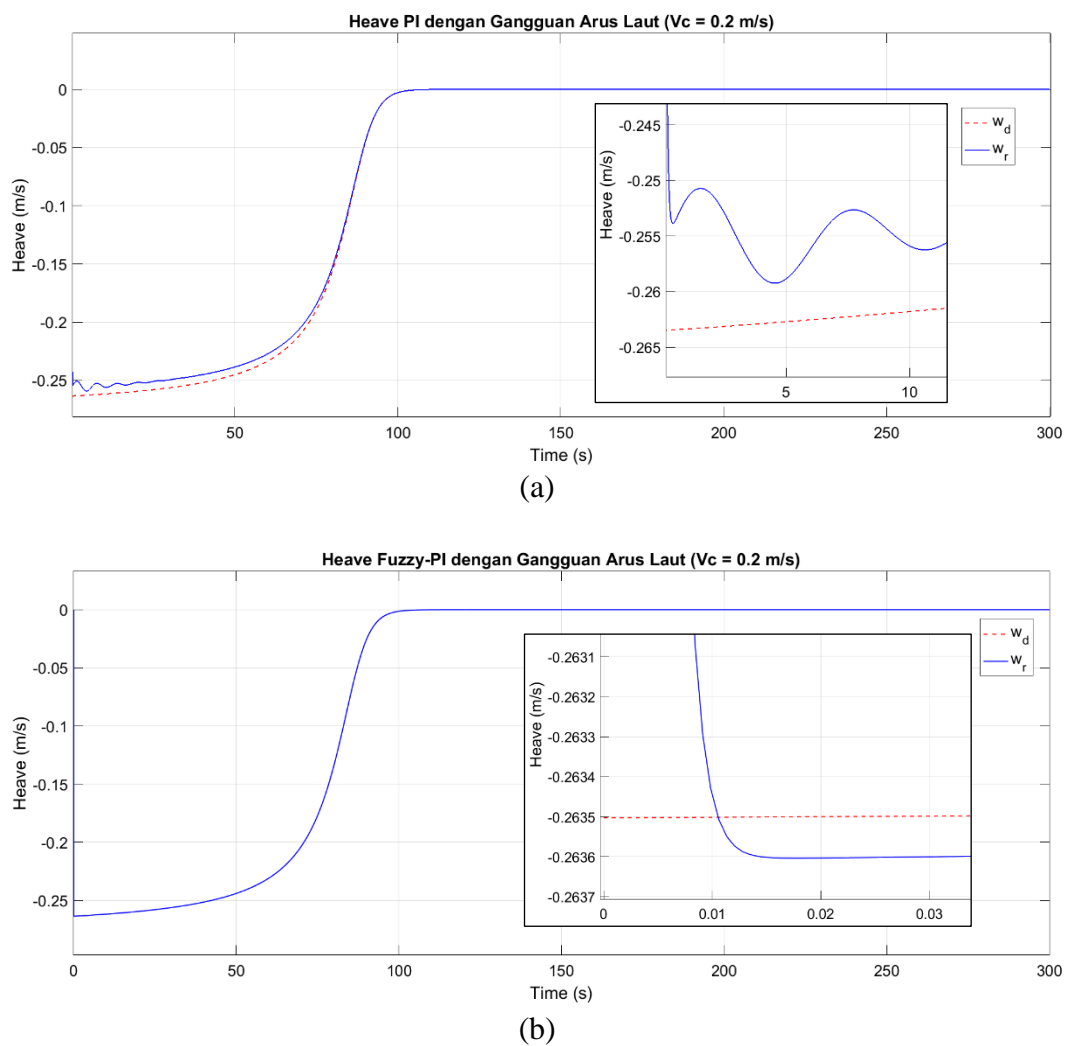
**Gambar 4. 37** Respon *Surge* pada Simulasi *Waypoint Following* dengan Gangguan Arus Laut ( $V_c = 0,2 \text{ m/s}$ ) Menggunakan: (a) PI; (b) *Fuzzy*-PI

Dari Gambar 4.37 dapat dilihat bahwa pada waktu antara 0 hingga 10 sekon, selisih antara *surge* yang diinginkan dengan *surge* AUV yang terukur pada sistem pengendali PI mencapai 0,02 sedangkan pada pengendali *fuzzy*-PI hanya 0,0002

sehingga dapat dibuktikan bahwa pengendali *fuzzy*-PI memiliki performa yang lebih baik bila dibandingkan pengendali PI dalam mengendalikan *surge* di bawah gangguan arus laut dengan kecepatan 0,2 m/s.

#### 4.4.2.1.3 Perbandingan *Heave*

Berikut merupakan perbandingan plot respon sistem pengendali PI dan *fuzzy*-PI dalam mengendalikan *heave* pada simulasi *waypoint following* dengan gangguan arus laut ( $V_c = 0,2 \text{ m/s}$ ):



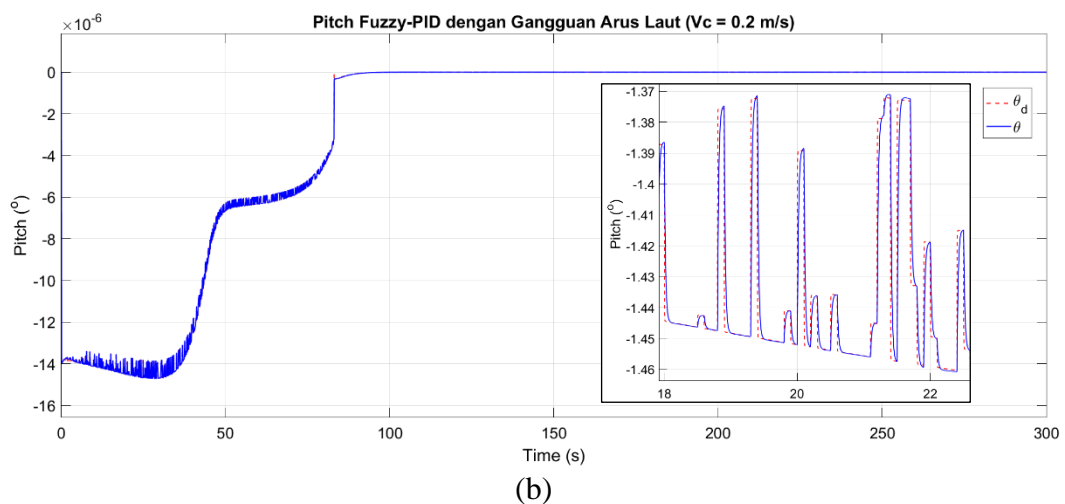
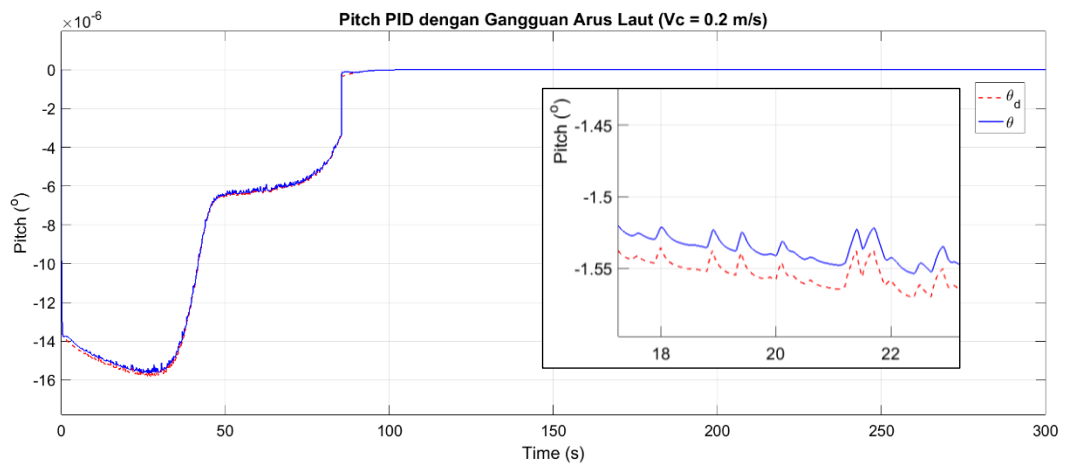
**Gambar 4. 38** Respon *Heave* pada Simulasi *Waypoint Following* dengan Gangguan Arus Laut ( $V_c = 0,2 \text{ m/s}$ ) Menggunakan: (a) PI; (b) *Fuzzy*-PI

Dari Gambar 4.38 dapat dilihat bahwa pada waktu antara 0 hingga 10 detik, selisih antara *heave* yang diinginkan dengan *heave* AUV yang terukur pada sistem

pengendali PI mencapai 0,01 sedangkan pada pengendali *fuzzy*-PI hanya 0,0001 sehingga dapat dibuktikan bahwa pengendali *fuzzy*-PI memiliki performa yang lebih baik bila dibandingkan pengendali PI dalam mengendalikan *heave* di bawah gangguan arus laut dengan kecepatan 0,2 m/s.

#### 4.4.2.1.4 Perbandingan *Pitch*

Berikut merupakan perbandingan plot respon sistem pengendali PID dan *fuzzy*-PID dalam mengendalikan *pitch* pada simulasi *waypoint following* dengan gangguan arus laut ( $V_c = 0,2 \text{ m/s}$ ):



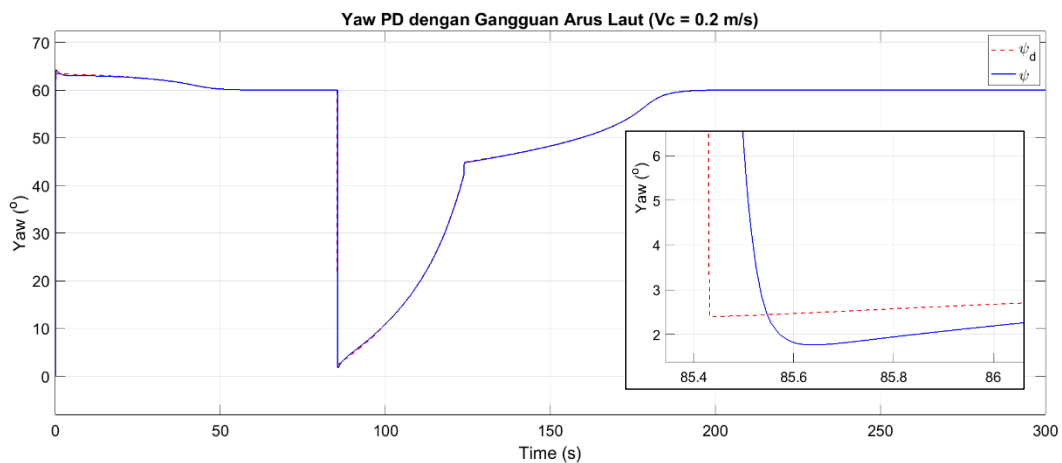
**Gambar 4. 39** Respon *Pitch* pada Simulasi *Waypoint Following* dengan Gangguan Arus Laut ( $V_c = 0,2 \text{ m/s}$ ) Menggunakan: (a) PID; (b) *Fuzzy*-PID

Dari Gambar 4.39 dapat dilihat bahwa pada waktu antara 18 hingga 22 sekon, selisih antara *pitch* yang diinginkan dengan *pitch* AUV yang terukur pada sistem

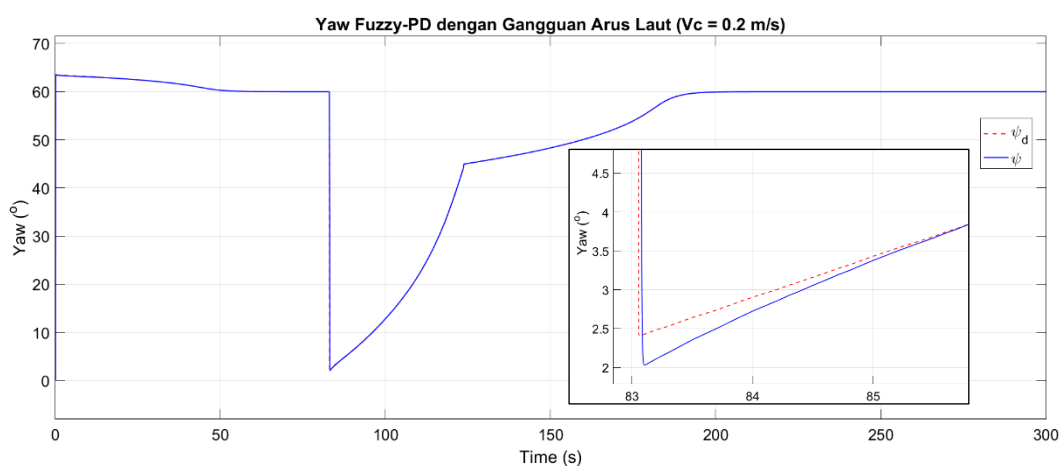
pengendali PID mencapai 0,02 sedangkan pada pengendali *fuzzy*-PID hanya kurang dari 0,01 sehingga dapat dibuktikan bahwa pengendali *fuzzy*-PID memiliki performa yang lebih baik bila dibandingkan pengendali PID dalam mengendalikan *pitch* di bawah gangguan arus laut dengan kecepatan 0,2 m/s.

#### 4.4.2.1.5 Perbandingan *Yaw*

Berikut merupakan perbandingan plot respon sistem pengendali PD dan *fuzzy*-PD dalam mengendalikan *yaw* pada simulasi *waypoint following* dengan gangguan arus laut ( $V_c = 0,2 \text{ m/s}$ ):



(a)



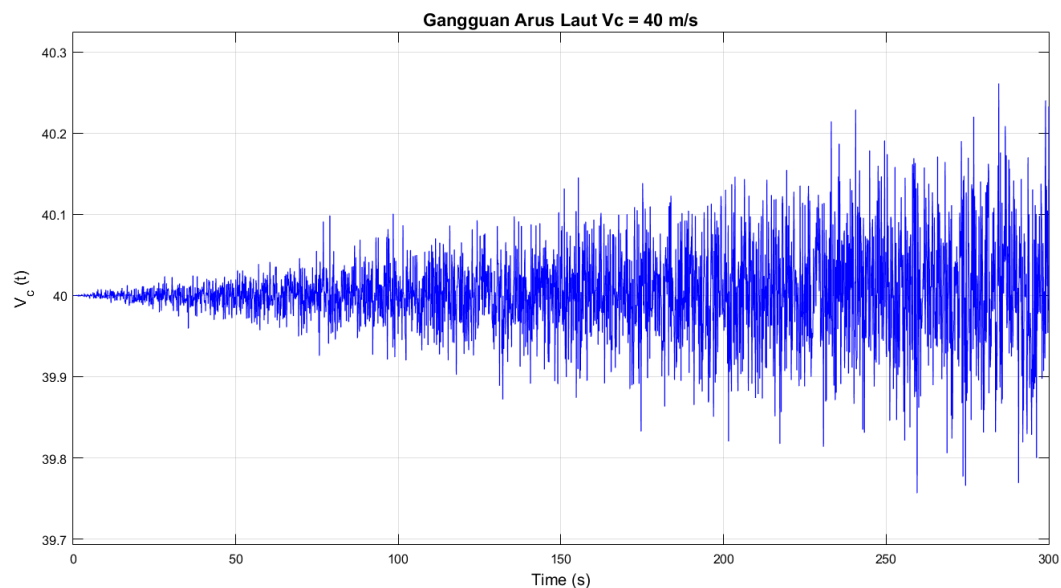
(b)

**Gambar 4. 40** Respon *Yaw* pada Simulasi *Waypoint Following* dengan Gangguan Arus Laut ( $V_c = 0,2 \text{ m/s}$ ) Menggunakan: (a) PD; (b) *Fuzzy*-PD

Dari Gambar 4.40 dapat dilihat bahwa pada waktu antara 83 hingga 86 sekon, selisih antara *yaw* yang diinginkan dengan *yaw* AUV yang terukur pada sistem pengendali PD mencapai 0,5 sedangkan pada pengendali *fuzzy*-PD hanya kurang dari 0,5 sehingga dapat dibuktikan bahwa pengendali *fuzzy*-PD memiliki performa yang lebih baik bila dibandingkan pengendali PD dalam mengendalikan *yaw* di bawah gangguan arus laut dengan kecepatan 0,2 m/s.

#### 4.4.2.2 Gangguan Arus 40 m/s

Gambar di bawah ini menunjukkan dinamika kecepatan arus laut terhadap waktu yang digunakan dalam simulasi. Adapun batas atas kecepatan arus laut ditetapkan 40,3 m/s sedangkan batas bawahnya adalah 39,7 m/s.



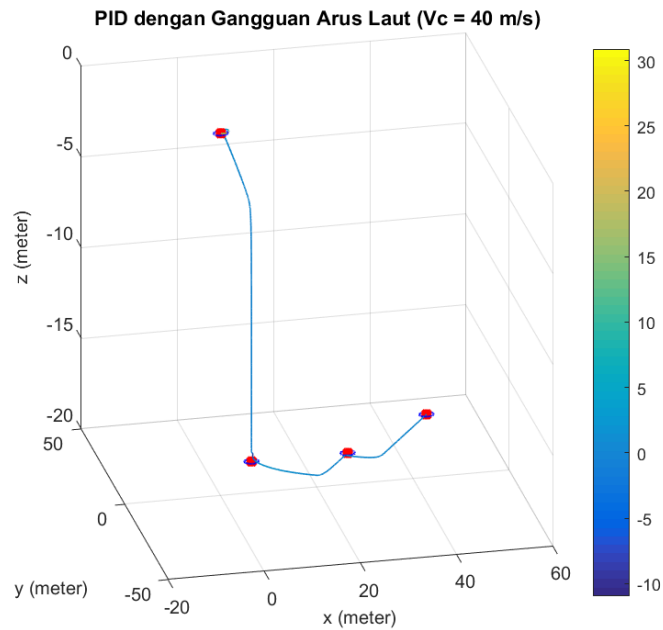
**Gambar 4. 41** Kecepatan Arus Laut terhadap Waktu (Variasi 40 m/s)

Berikut merupakan perbandingan lintasan dan performansi masing-masing kontroler di bawah gangguan arus laut sebesar 40 m/s.

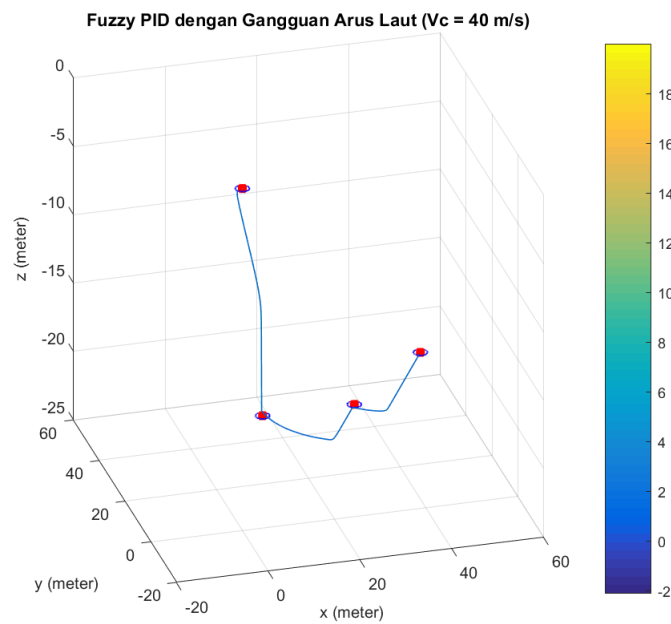
##### 4.4.2.2.1 Perbandingan Lintasan

Berikut merupakan perbandingan lintasan hasil simulasi *waypoint following* menggunakan sistem pengendali PID dan *fuzzy*-PID dengan adanya gangguan arus laut sebesar 40 m/s dan *side slip angle* 120°:

(a)

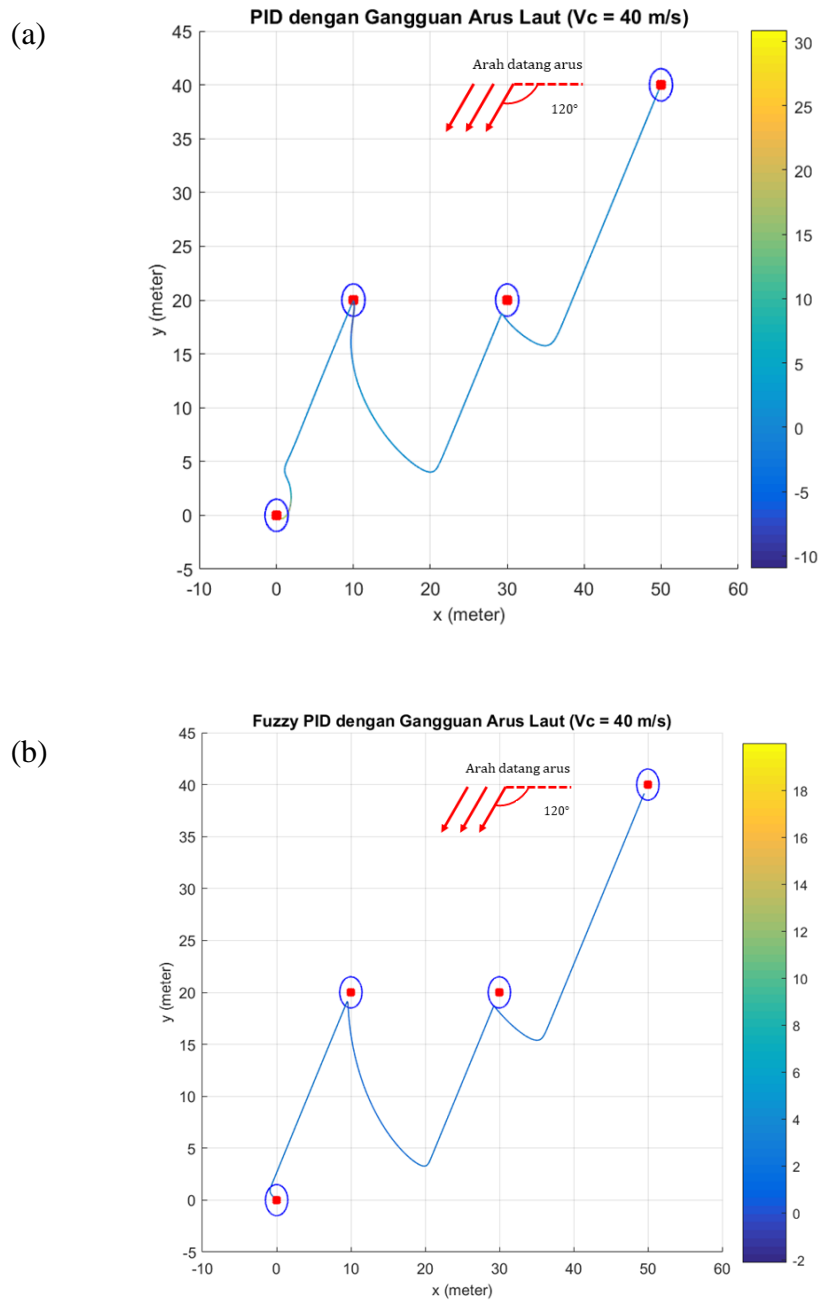


(b)



**Gambar 4. 42** Lintasan X-Y-Z Simulasi dengan Gangguan Arus Laut ( $V_c = 40$  m/s) Menggunakan: (a) PID; (b) *Fuzzy*-PID

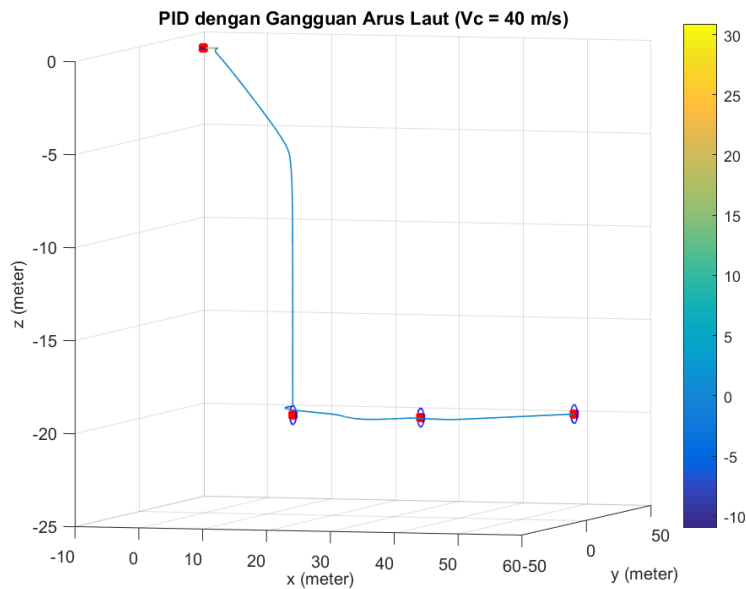
Gambar 4.42 menunjukkan bahwa kedua sistem pengendali mampu mencapai titik-titik koordinat yang dituju bila ditinjau secara 3D, namun dari kontur kecepatan dapat dilihat bahwa pengendali PID memiliki *overshoot* hingga 30 m/s sedangkan pengendali *fuzzy*-PID memiliki *overshoot* yang lebih rendah: 18 m/s.



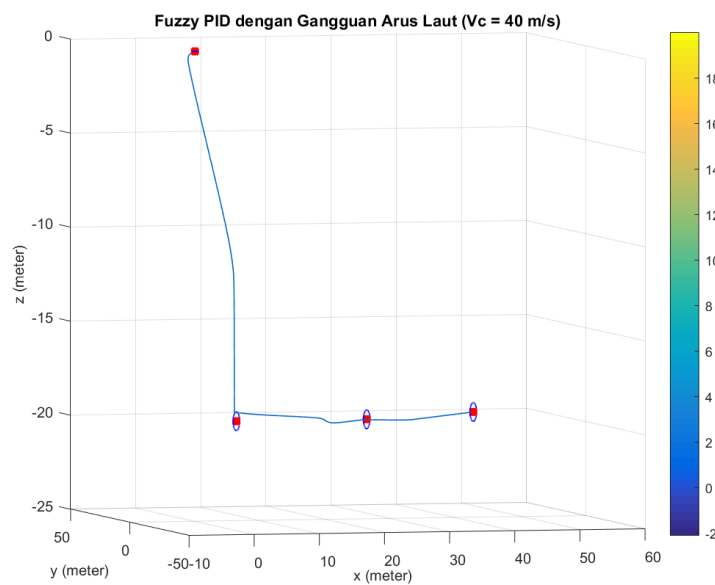
**Gambar 4. 43** Lintasan X-Y Simulasi dengan Gangguan Arus Laut ( $V_c = 40$  m/s) Menggunakan: (a) PID; (b) *Fuzzy*-PID

Gambar 4.43 menunjukkan bahwa kedua sistem pengendali mampu mencapai menggerakkan AUV hingga memasuki COA bidang X-Y, namun pada pengendali PID ketika AUV meninggalkan titik awal terjadi pembelokan yang lebih jauh dibandingkan pada pengendali *fuzzy*-PID.

(a)



(b)



**Gambar 4. 44** Lintasan X-Z Simulasi dengan Gangguan Arus Laut ( $V_c = 40$  m/s) Menggunakan: (a) PID; (b) *Fuzzy*-PID

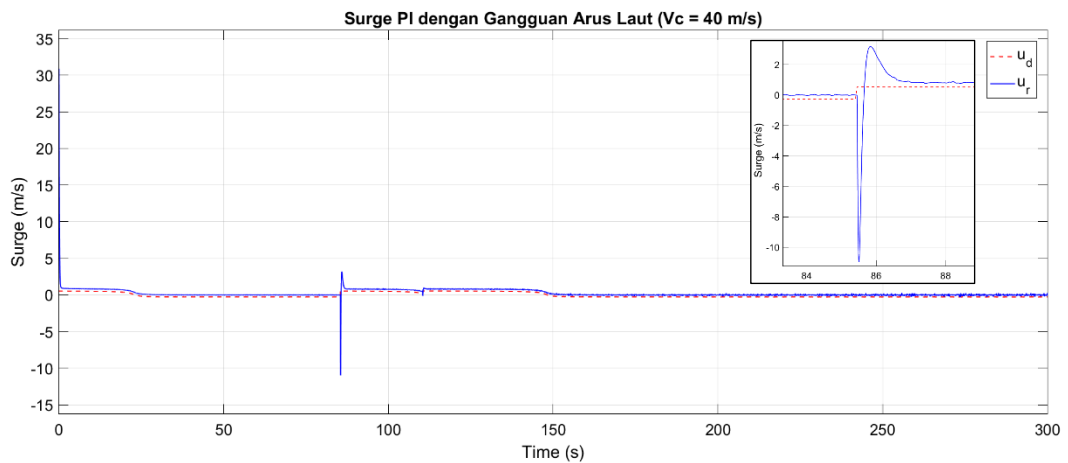
Gambar 4.44 menunjukkan bahwa kedua sistem pengendali mampu mencapai menggerakkan AUV hingga memasuki COA bidang X-Z, namun karena *overshoot* kecepatan pada PID lebih besar maka algoritma *waypoint following* menentukan



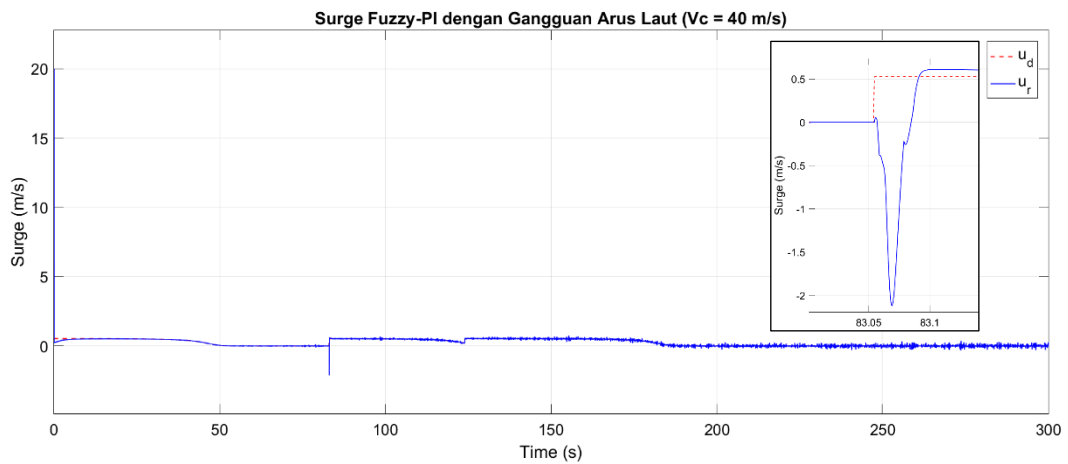
nilai *pitch* pada PID di awal keberangkatan lebih besar sehingga lintasan menyelim tampak lebih menukik dibandingkan pada *fuzzy*-PID.

#### 4.4.2.2 Perbandingan Surge

Berikut merupakan perbandingan plot respon sistem pengendali PI dan *fuzzy*-PI dalam mengendalikan *surge* pada simulasi *waypoint following* dengan gangguan arus laut ( $V_c = 40 \text{ m/s}$ ):



(a)



(b)

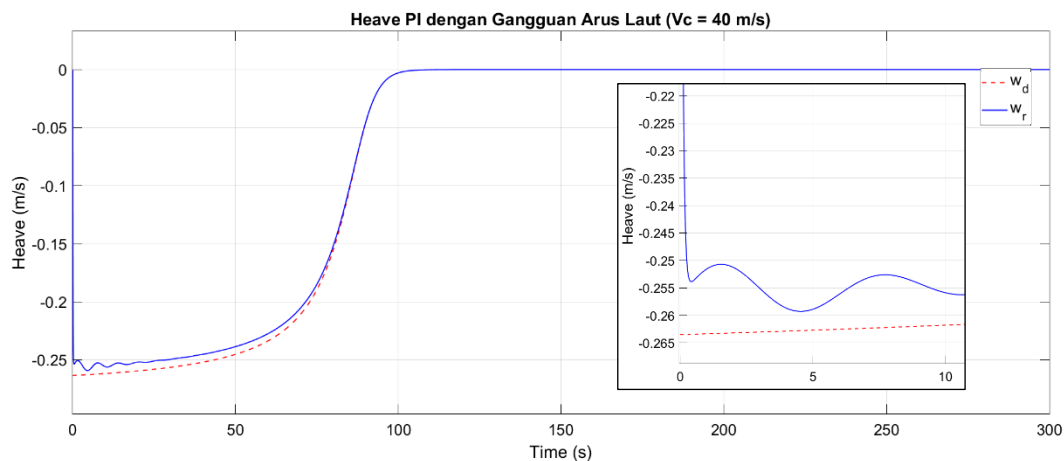
**Gambar 4. 45** Respon *Surge* pada Simulasi *Waypoint Following* dengan Gangguan Arus Laut ( $V_c = 40 \text{ m/s}$ ) Menggunakan: (a) PI; (b) *Fuzzy*-PI

Dari Gambar 4.45 dapat dilihat bahwa *overshoot* pada respon sistem pengendali PI mencapai 35 m/s ketika  $t=0$  sekon, sedangkan pada *fuzzy*-PI hanya mencapai 20 m/s. Selain itu, *fuzzy*-PI menunjukkan respon sistem yang lebih cepat terhadap perubahan *setpoint*, seperti tampak dalam gambar pada rentang waktu 83-

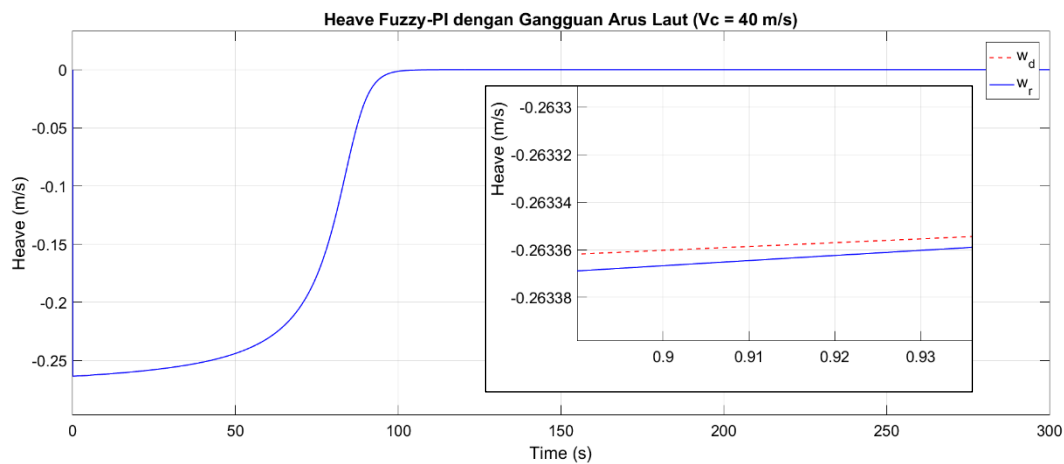
84 sekon. Hal ini menunjukkan bahwa pengendali *fuzzy*-PI memiliki performa yang lebih baik dalam mengendalikan *surge* dibandingkan pengendali PI.

#### 4.4.2.2.3 Perbandingan *Heave*

Berikut merupakan perbandingan plot respon sistem pengendali PI dan *fuzzy*-PI dalam mengendalikan *heave* pada simulasi *waypoint following* dengan gangguan arus laut ( $V_c = 40 \text{ m/s}$ ):



(a)



(b)

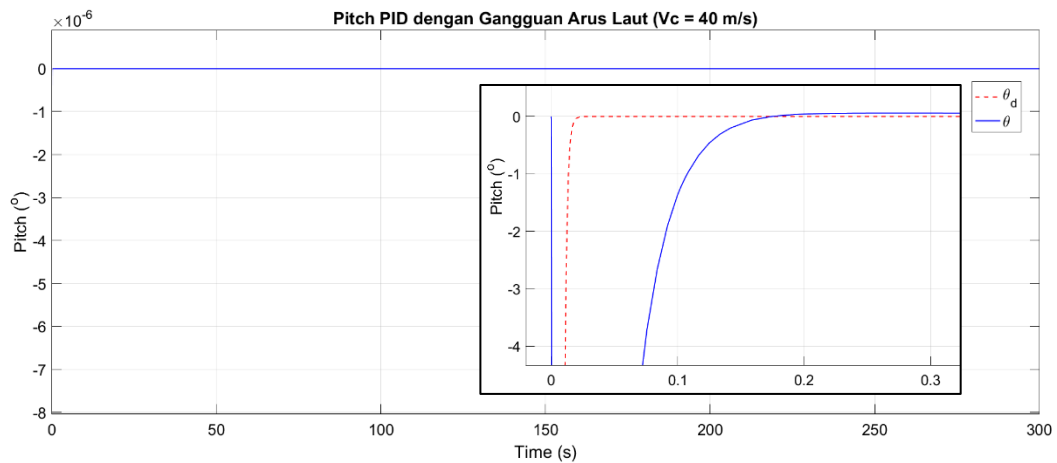
**Gambar 4. 46** Respon *Heave* pada Simulasi *Waypoint Following* dengan Gangguan Arus Laut ( $V_c = 40 \text{ m/s}$ ) Menggunakan: (a) PI; (b) *Fuzzy*-PI

Dari Gambar 4.46 dapat dilihat bahwa pada waktu antara 0 hingga 1 sekon, selisih antara *heave* yang diinginkan dengan *heave* AUV yang terukur pada sistem pengendali PI mencapai 0,01 sedangkan pada pengendali *fuzzy*-PI hanya 0,00001 sehingga dapat dibuktikan bahwa pengendali *fuzzy*-PI memiliki performa yang

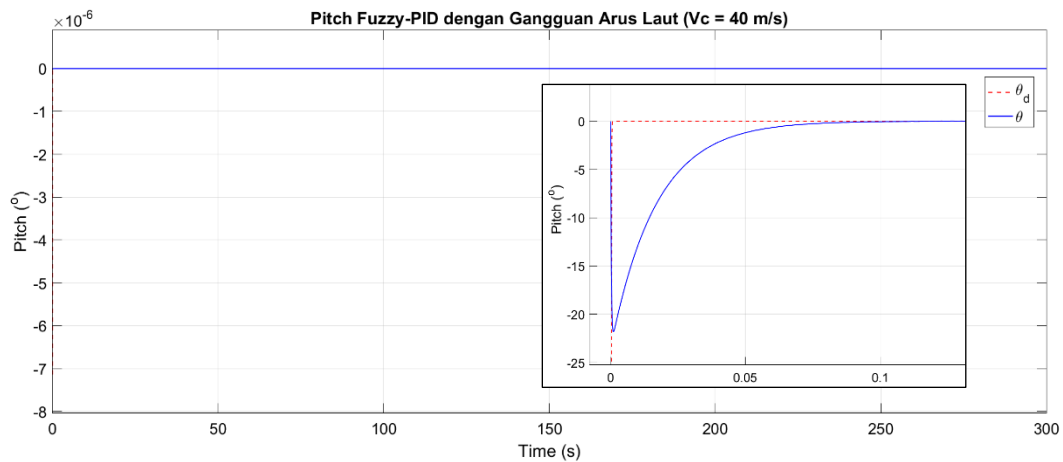
lebih baik bila dibandingkan pengendali PI dalam mengendalikan *heave* di bawah gangguan arus laut dengan kecepatan 40 m/s.

#### 4.4.2.2.4 Perbandingan *Pitch*

Berikut merupakan perbandingan plot respon sistem pengendali PID dan *fuzzy*-PID dalam mengendalikan *pitch* pada simulasi *waypoint following* dengan gangguan arus laut ( $V_c = 40 \text{ m/s}$ ):



(a)



(b)

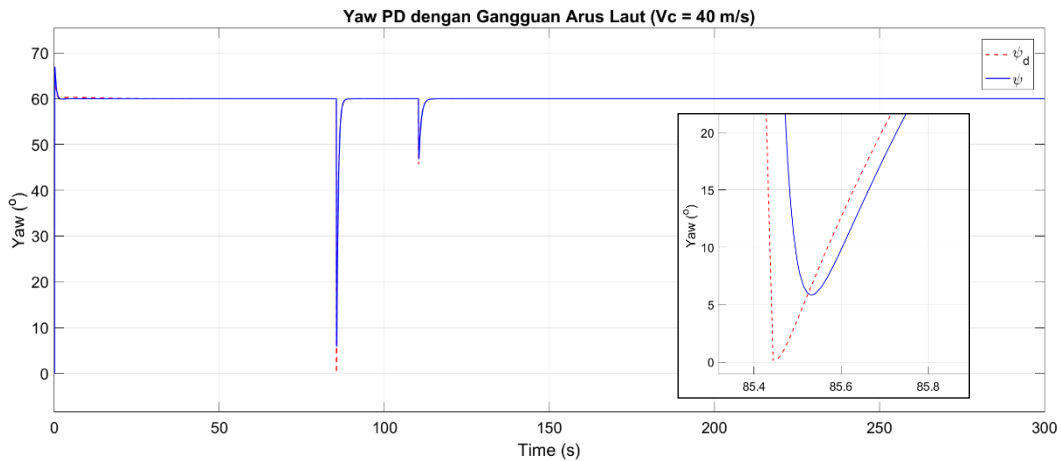
**Gambar 4. 47** Respon *Pitch* pada Simulasi *Waypoint Following* dengan Gangguan Arus Laut ( $V_c = 40 \text{ m/s}$ ) Menggunakan: (a) PID; (b) *Fuzzy*-PID

Pada Gambar 4.47 dapat dilihat bahwa sekilas nilai *pitch* tampak nihil karena seperti tampak pada Gambar 4.44, lintasan kontur kecepatan AUV hampir selalu berwarna biru yang menunjukkan bahwa AUV bergerak dalam kecepatan rendah. Apabila AUV bergerak dalam kecepatan rendah, maka algoritma *waypoint*

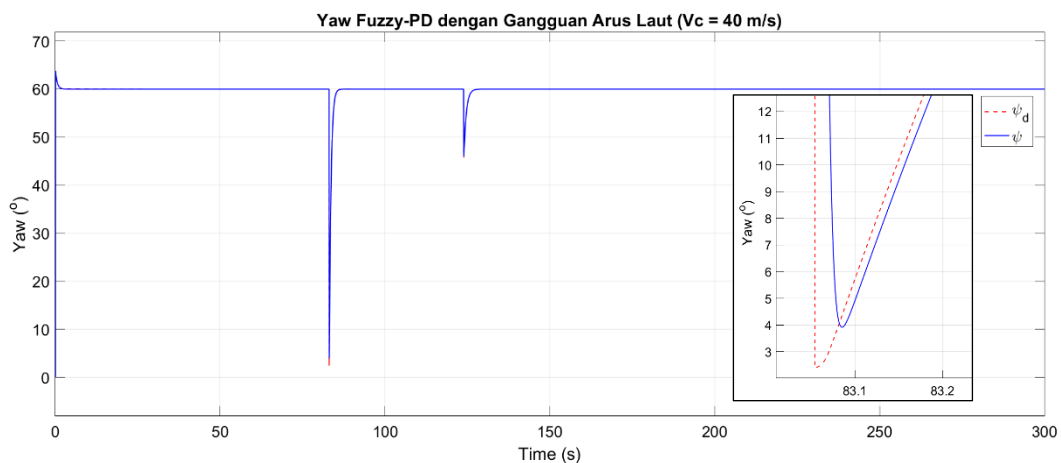
*following* akan menggunakan *heave* untuk mencapai kedalaman yang diinginkan karena lebih efektif. Namun dapat dilihat bahwa pengendali PID memiliki *error steady state* yang lebih besar dibandingkan pengendali *fuzzy*-PID sehingga dapat disimpulkan bahwa *fuzzy*-PID masih lebih unggul dibandingkan PID dalam mengendalikan *pitch* di bawah gangguan arus laut ekstrim dengan kecepatan 40 m/s.

#### 4.4.2.2.5 Perbandingan *Yaw*

Berikut merupakan perbandingan plot respon sistem pengendali PD dan *fuzzy*-PD dalam mengendalikan *yaw* pada simulasi *waypoint following* dengan gangguan arus laut ( $V_c = 40 \text{ m/s}$ ):



(a)



(b)

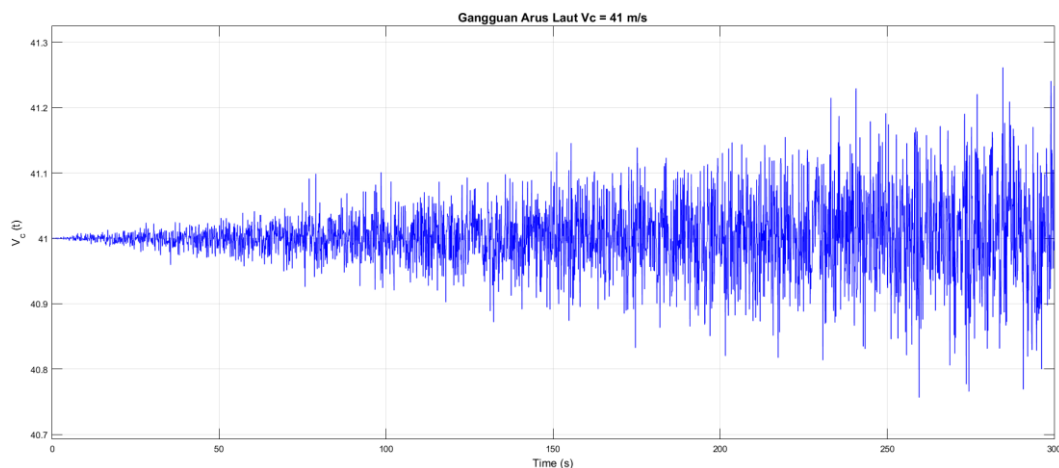
**Gambar 4. 48** Respon *Yaw* pada Simulasi *Waypoint Following* dengan Gangguan Arus Laut ( $V_c = 40 \text{ m/s}$ ) Menggunakan: (a) PD; (b) *Fuzzy*-PD

Dari Gambar 4.48 dapat dilihat bahwa pada waktu antara 83 hingga 86 sekon, selisih antara *yaw* yang diinginkan dengan *yaw* AUV yang terukur pada sistem pengendali PD mencapai 5 derajat lebih sedangkan pada pengendali *fuzzy*-PD hanya 1,2 derajat sehingga dapat dibuktikan bahwa pengendali *fuzzy*-PD memiliki performa yang lebih baik bila dibandingkan pengendali PD dalam mengendalikan *yaw* di bawah gangguan arus laut dengan kecepatan 40 m/s.

#### 4.4.2.3 Gangguan Arus 41 m/s

Simulasi menggunakan variasi gangguan arus 41 m/s bertujuan untuk membandingkan ketahanan dari kedua sistem pengendali yang telah dirancang apabila beroperasi di bawah gangguan ekstrim.

Gambar di bawah ini menunjukkan dinamika kecepatan arus laut terhadap waktu yang digunakan dalam simulasi. Adapun batas atas kecepatan arus laut ditetapkan 41,3 m/s sedangkan batas bawahnya adalah 40,7 m/s.



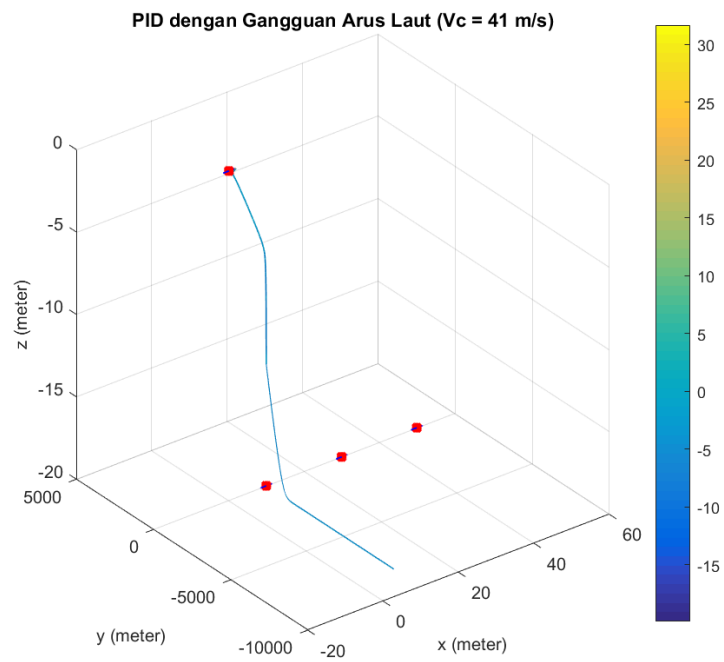
**Gambar 4. 49** Kecepatan Arus Laut terhadap Waktu (Variasi 41 m/s)

Berikut merupakan perbandingan lintasan dan performansi masing-masing kontroler di bawah gangguan arus laut sebesar 41m/s.

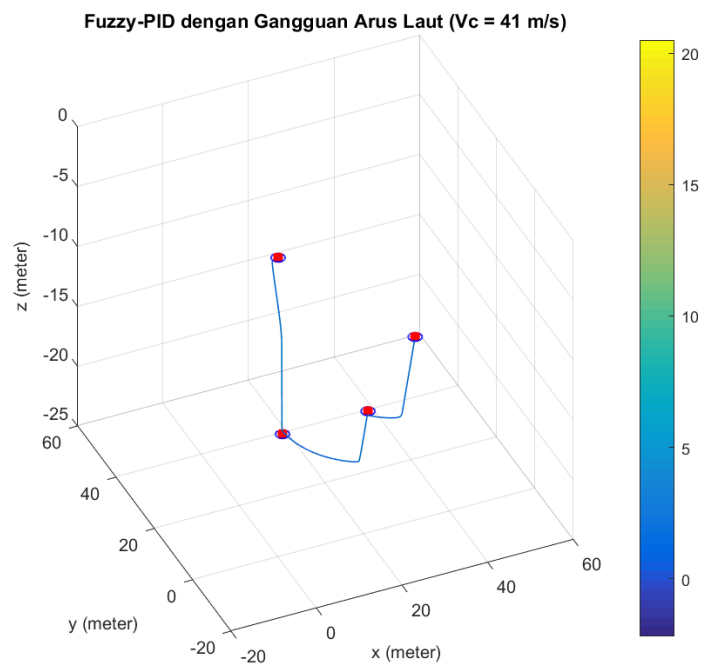
##### 4.4.2.3.1 Perbandingan Lintasan

Berikut merupakan perbandingan lintasan hasil simulasi *waypoint following* menggunakan sistem pengendali PID dan *fuzzy*-PID dengan adanya gangguan arus laut sebesar 41 m/s dan *side slip angle* 120°:

(a)

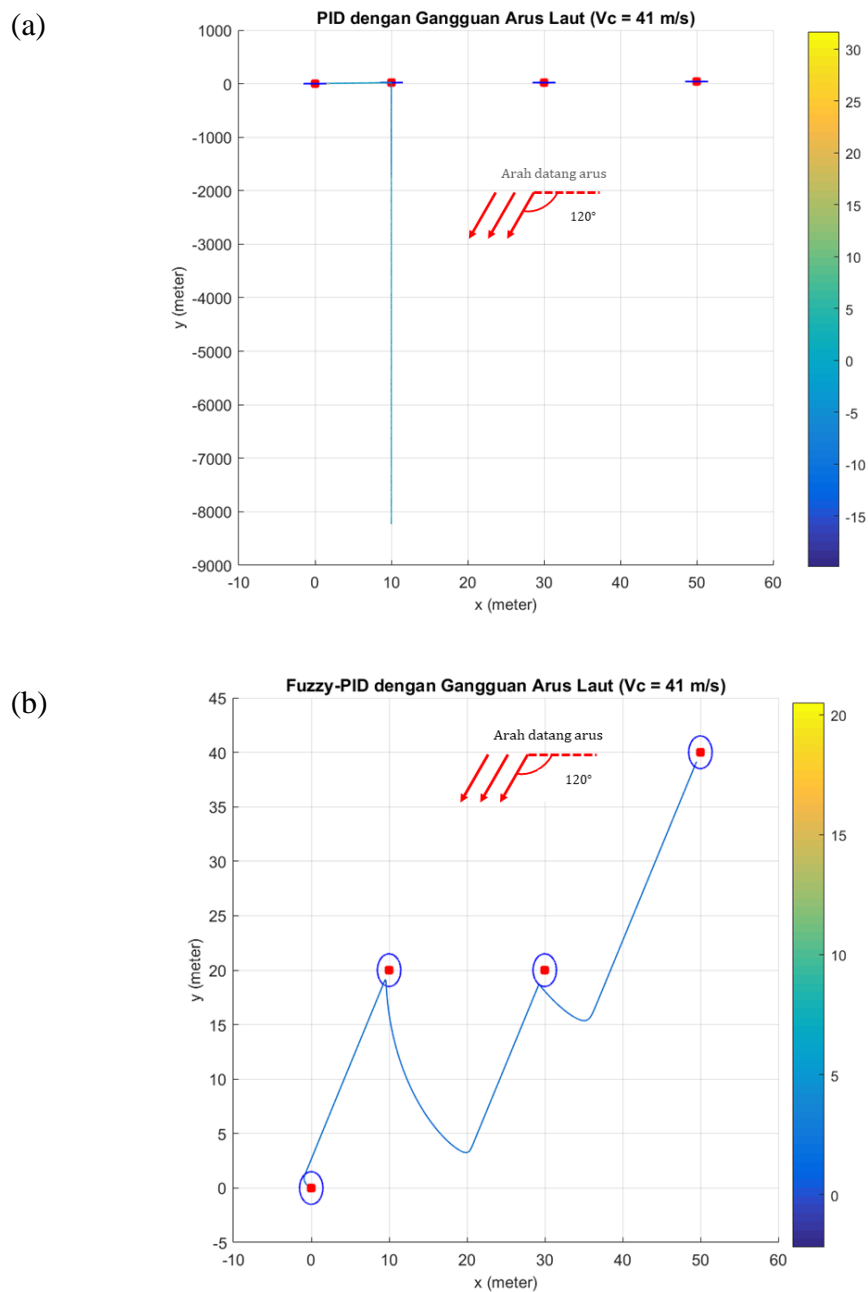


(b)



**Gambar 4. 50** Lintasan X-Y-Z Simulasi dengan Gangguan Arus Laut ( $V_c = 41$  m/s) Menggunakan: (a) PID; (b) *Fuzzy*-PID

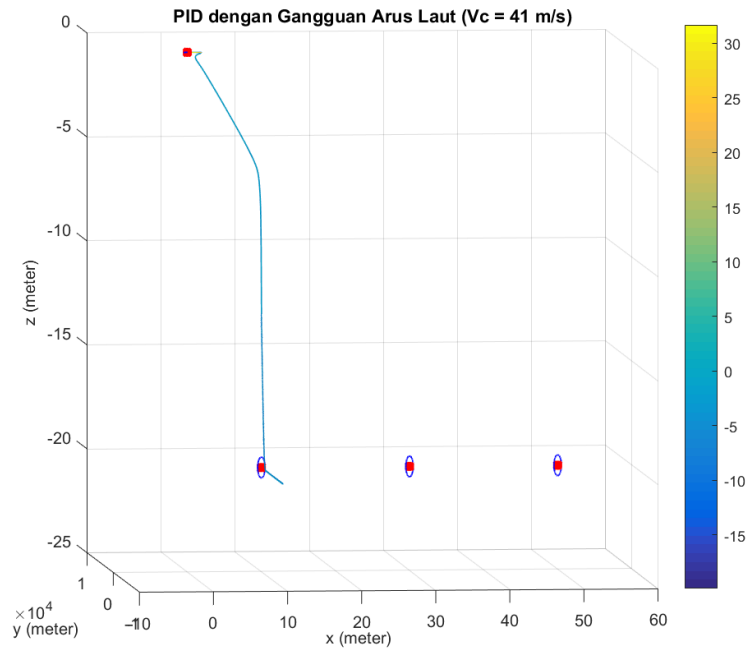
Dari Gambar 4.50 dapat dilihat bahwa dengan gangguan arus laut berkecepatan 41 m/s, sistem pengendali PID tidak lagi dapat bekerja dengan baik, namun sistem pengendali *fuzzy*-PID masih mampu menggerakkan AUV menuju titik koordinat yang dituju.



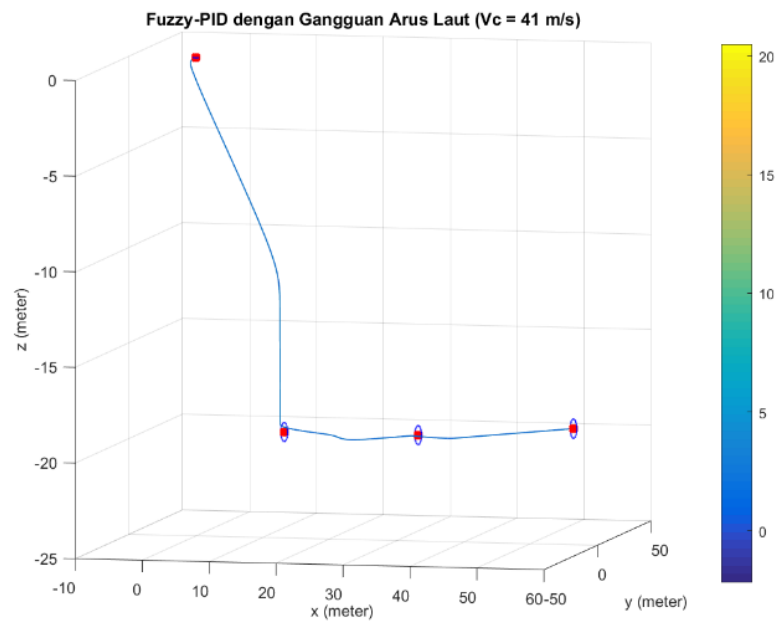
**Gambar 4. 51** Lintasan X-Y Simulasi dengan Gangguan Arus Laut ( $V_c = 41 \text{ m/s}$ ) Menggunakan: (a) PID; (b) *Fuzzy*-PID

Gambar 4.51 menunjukkan bahwa sistem pengendali *fuzzy*-PID yang dirancang masih mampu mencapai COA di bidang X-Y, sedangkan sistem pengendali PID sudah tidak mampu mencapai COA di bidang X-Y.

(a)



(b)



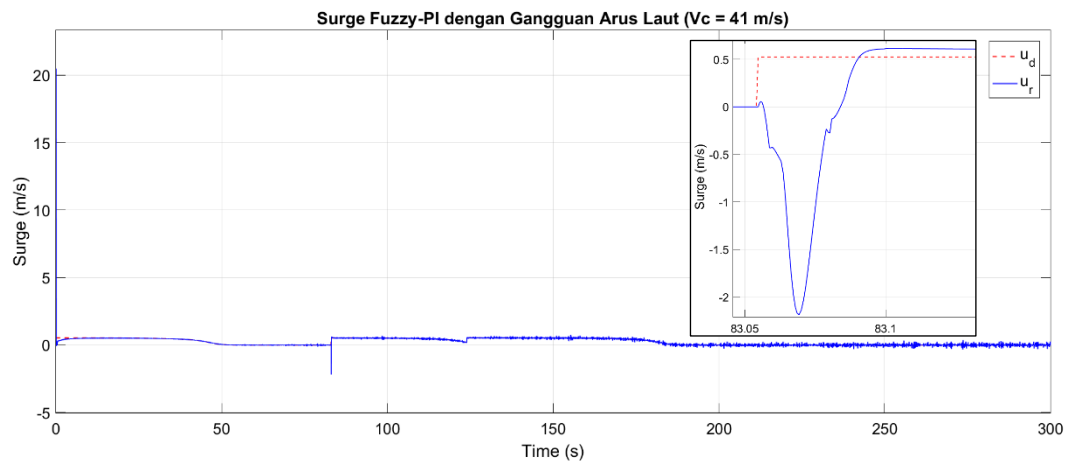
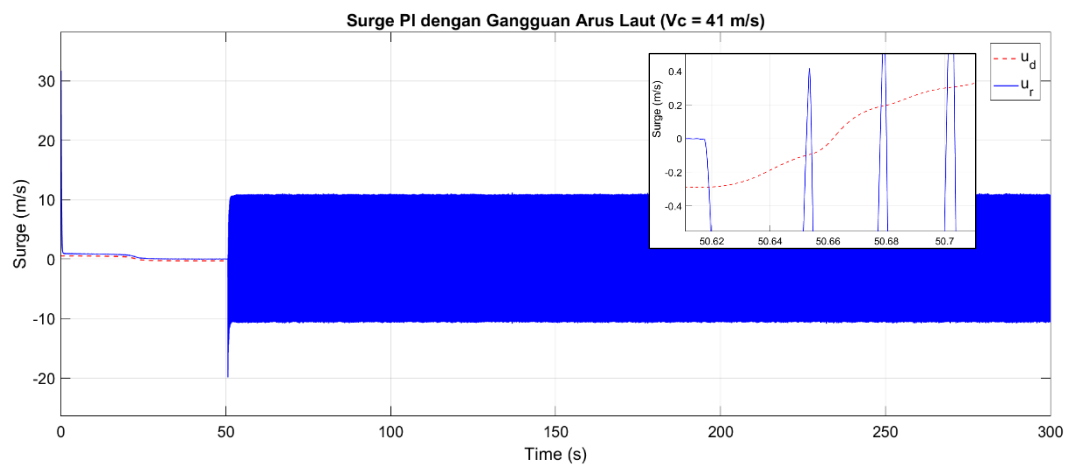
**Gambar 4. 52** Lintasan X-Z Simulasi dengan Gangguan Arus Laut ( $V_c = 41 \text{ m/s}$ ) Menggunakan: (a) PID; (b) *Fuzzy*-PID



Gambar 4.52 menunjukkan bahwa baik sistem pengendali *fuzzy*-PID maupun PID masih mampu mencapai COA di bidang X-Z, namun untuk PID hanya pada titik *waypoint* pertama saja.

#### 4.4.2.3.2 Perbandingan *Surge*

Berikut merupakan perbandingan plot respon sistem pengendali PI dan *fuzzy*-PI dalam mengendalikan *surge* pada simulasi *waypoint following* dengan gangguan arus laut ( $V_c = 41 \text{ m/s}$ ):

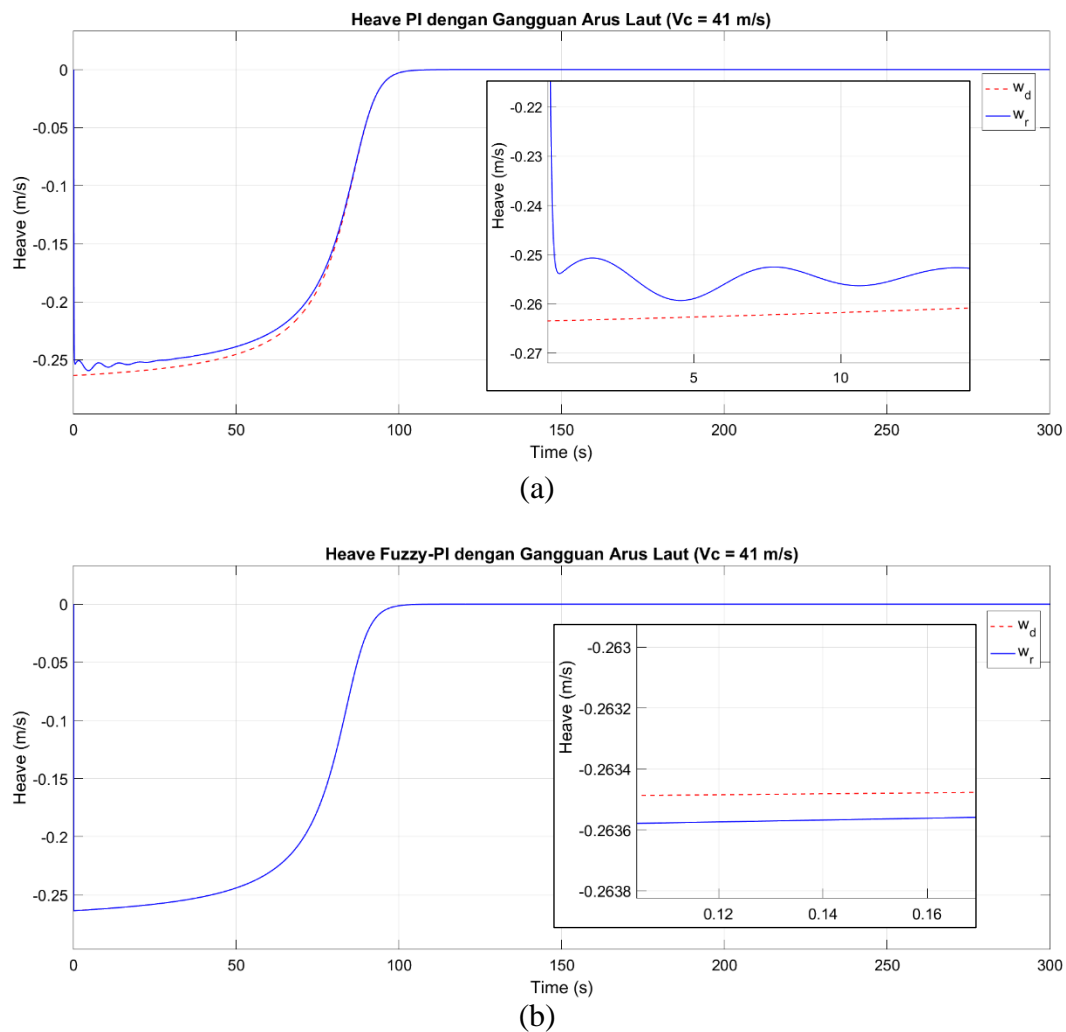


**Gambar 4. 53** Respon *Surge* pada Simulasi *Waypoint Following* dengan Gangguan Arus Laut ( $V_c = 41 \text{ m/s}$ ) Menggunakan: (a) PI; (b) *Fuzzy*-PI

Gambar 4.53 menunjukkan bahwa pengendali *fuzzy*-PI memiliki performa yang lebih baik karena masih mampu mengendalikan *surge* di bawah gangguan ekstrim dibandingkan pengendali PI tidak mampu lagi mengendalikan osilasi.

#### 4.4.2.3.3 Perbandingan *Heave*

Berikut merupakan perbandingan plot respon sistem pengendali PI dan *fuzzy*-PI dalam mengendalikan *heave* pada simulasi *waypoint following* dengan gangguan arus laut ( $V_c = 41 \text{ m/s}$ ):



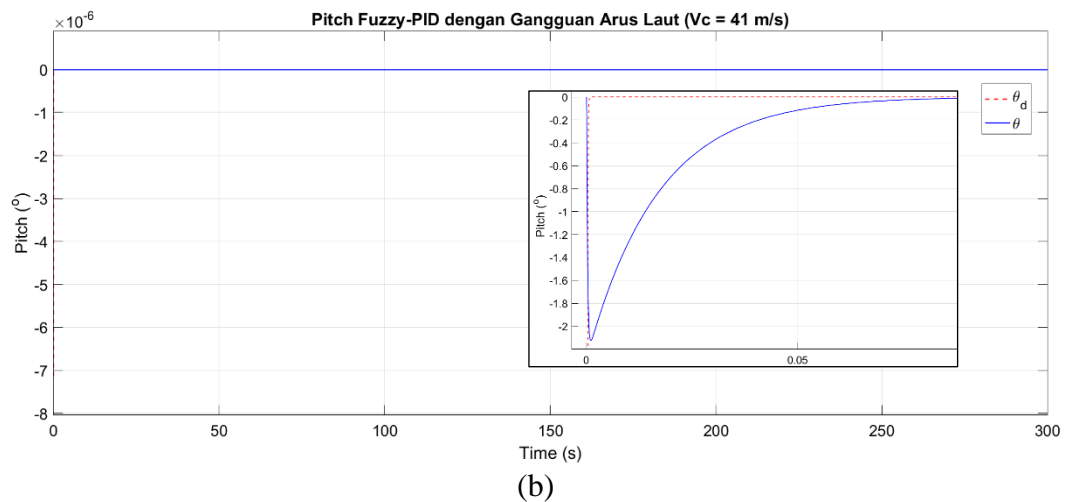
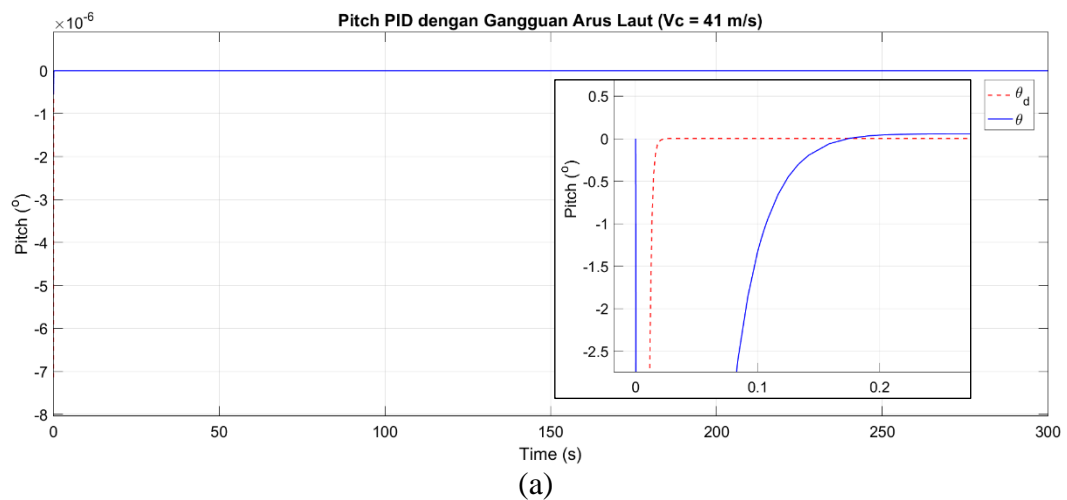
**Gambar 4. 54** Respon *Heave* pada Simulasi *Waypoint Following* dengan Gangguan Arus Laut ( $V_c = 41 \text{ m/s}$ ) Menggunakan: (a) PI; (b) *Fuzzy*-PI

Dari Gambar 4.54 dapat dilihat bahwa pada waktu antara 0 hingga 1 detik, selisih antara *heave* yang diinginkan dengan *heave* AUV yang terukur pada sistem pengendali PI mencapai 0,01 sedangkan pada pengendali *fuzzy*-PI hanya 0,0001

sehingga dapat dibuktikan bahwa pengendali *fuzzy*-PI memiliki performa yang lebih baik bila dibandingkan pengendali PI dalam mengendalikan *heave* di bawah gangguan arus laut ekstrim dengan kecepatan 41 m/s.

#### 4.4.2.3.4 Perbandingan *Pitch*

Berikut merupakan perbandingan plot respon sistem pengendali PID dan *fuzzy*-PID dalam mengendalikan *pitch* pada simulasi *waypoint following* dengan gangguan arus laut ( $V_c = 41 \text{ m/s}$ ):



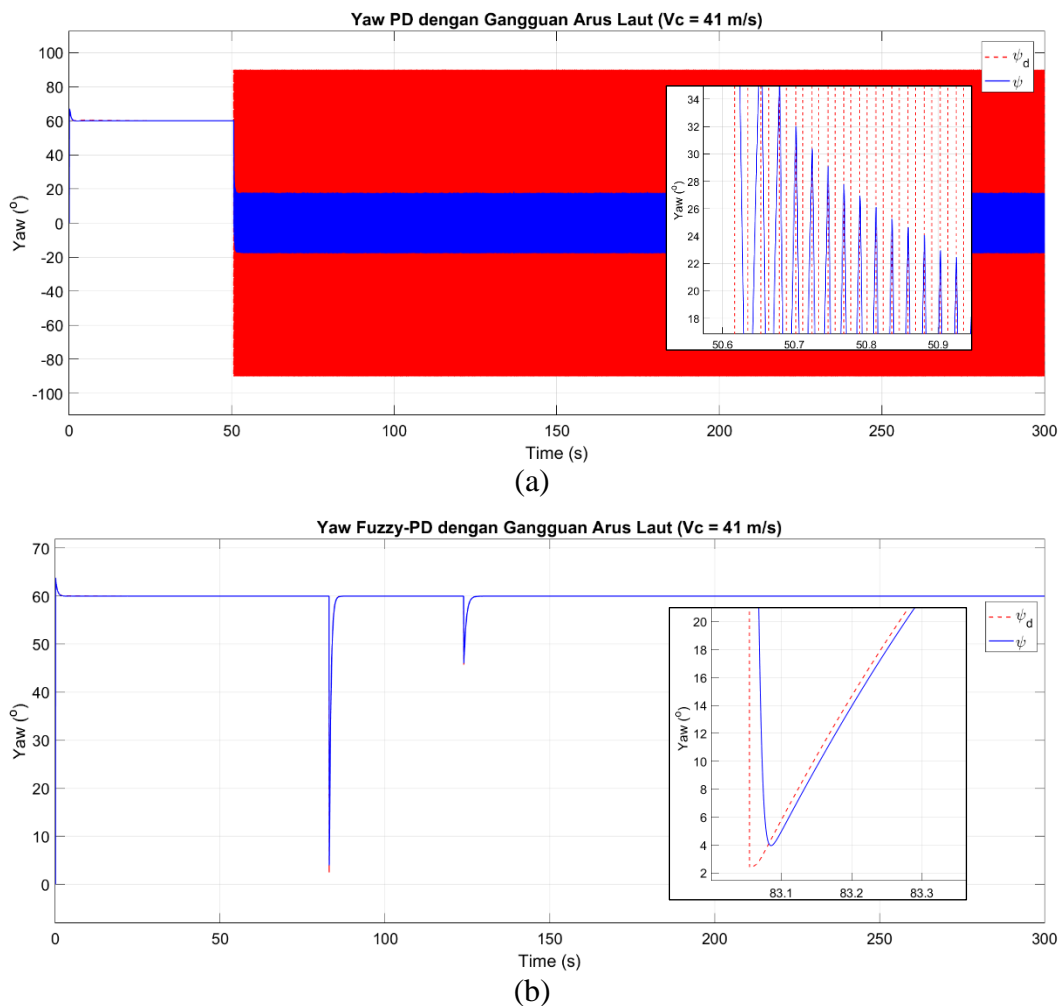
**Gambar 4. 55** Respon *Pitch* pada Simulasi *Waypoint Following* dengan Gangguan Arus Laut ( $V_c = 41 \text{ m/s}$ ) Menggunakan: (a) PID; (b) *Fuzzy*-PID

Pada Gambar 4.55 dapat dilihat bahwa sekilas nilai *pitch* tampak nihil karena seperti tampak pada Gambar 4.52, lintasan kontur kecepatan AUV hampir selalu berwarna biru yang menunjukkan bahwa AUV bergerak dalam kecepatan

rendah. Apabila AUV bergerak dalam kecepatan rendah, maka algoritma *waypoint following* akan menggunakan *heave* untuk mencapai kedalaman yang diinginkan karena lebih efektif. Namun dapat dilihat bahwa pengendali PID memiliki *error steady state* yang lebih besar dibandingkan pengendali *fuzzy*-PID sehingga dapat disimpulkan bahwa *fuzzy*-PID masih lebih unggul dibandingkan PID dalam mengendalikan *pitch* di bawah gangguan arus laut ekstrim dengan kecepatan 41 m/s.

#### 4.4.2.3.5 Perbandingan Yaw

Berikut merupakan perbandingan plot respon sistem pengendali PD dan *fuzzy*-PD dalam mengendalikan *yaw* pada simulasi *waypoint following* dengan gangguan arus laut ( $V_c = 41 \text{ m/s}$ ):



**Gambar 4. 56** Respon Yaw pada Simulasi *Waypoint Following* dengan Gangguan Arus Laut ( $V_c = 41 \text{ m/s}$ ) Menggunakan: (a) PD; (b) *Fuzzy*-PD

Gambar 4.56 menunjukkan bahwa pengendali *fuzzy*-PD memiliki performa yang lebih baik karena dapat menyamai kurva *setpoint*, bahkan dengan selisih nilai kurang dari 2 derajat sedangkan pengendali PD sudah tidak mampu mengendalikan osilasi yang terjadi.

Berdasarkan perbandingan yang telah disebutkan di atas, dapat disimpulkan bahwa performansi dari sistem pengendali AUV tipe *double-hull* menggunakan *fuzzy*-PID unggul dibandingkan performansi sistem pengendali AUV menggunakan PID dalam hal mengatasi adanya nonlinearitas dan gangguan eksternal seperti arus laut, karena masih mampu mencapai *circle of acceptance* dari *waypoint* yang dituju di bawah gangguan arus laut ekstrim dengan kecepatan  $V_c = 41$  m/s.

*Halaman ini sengaja dikosongkan*

## BAB V

### KESIMPULAN DAN SARAN

#### 5.1 Kesimpulan

Berdasarkan data dan analisa yang telah dilakukan, ada beberapa poin kesimpulan yang dapat diambil dari penelitian ini, yakni:

- Perancangan sistem pengendali AUV tipe *double-hull* menggunakan *fuzzy*-PID dapat dilakukan dengan menentukan parameter awal dari *gain* proporsional, integral, dan derivatif dari PID kemudian membangun arsitektur logika *fuzzy* untuk menentukan nilai *gain* yang harus ditambahkan ( $dK_p, dK_i, dK_d$ ) kepada kontroler berdasarkan masukan berupa *error* dan *error rate*.
- Penentuan parameter terbaik untuk sistem pengendali logika *fuzzy* dapat dilakukan dengan melakukan iterasi terhadap *range tuning* parameter *gain* proporsional, integral, dan derivatif pada PID kemudian dilakukan analisa performansi berdasarkan respon sistem. Untuk pengendali *surge*, respon terbaik dengan *overshoot* 0,07% dan  $e_{ss}$  9,14E-05 didapatkan oleh *range tuning* parameter *gain* sebesar 100 kali lipat dari nilai *gain* pada kontroler PI. Untuk pengendali *heave*, respon terbaik dengan *overshoot* 0,05% dan  $e_{ss}$  2,05E-05 didapatkan oleh *range tuning* parameter *gain* sebesar 100 kali lipat dari nilai *gain* pada kontroler PI. Untuk pengendali *pitch*, respon terbaik dengan *overshoot* 1,79% dan  $e_{ss}$  7,61E-04 didapatkan oleh *range tuning* parameter *gain* sebesar 1 kali lipat dari nilai *gain* pada kontroler PID. Untuk pengendali *yaw*, respon terbaik dengan *overshoot* 0,24% dan  $e_{ss}$  1,30E-04 didapatkan oleh *range tuning* parameter *gain* sebesar 10 kali lipat dari nilai *gain* pada kontroler PD.
- Performansi dari sistem pengendali AUV tipe *double-hull* menggunakan *fuzzy*-PID unggul dibandingkan performansi sistem pengendali AUV menggunakan PID dalam hal mengatasi adanya gangguan eksternal seperti arus laut, karena masih mampu mencapai *circle of acceptance* dari *waypoint* yang dituju di bawah gangguan arus laut dengan kecepatan  $V_c = 41$  m/s.

## 5.2 Saran

Penelitian ini masih dapat dikembangkan lebih lanjut dengan cara mengintegrasikan sistem pengendali dengan *path following guidance*. Selain itu, dinamika aktuator dan sensor yang tidak dipertimbangkan dalam penelitian ini juga dapat dimodelkan. Algoritma adaptif lainnya seperti *fuzzy* tipe 2 atau jaringan saraf tiruan juga dapat diuji-cobakan untuk memperoleh sistem pengendali dengan performansi terbaik.



## DAFTAR PUSTAKA

- Alaoui, E. C., Ayad, H., & Doubabi, S. (2006). Fuzzy Anti-Windup Schemes for PID Controller. *International Journal of Applied Engineering Research*, 295-306.
- Åström, K. J., & Hägglund, T. (2005). *Advanced PID Control*. ISA - Instrumentation, Systems, and Automation Society, Research Triangle Park, NC 27709.
- Astrom, K. J., & Rundqwist, L. (1989). Integrator Windup and How to Avoid It. *American Control Conference*. Pittsburgh, PA, USA: IEEE.
- Åström, K. J., & Wittenmark, B. (2011). *Computer-Controlled Systems*. Courier Corporation.
- Bazoune, A. (n.d.). *Chapter 10: Time-Domain Analysis and Designs of Control Systems*. Retrieved from King Fahd University of Petroleum and Minerals: <https://faculty.kfupm.edu.sa/me/qahtanih/ME413Note/Chapter10.pdf>
- Bentes, C. A. (2016). *Modeling of an Autonomous Underwater Vehicle*. Covilha: University of Beira Interior Engineering.
- Bolton, W. (2004). *Instrumentation and Control Systems*. Elsevier. doi:<https://doi.org/10.1016/B978-0-7506-6432-5.X5000-1>
- Bolton, W. (2021). *Instrumentation and Control Systems (Chapter 10 - System Response)* (3rd ed.). Newnes. doi:<https://doi.org/10.1016/B978-0-12-823471-6.00010-1>
- Driscoll, F. R., Alsenas, G. M., Beaujean, P. P., Ravenna, S., Raveling, J., Busold, E., & Slezycski, C. (2008). A 20 KW Open Ocean Current Test Turbine. *OCEANS* (pp. 1-6). Quebec City, Canada: IEEE. doi:10.1109/OCEANS.2008.5152104
- Electrical4U. (2021, January 24). *Bode Plot, Gain Margin and Phase Margin (Plus Diagrams)*. Retrieved from Electrical4U: <https://www.electrical4u.com/bode-plot-gain-margin-phase->

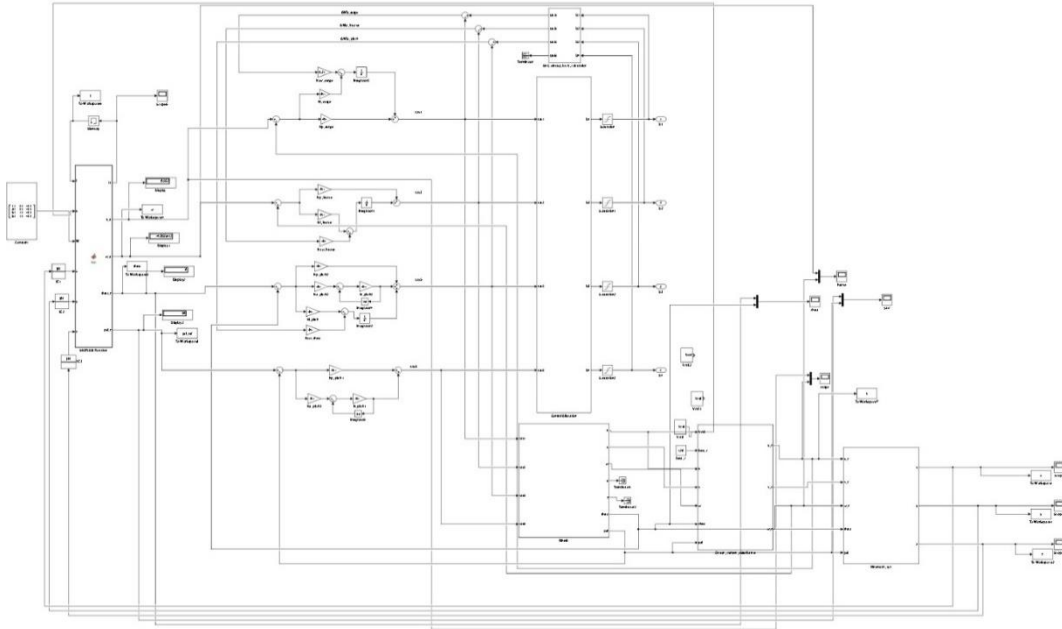


- Jennings, D. (n.d.). Performance of Feedback Control Systems.
- Kang, S., Yu, J., Zhang, J., & Jin, Q. (2020). Development of Multibody Marine Robots: A Review. *IEEE Access*, 21178-21195.
- Keller, J. (2016, December 15). Navy Taps Metron for Advanced Development of Machine Intelligence for UUV Surveillance. Arlington, Texas, United States of America. Retrieved July 14, 2022, from <https://www.militaryaerospace.com/computers/article/16714822/navy-taps-metron-for-advanced-development-of-machine-intelligence-for-uuv-surveillance>
- Khodayari, M. H., & Balochian, S. (2015). Modeling and Control of Autonomous Underwater Vehicle (AUV) in Heading and Depth Attitude via Self-Adaptive Fuzzy PID Controller. *Journal of Marine Science Technology*.
- Mendes, C. H. (2017). *Robust Controller Design for an Autonomous Underwater Vehicle*. Covilha: Universidade Da Beira Interior.
- Ogata, K. (2002). *Modern Control Engineering*. New Jersey: Prentice-Hall, Inc.
- Petsko, E. (2020, June 8). *Why Does So Much of the Ocean Remain Unexplored and Unprotected?* Retrieved January 11, 2022, from OCEANA: <https://oceana.org/blog/why-does-so-much-ocean-remain-unexplored-and-unprotected/>
- The MathWorks, Inc. (2022). *Documentation: Stepinfo*. Retrieved from MathWorks Help Center: [https://www.mathworks.com/help/control/ref/lti.stepinfo.html#mw\\_9d8cc8c2-e27d-4cc9-a4fa-aaebfffa02c](https://www.mathworks.com/help/control/ref/lti.stepinfo.html#mw_9d8cc8c2-e27d-4cc9-a4fa-aaebfffa02c)
- Ullah, B., Ovinis, M., Baharom, M. B., Ali, S. S., Khan, B., & Javaid, M. Y. (2019). Effect of Waves and Current on Motion Control of Underwater Gliders. *Journal of Marine Science and Technology*.
- Usta, M. A., Akyazi, Ö., & Altaş, İ. H. (2011). Design and Performance of Solar Tracking System with Fuzzy Logic Controller Used Different Membership Functions. *2011 7th International Conference on Electrical and Electronics Engineering (ELECO)* (pp. II-381-II-385). Bursa, Turkey: IEEE.

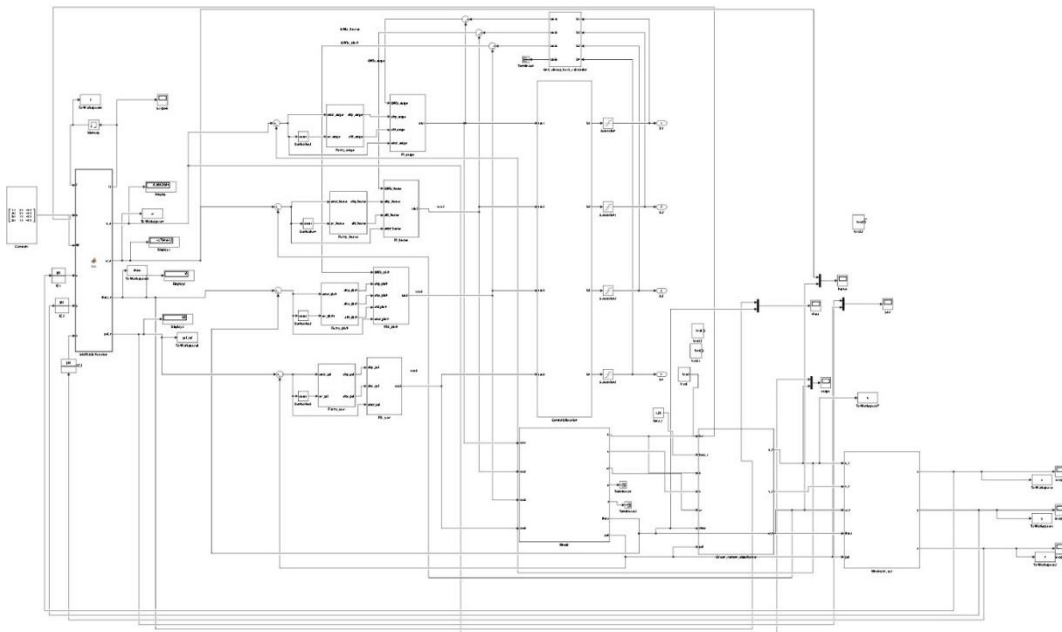
- Villa, J., Vallicrosa, G., Aaltonen, J., Ridao, P., & Koskinen, K. T. (2021). Model-Validation and Implementation of a Path-Following Algorithm in an Autonomous Underwater Vehicle. *Applied Sciences MDPI*.
- Vu, M. T., Le, T.-H., Thanh, H., Huynh, T.-T., Van, M., Hoang, Q.-D., & Do, T. (2021). Robust Position Control of an Over-actuated Underwater Vehicle under Model Uncertainties and Ocean Current Effects Using Dynamic Sliding Mode Surface and Optimal Allocation Control. *Sensors*, 747.
- WangReady. (2012, June 25). *Kendali PID*. Retrieved from WangReady: Robotics, Electronics, & Hardware Programming: <https://wangready.wordpress.com/2012/06/25/kendali-pid/>
- Xiang, X., Yu, C., & Zhang, Q. (2017). Robust Fuzzy 3D Path Following for Autonomous Underwater Vehicle Subject to Uncertainties. *Computers and Operations Research*.
- Xiang, X., Yu, C., Lapierre, L., & Zhang, J. (2017). Survey on Fuzzy-Logic-Based Guidance and Control of Marine Surface Vehicles and Underwater Vehicles. *International Journal of Fuzzy Systems*.
- Xu, H., Zhang, G.-c., Cao, J., Pang, S., & Sun, Y.-s. (2019). Tracking Control Based on Command Filter and Disturbance Observer. *Sensors MDPI*.
- Xu, H., Zhang, G.-c., Cao, J., Pang, S., & Sun, Y.-s. (2019). Underactuated AUV Nonlinear Finite-Time Tracking Control Based on Command Filter and Disturbance Observer. *Sensors*. doi:10.3390/s19224987
- Yang, R. (2016). *Modeling and Robust Control Approach for Autonomous Underwater Vehicle*. Qingdao: Université de Bretagne occidentale.
- Yazdani, A. M., Sammut, K., Lammas, A., & Tang, Y. (2016). *Real-Time Quasi-Optimal Trajectory Planning for Autonomous Underwater Docking*. Retrieved Juni 4, 2022, from [https://www.researchgate.net/publication/301819394\\_Real-time\\_Quasi-Optimal\\_Trajectory\\_Planning\\_for\\_Autonomous\\_Underwater\\_Docking](https://www.researchgate.net/publication/301819394_Real-time_Quasi-Optimal_Trajectory_Planning_for_Autonomous_Underwater_Docking)

## LAMPIRAN

### A. Representasi Simulink



**Gambar A. 1** Rangkaian Simulink untuk Sistem Kendali Berbasis PID



**Gambar A. 2** Rangkaian Simulink untuk Sistem Kendali Berbasis *Fuzzy*-PID

## B. Kode MATLAB untuk *Waypoint Following Guidance*

```

function [i1, u_d, w_d, theta_r, psi_r] = fcn(i,u,W,x,y,z)
%#codegen

%Calculate max i
K=size(W,1);

%COA based on simulation parameters, Mendes (2017:66)
rho_k = 1.5; % in meter
rho_z = 0.5; % in meter
rho_u = 1; % in m/s

%Limit based on simulation parameters, Mendes (2017:66)
k_u = 0.8; % in m/s
k_s = 0.8; % ngasal, dari Mendes (2017:38)
k_w = 0.4; % in m/s

%Calculate dk ez
%Identify next waypoint
next_waypoint = W(i,:);
x_k = next_waypoint(1);
y_k = next_waypoint(2);
z_k = next_waypoint(3);
%Calculate distance between vehicle position to the next waypoint
dk=sqrt(((x_k-x)^2)+((y_k-y)^2));
%Calculate depth error
e_z = z_k-z;

if i+1==K(1),
    i1=i;
else
if dk<=rho_k && abs(z-z_k)<rho_z,
    i=i+1;
    i1=i;
else

```

```

        il=i;
    end
end

% Calculate common mode vs differential mode weight
ucms = 0.5; % in m/s
udms = 0.5; % in m/s
ucmi = 0.3; % in m/s
udmi = 0.3; % in m/s
sigma_cm = 0.05;
sigma_dm = 0.05;

Wcm = 1-0.5*(tanh((u-((ucms+ucmi)/2)/sigma_cm))+1);
Wdm = 0.5*(tanh((u-((udms+udmi)/2)/sigma_dm))+1);

% calculate u_d
d_u = dk-rho_u;
u_d = k_u*asin((d_u/(abs(d_u)+k_s))*2/pi);

% calculate w_d
w_d = Wcm*k_w*asin((e_z/(abs(e_z)+k_s))*2/pi);

% calculate theta_r
theta_r = rad2deg(Wdm*atan2(z_k-z,x_k-x));
if theta_r>=25,
    theta_r=25;
else
    if theta_r<-25,
        theta_r=-25;
    end
end

% calculate psi_r
psi_r = atand((y_k-y)/(x_k-x));
end

```

### C. Kode MATLAB untuk Membuat Plot X-Y-Z

```

%%Circle 1 XY-plane
% Define circle parameters:
radius1 = 1.5;
xCenter1 = 0;
yCenter1 = 0;
zCenter1 = 0;
% Make an array for all the angles:
theta1 = linspace(0, 2 * pi, 2000);
% Create the x and y locations at each angle:
x1 = radius1 * cos(theta1) + xCenter1;
y1 = radius1 * sin(theta1) + yCenter1;
% Need to make a z value for every (x,y) pair:
z1 = zeros(1, numel(x1)) + zCenter1;

%%Circle 2 XY-plane
% Define circle parameters:
radius2 = 1.5;
xCenter2 = 10;
yCenter2 = 20;
zCenter2 = -20;
% Make an array for all the angles:
theta2 = linspace(0, 2 * pi, 2000);
% Create the x and y locations at each angle:
x2 = radius2 * cos(theta2) + xCenter2;
y2 = radius2 * sin(theta2) + yCenter2;
% Need to make a z value for every (x,y) pair:
z2 = zeros(1, numel(x2)) + zCenter2;

%%Circle 3 XY-plane
% Define circle parameters:
radius3 = 1.5;
xCenter3 = 30;
yCenter3 = 20;
zCenter3 = -20;

```



```

% Make an array for all the angles:
theta3 = linspace(0, 2 * pi, 2000);
% Create the x and y locations at each angle:
x3 = radius3 * cos(theta3) + xCenter3;
y3 = radius3 * sin(theta3) + yCenter3;
% Need to make a z value for every (x,y) pair:
z3 = zeros(1, numel(x3)) + zCenter3;

%%Circle 4 XY-plane
% Define circle parameters:
radius4 = 1.5;
xCenter4 = 50;
yCenter4 = 40;
zCenter4 = -20;
% Make an array for all the angles:
theta4 = linspace(0, 2 * pi, 2000);
% Create the x and y locations at each angle:
x4 = radius4 * cos(theta4) + xCenter4;
y4 = radius4 * sin(theta4) + yCenter4;
% Need to make a z value for every (x,y) pair:
z4 = zeros(1, numel(x4)) + zCenter4;

figure(1)
plot3(x.Data,y.Data,z.Data,'LineWidth',0.8);
color_line3(x.Data,y.Data,z.Data,u.Data);
hold on;

%%Circle 1 XY-plane
% Do the plot:
% First plot the center:
plot3(xCenter1, yCenter1, zCenter1, 'r*', 'LineWidth', 2,
'MarkerSize', 5);
hold on; % Don't let circle blow away our center.
% Next plot the circle:
plot3(x1, y1, z1, 'b-', 'LineWidth', 1);

%%Circle 2 XY-plane

```

```

% Do the plot:
% First plot the center:
plot3(xCenter2, yCenter2, zCenter2, 'r*', 'LineWidth', 2,
'MarkerSize', 5);
hold on; % Don't let circle blow away our center.
% Next plot the circle:
plot3(x2, y2, z2, 'b-', 'LineWidth', 1);

%%Circle 3 XY-plane
% Do the plot:
% First plot the center:
plot3(xCenter3, yCenter3, zCenter3, 'r*', 'LineWidth', 2,
'MarkerSize', 5);
hold on; % Don't let circle blow away our center.
% Next plot the circle:
plot3(x3, y3, z3, 'b-', 'LineWidth', 1);

%%Circle 4 XY-plane
% Do the plot:
% First plot the center:
plot3(xCenter4, yCenter4, zCenter4, 'r*', 'LineWidth', 2,
'MarkerSize', 5);
hold on; % Don't let circle blow away our center.
% Next plot the circle:
plot3(x4, y4, z4, 'b-', 'LineWidth', 1);

grid on;
axis('square');
xlabel('x (meter)', 'FontSize', 10);
ylabel('y (meter)', 'FontSize', 10);
zlabel('z (meter)', 'FontSize', 10);

```

#### **D. Kode MATLAB untuk Menghitung Performansi Uji *Closed-Loop***

```

stepinfo (heaveFUZZY.Data,heaveFUZZY.Time)
sserror=(0.4- (heaveFUZZY.Data (end,end)))

```

## BIODATA PENULIS



Carollina Kusumawidjaya lahir di Semarang pada tanggal 23 Desember, kini mahasiswa tingkat akhir di Departemen Teknik Fisika, Institut Teknologi Sepuluh Nopember, Surabaya dan merupakan lulusan dari SMA Kolese Loyola Semarang.

Semasa kuliah, penulis aktif di berbagai organisasi di kampus seperti *Society of Petroleum Engineers ITS Student Chapter*, Keluarga Mahasiswa Katolik ITS, dan Himpunan Mahasiswa Teknik Fisika ITS. Selain itu, penulis juga aktif mengikuti kompetisi dan memiliki beberapa pencapaian, di antaranya: menjadi salah satu penerima Djarum Beasiswa Plus 2020/2021; menjadi delegasi mahasiswa Indonesia dalam *International Petroleum Technology Conference 2022* di Riyadh, Saudi Arabia yang disponsori oleh Saudi Aramco; dan terpilih sebagai salah satu peserta *Young Leaders for Indonesia Wave 14* yang diadakan oleh *McKinsey & Company* pada 2022.