TUGAS AKHIR – TE 141599

DESAIN DAN IMPLEMENTASI KONTROLER *SELF-TUNING* PID DENGAN PENDEKATAN INTERAKSI ADAPTIF PADA SISTEM PENGATURAN LEVEL

Muhammad Zakki Ghufron
NRP 2212100064

Dosen Pembimbing
Ir. Ali Fatoni, MT.
Imam Arifin, ST., MT.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

FINAL PROJECT – TE 141599

DESIGN AND IMPLEMENTATION OF SELF-TUNING PID CONTROLLER BY ADAPTIVE INTERACTION APPROACH IN LEVEL CONTROL SYSTEM

Muhammad Zakki Ghufron
NRP 2212100064

Supervisor
Ir. Ali Fatoni, MT.
Imam Arifin, ST., MT.

DEPARTMENT OF ELECTRICAL ENGINEERING
Faculty of Industrial Technology
Sepuluh Nopember Institute of Technology
Surabaya 2016

DESAIN DAN IMPLEMENTASI KONTROLER *SELF TUNING* PID DENGAN PENDEKATAN INTERAKSI ADAPTIF PADA SISTEM PENGATURAN LEVEL

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Teknik Sistem Pengaturan
Jurusan Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui :

Dosen Pembimbing I	Dosen Pembimbing II
Ir. Ali Fatoni, MT.	**Imam Arifin, S.T., M.T.**
NIP. 196206031989031002	NIP. 197302222002121001

SURABAYA
JANAURI, 2016

DESAIN DAN IMPLEMENTASI KONTROLER *SELF-TUNING* PID DENGAN PENDEKATAN INTERAKSI ADAPTIF PADA SISTEM PENGATURAN LEVEL

Nama : Muhammad Zakki Ghufron
Pembimbing : Ir. Ali Fatoni, MT.
Imam Arifin, ST., MT.

ABSTRAK

Kontroler PID sampai saat ini masih mampu menghasilkan performa sistem sesuai dengan keinginan ketika di-*tuning* dengan benar. Kata "di-*tuning* dengan benar" menjadi sorotan utama dalam perkembangan kontroler PID, karena performansinya sangat bergantung pada proses *tuning* yang dilakukan. Penerapan kontroler PID pada sistem pengaturan level masih dinilai kurang maksimal ketika terjadi proses pembebanan. Kontroler PID harus di-*tuning* ulang agar dapat menghasilkan performa yang sesuai keinginan. Pada penelitian ini akan diterapkan metode *self-tuning* PID dengan menggunakan pendekatan interaksi adaptif. Metode interaksi adaptif mampu mengadaptasi perubahan yang terjadi pada *plant* seperti pembebanan, sehingga ketergantungan kontroler PID terhadap proses tuning ulang bisa dihilangkan. Metode *self-tuning* PID dengan interaksi adaptif dirancang dengan tiga buah kriteria *kesalahan* yaitu menggunakan *Square Kesalahan* (SE), *Time Multiplied Square Kesalahan* (TSE), *Square Time Multiplied Square Kesalahan* (STSE). Ketika Simulasi kontroler *self tuning* PID kriteria STSE lebih unggul dalam hal perubahan *set point* dengan RMSE sebesar 3,64 % dan dalam hal pembebanan dengan RMSE dan *recovery time* sebesar 0,093 % dan 2,2 detik. Ketika implementasi kontroler PID lebih unggul dalam hal perubahan *set point* dengan RMSE sebesar 2,34% dan kontroler *self tuning* PID kriteria TSE lebih unggul dalam hal pembebanan dengan RMSE dan *recovery time* sebesar 13,74 % dan 29 detik.

Kata Kunci: *Self-Tuning PID,* Interaksi Adaptif, Sistem Pengaturan level

DESIGN AND IMPLEMENTATION OF SELF-TUNING PID CONTROLLER BY ADAPTIVE INTERACTION APPROACH IN LEVEL CONTROL SYSTEM

Name : Muhammad Zakki Ghufron
Supervisor : Ir. Ali Fatoni, MT.
Imam Arifin, ST., MT.

ABSTRACT

Furthermore, a well-tuned PID controller still able to give excellent system performance. Word "well-tuned" become main focus in its development because performance of PID controller is dependent on tuning process. Implementation in level control system still considered that PID controler can't give excelent performance when loading effect occur and plant parameter changes. PID controller tuning must be repeated in order to get an excellent performance as desired. Therefore, in this final project report will be implemented a method of self-tuning PID by adaptive interaction approach.. Adaptive interaction is able to adapt in plant parameter changes such as loading effect so repeated PID controller tuning can be vanish. Self-tuning PID with adaptive interaction is designed with three criteria of kesalahan using Square Kesalahan (SE), Time Multiplied Square Kesalahan (TSE), and Square Time Multiplied Square Kesalahan (STSE). In Simulation self tuning PID controller by STSE criteria give better performance in set point changes with RMSE 3,64 % and load disturbance with RMSE dan recovery time about 0,093 % and 2,2 second. In implementation PID Controller give better performance in set point changes with RMSE about 2,34% and self tuning PID by TSE criteria better in load disturbance with RMSE and recovery time about 13,74 % and 29 second.

***Keywords**: Self-Tuning PID, Adaptive Interaction, Level Control System*

KATA PENGANTAR

Segala Puji bagi Allah SWT atas segala rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan laporan penelitian ini dengan segala kekurangannya. Sholawat serta salam semoga senantiasa tercurahkan terhadap Nabi besar Muhammad SAW, yang telah membawa umat manusia dari jaman yang gelap menuju jaman yang terang benderang.

Laporan penelitian ini berjudul **"Desain dan Implementasi Kontroler *Self-Tuning* PID dengan Pendekatan Interaksi Adaptif Pada Sistem Pengaturan Level"** ini dipersembahkan untuk kemajuan riset dalam dunia teknologi dan guna memenuhi persyaratan untuk mencapai Gelar Sarjana Teknik pada Bidang Studi Teknik Sistem Pengaturan, Jurusan Teknik Elektro Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember (ITS) Surabaya.

Pada kesempatan yang berbahagia ini penulis mengucapkan terima kasih kepada semua pihak yang telah banyak membantu baik secara langsung maupun tidak langsung, hingga penelitian ini dapat diselesaikan. Terima kasih tak ternilai kepada keluarga penulis khususnya kedua orang tua yang selalu memberikan motivasi dan dukungan dalam penelitian ini. Selain itu, Penulis secara khusus mengucapkan terima kasih yang sebesar-besarnya kepada Bapak Ir. Ali Fatoni, MT. dan Bapak Imam Arifin, ST., MT. atas bimbingan selama mengerjakan tugas akhi ini. Ucapan terima kasih juga penulis ucapkan kepada Bapak Moh. Abdul Hady, ST., MT. atas bimbingan, saran, dan nasihat dalam penelitian ini. Teman-teman asisten laboratorium teknik sistem pengaturan dasar AJ-104, dan juga kepada teman-teman angakatan E-52. Serta berbagai pihak yang turut membantu dalam pengerjaan penelitian ini yang penulis tidak dapat menyebutkannya satu persatu.

Laporan penelitian ini masih jauh dari kata sempurna baik dalam penyusunan, maupun dalam pembahasan dan analisa permasalahan. Untuk itu pada kesempatan ini pula penulis mohon maaf atas segala ketidaksempurnaan yang ada. Semoga laporan penelitian ini dapat memberikan manfaat bagi pembaca sebagai acuan penelitian selanjutnya.

Surabaya, Januari 2016

Penulis

DAFTAR ISI

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Level merupakan salah satu variabel kontrol yang hampir ada di setiap industri. Sistem pengaturan level mempunyai karakteristik *time-varying* dan nonlinear pada industri skala besar [1]. Berbagai metode kontrol sudah diterapkan dalam sistem pengaturan level seperti kontroler PID dan *fuzzy*-PID keduanya belum mampu memberikan hasil yang presisi [1]. Kombinasi PID *neural network* membutuhkan waktu yang lama untuk proses *learning* [2]. Kombinasi PID dengan metode adaptif memberikan hasil yang baik ketika digunakan dalam sistem dengan *plant* dinamik[1][3]. Metode adaptif yang sudah sering diterapkan seperti *Model Reference Adaptive Controller* (MRAC) masih menggunakan sistem yang kurang simpel seperti adanya identifikasi sistem sebelum melakukan proses adaptasi [4].

Sampai saat ini, kontroler PID masih mampu menghasilkan performansi sistem sesuai dengan keinginan ketika di-*tuning* dengan benar. Kata di-*tuning* dengan benar menjadi sorotan utama dalam perkembangannya karena performansi PID sangat bergantung pada proses *tuning* yang dilakukan. Terdapat dua metode *tuning* yaitu *offline tuning* dan *online tuning*. *Offline tuning* dilakukan secara manual oleh campur tangan manusia, sedangkan *tuning online* dilakukan secara otomatis sesuai dengan metode yang digunakan. Pada penelitian ini diterapkan metode *online tuning* pada kontroler PID. Menurut Feng Lin dan Robert D. Brandt [5], terdapat dua alasan mengapa menggunakan *online tuning*. Pertama, tujuan dan kebutuhan dari kontroler PID sering berubah selama penerapan dalam kondisi yang berbeda. Sering kali performansi sistem yang diinginkan adalah respon yang cepat, tetapi juga dapat mengurangi kesalahan keadaan tunak. Pada kenyataanya respon yang cepat dan mengurangi kesalahan keadaan tunak membutuhkan parameter yang berbeda jika menggunakan kontroler PID. Jika tetap diinginkan respon dengan spesifikasi cepat dan tanpa kesalahan, maka pada kondisi awal disesuaikan untuk mendapatkan respon yang cepat kemudian dilakukan *tuning* ulang untuk mengurangi kesalahan keadaan tunak. Alasan kedua adalah dikarenakan dalam kenyataanya di dunia industri *plant* yang dikendalikan mengalami perubahan parameter pada kondisi-kondisi tertentu. Untuk mengatasi perubahan parameter *plant*

diperlukan metode yang dapat beradaptasi langsung dengan perubahan tersebut. Pada penelitian yang telah dilakukan sebelumnya banyak digunakan metode kontrol adaptif dengan berbagai metode untuk mengatasi perubahan *plant* tersebut. Metode yang digunakan baik melaui pendekatan model, identifikasi parameter estimasi, dan pendekatan kecerdasan buatan seperti *fuzzy* dan jaringan saraf tiruan. Dalam penelitian ini ditawarkan metode *tuning* parameter PID dengan pendekatan interaksi adaptif yang diusulkan oleh Feng Lin dan Robert D. Brandt [5]. Keistimewaan metode interaksi adaptif adalah penerapannya yang tidak membutuhkan banyak informasi seperti parameter estimasi dan model acuan dari *plant*. Hanya dibutuhkan nilai *kesalahan* dari sistem untuk melakukan adaptasi. Kontroler dengan *tuning* interaksi adaptif akan divariasi pada bagian indeks performansinya untuk menunjukkan perbandingan penentuan spesifikasi performansi dari sistem yang akan diadaptasi. Indeks performansi tersebut adalah *Square Kesalahan* (SE), *Time Multiplied Squared Kesalahan* (TSE), dan *Squared Time Multiplied Squared Kesalahan* (STSE).

1.2 Perumusan Masalah

Kontroler PID pada dasarnya bersifat tetap selama parameter P, I, dan D tidak diubah. Ketika terjadi perubahan parameter *plant* yang diakibatkan oleh pembebanan, terkadang sinyal kontrol yang diberikan tidak sesuai dengan kondisi *plant* saat ini sehingga muncul pelonjakan atau penurunan nilai. Waktu tunak yang dibutuhkan sistem dengan kontroler PID masih terlalu lama. Selain itu, besarnya penurunan atau lonjakan nilai yang terjadi masih terlalu besar, sehingga dapat disimpulkan kontroler PID belum mampu mengatasi pembebanan jika tidak dilakukan *tuning* ulang pada parameter P, I, dan D.

1.3 Batasan Masalah

Pada penelitian ini hanya akan membahas analisis mengenai nilai RMSE pada setiap pengujian yang dilakukan. Pada pengujian dengan pembebanan akan ditambahkan analisis mengenai *recovery time* dari sistem.

1.4 Tujuan

Penelitian ini diharapkan mampu menerapkan sebuah metode kontrol untuk mengurangi besarnya penurunan nilai dan mempercepat *recovery time* ketika terjadi pembebanan pada sistem.

1.5 Sistematika Penyusunan Laporan

Penyusunan laporan penelitian ini akan dibagi menjadi lima Bab dengan sistematika sebagai berikut:

BAB 1 : PENDAHULUAN

Penjelasan dalam bab ini meliputi latar belakang, perumusan masalah, batasan masalah, tujuan penelitian, sistematika laporanm, dan juga manfaat.

BAB 2 : DASAR TEORI

Teori dasar sebagai pendukung dalam penelitian ini diantaranya *plant* PCT-100, kontroler PID, kriteria *kesalahan*, interaksi adaptif. Selain itu terdapat teori mengenai *software* yang digunakan untuk simulasi dan juga menghubungkannya terhadap *plant* dibahas pada Bab ini.

BAB 3 : PERANCANGAN SISTEM

Perancangan sistem terdiri dari pemodelan *plant* menggunakan hukum fisika, desain dan perancangan kontroler PID dan *self-tuning* dengan interaksi adaptif berdasarkan teori pada BAB II. Selain itu dibahas juga mengenai *debitchart* algoritma kontrol yang dibuat.

BAB 4 : HASIL PENGUJIAN DAN ANALISA

Data hasil simulasi, implementasi, dan analisis mengenai hasil yang didapatkan dijelaskan pada Bab ini

BAB 5 : PENUTUP

Kesimpulan dari hasil pengujian mengenai penelitian ini dibahas pada Bab ini. Selain itu juga disertakan saran mengenai kekurangan dari penelitian ini dan rencana riset selanjutnya yang bisa dikembangkan.

1.6 Manfaat

Penelitian ini akan memberikan inovasi terbaru dalam hal metode kontrol untuk mengatasi pembebanan. Metode *self-tuning* PID interaksi adaptif pada penelitian ini dapat diaplikasikan pada dunia industri dengan kondisi beban yang sulit diprediksi dan berubah-ubah setiap waktunya.. Selain itu, penelitian ini juga dapat digunakan sebagai acuan untuk pengembangan kontroler adaptif yang dikombinasikan dengan kontroler lainnya.

(Halaman ini sengaja dikosongkan)

BAB 2
SISTEM PENGATURAN LEVEL PCT-100

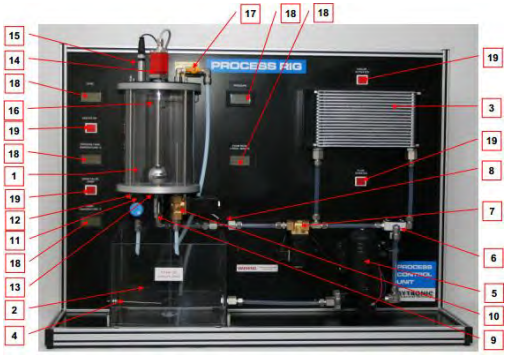
2.1 PCT-100 [6]

PCT-100 adalah miniatur sistem pengaturan proses pada dunia industri seperti pada industri makanan dan minuman, industri kimia, pemurnian air, dan pengolahan limbah. PCT-100 merupakan produk dari vendor *Bytronic*. Perangkat lunak yang mendukung PCT-100 mempunyai fasilitas untuk melakukan pengaturan terhadap empat buah variabel kontrol diantaranya debit, level, suhu, dan tekanan. Tujuan dari PCT-100 adalah untuk memberikan pembelajaran yang mencerminkan masalah pengaturan sebuah sisem sebenarnya dalam dunia industri, mempermudah untuk melakukan analisis dan menerapkan metode kontrol yang lain. PCT-100 juga dapat digunakan untuk menggambarkan secara sederhana dan jelas, karakteristik dari kontroler P, I, dan D.

Beberapa sasaran PCT-100 antara lain memberikan keuntungan yang dalam hal penerapan pengaturan berbasiskan mikroprosesor untuk proses otomasi, menyediakan proses skala kecil dengan gambaran masalah yang ditemukan dalam dunia industri sehingga dapat diterapkan teknik kontrol yang berbeda, menunjukkan kesederhanaan dan efektifitas dari kontroler yang banyak digunakan yaitu PID, dapat melakukan pengamatan terhadap sinyal digital dan analog untuk mengembangkan pemahaman tentang metode yang sedang digunakan, dan menyediakan fasilitas cocok untuk para operator dalam praktek menemukan dan menyelesaikan permasalahan. Komponen utama pada PCT-100 terdiri dari process rig dan control module. Process rig dan control module dihubungkan oleh *port* serial untuk melakukan proses pertukaran data. Penjelasan lebih detail mengenai masing-masing komponen disajikan pada subbab selanjutnya.

2.1.1 Process Rig [6]

Process rig merupakan tampilan miniatur proses pada dunia industri sepeti pada Gambar 2.1. Process rig terdiri dari 19 komponen penyusun utama yang dilambangkan oleh angka 1 sampai 19 pada Gambar 2.1. Process rig mempunyai empat variabel yang dapat dikontrol diantaranya debit, level, suhu, dan tekanan. Keempat variabel tersebut dapat dikontrol melalui komputer atau *Programmable Logic Controller* (PLC) yang terhubung dengan control module.



Gambar 2.1 *Process rig* PCT-100[6]

Penjelasan mengenai angka yang terdapat pada Gambar 2.1 ditunjukkan pada Tabel 2.1.

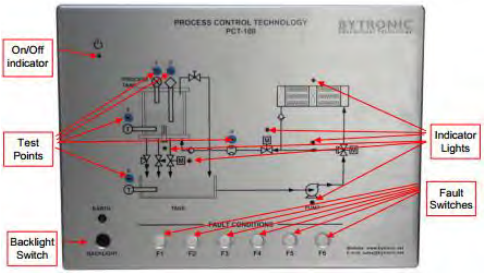
Tabel 2.1 Komponen penyusun process rig

No	Keterangan
1	Tangki proses
2	Tangki penampungan air
3	Pendingin
4	Sensor suhu tangki penampungan
5	Pompa
6	3/2 *diverter valve*
7	2/2 *proportional control valve*
8	Sensor *debit*
9	*One way check valve*
10	2/2 *proportional drain valve*
11	Needle valve
12	*Pressure relief valve*
13	Pemanas
14	Sensor level
15	*Transducer* tekanan
16	*Float switch*
17	*Overdebit valve*
18	*Digital LCD displays*
19	Lampu indikator

Tangki penampungan berisi air yang akan dipompa menuju tangki proses. Dalam tangki penampungan terdapat sensor suhu berupa *Platinum Resistance Thermometer* (PRT) yang nilainya ditampilkan melalui *digital display*. Pompa pada process rig bertugas melakukan sirkulasi air ke semua sistem. Pompa tersebut diatur oleh motor DC dengan kecepatan yang mampu divariasikan untuk menghasilkan debit air yang berbeda-beda. Debit air yang melewati pipa akan diukur oleh sensor debit yang nilainya ditampilkan melalui *digital display*. Proses utama terjadi pada tangki proses yang terdiri dari sensor level untuk mengukur level air dalam tangki, elemen pemanas, PRT, *drain valve* yang digunakan untuk mengeluarkan air dari tangki proses, *needle valve* yang digunakan untuk menambahkan gangguan pada proses, *transducer* tekanan, *float switch* dan *one way check valve.*

2.1.2 Control Module [6]

Control module mengkoordinasikan semua rangkaian elektronik dan juga sebagai penghubung antara process rig dan kontroler. Control module berisi indikator aktif dari keseluruhan sistem dan titik pengujian.



Gambar 2.2 Control module (tampak depan)[6]

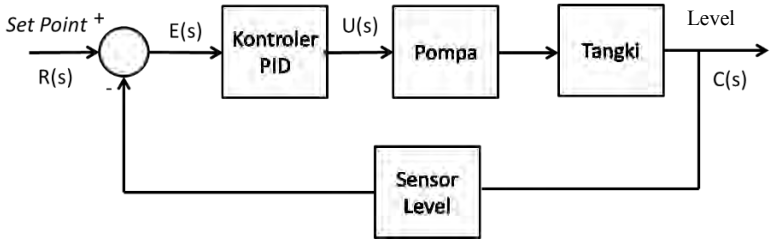
Proses pengiriman data baik berupa analog dan digital dilakukan melalui control module. Selain itu, control module juga memiliki pengaman dan sistem untuk memberikan masalah atau disebut *switched faults*. *Switched faults* berfungsi memberikan masalah nyata seperti terdapat pada dunia industri sehingga kita dapat melatih kemampuan kita dalam menemukan dan memecahkan permasalahan yang ada. Masalah yang ada meliputi hal-hal pada pada Tabel 2.2. *Switched faults* berisi tombol pada bagian depan control module ditunjukkan pada Gambar 2.2.

Tabel 2.2 Daftar *switched fault*

Fault No	Fungsi	Efek
F1	AD *Fault*	ADC terkunci
F2	Pompa	Pompa tidak bisa diaktifkan
F3	Suhu	*Display* proses suhu permanen membaca skala penuh
F4	Pendingin	Pendingin permanen aktif
F5	Pemanas	*Display* pemanas memberikan pembacaan yang salah
F6	Sensor *debit*	Tidak ada umpan balik *debit*

2.1.3 Sistem Pengaturan Level PCT-100

Sistem pengaturan level memiliki tujuan untuk mempertahankan nilai level fluida pada referensi tertentu meskipun terjadi gangguan atau bisa disebut *regulator*. Diagram blok sistem pengaturan level pada PCT-100 ditunjukkan oleh Gambar 2.3. Komponen penyusun sistem ini adalah kontroler PID, pompa sebagai aktuator, tangki proses sebagai *plant,* kemudian sensor level sebagai *measurement system.*

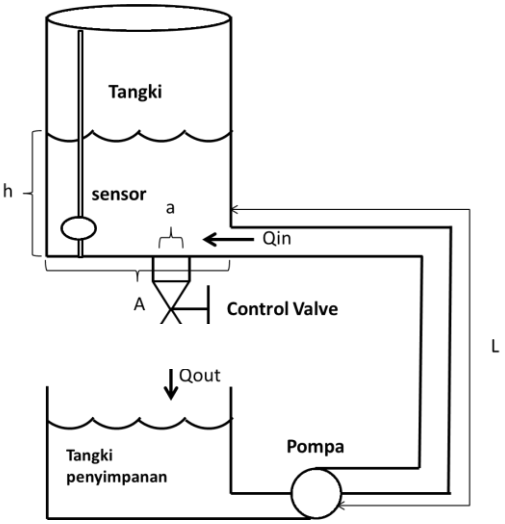


Gambar 2.3 Diagram blok sistem pengaturan level

Mekanisme kerja sistem adalah ketika air pada tanki penampungan dipompa menuju tangki proses, pada tangki proses level air akan diukur dan kemudian dibandingkan dengan referensi yang dipilih. Jika nilai yang terukur masih belum sesuai dengan referensi maka kontroler bertugas memberikan sinyal koreksi atau sinyal kontrol kepada aktuator sehingga mampu memberikan aksi yang tepat. Keluaran dari sistem berupa level dari air dalam cm yang akan dikonversikan menjadi tegangan dengan skala 0 – 10 volt oleh sensor level.

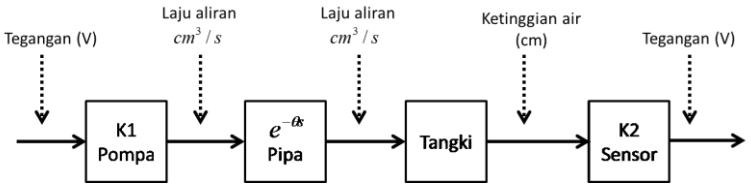
2.2 Pemodelan *Plant* [7]

Plant yang digunakan pada Gambar 2.4 merepresentasikan dua proses utama dalam sistem pengaturan level yaitu proses pengisian dan pembuangan air. Proses pembuangan air dalam tangki dilakukan oleh *control valve* sedangkan proses pengisian air dalam tangki dilakukan oleh pompa.



Gambar 2.4 Gambaran *plant*

Proses pemodelan *plant* pada sistem pengaturan level digambarkan seperti diagram alir pada Gambar 2.5. Proses pemodelan dimulai dari pemodelan pompa, pipa, tangki, dan sensor.



Gambar 2.5 Diagram blok pemodelan sistem pengaturan level

Semua parameter pemodelan *plant* pada Tabel 2.3 beserta keterangan dan satuan yang digunakan nilainya diambil dari *data sheet* dan pengukuran langsung pada *plant* PCT-100.

Tabel 2.3 Parameter pemodelan *plant*

Simbol	Keterangan	Satuan
Q_{in}	Debit air yang masuk ke tangki	cm^3/s
Q_{out}	Debit air yang keluar dari tangki	cm^3/s
Q	Debit air dalam tangki	cm^3/s
H	Ketinggian air dalam tangki	cm
A	Luas alas tanki	cm^2
a	Diameter pipa	cm
h_0	Ketinggian air saat kondisi di titik operasi	cm
Q_0	Laju aliran keluar saat kondisi di titik operasi	cm^3/s
R	Resistansi *control valve*	s/cm^2
C	Kapasitansi tangki	cm^2
L	Panjang pipa	cm
θ	Waktu tunda	detik

2.2.1 Pemodelan Pompa

Pemodelan pompa bertujuan untuk mendapatkan nilai penguatan dari pompa yang dilambangkan dengan K_1. Mencari nilai K_1 dirumuskan seperti Persamaan 2.1

$$K_1 = \frac{\text{Rentang debit air yang keluar dari pompa } (cm^3/\text{s})}{\text{Rentang tegangan masukan } (V)} \qquad (2.1)$$

2.2.2 Pemodelan Pipa

Pemodelan pipa dilakukan untuk mencari tahu besarnya waktu penundaan yang diakibatkan oleh panjang pipa. Hubungan waktu tunda dengan panjang pipa dapat dirumuskan pada Persamaan 2.2.

$$\theta = \frac{\text{Volume pipa}(cm^3)}{\text{Debit air yang masuk ke pipa}(cm^3/\text{s})} \quad (2.2)$$

Jika dihubungkan dengan sistem pengaturan level maka waktu tunda yang terjadi merupakan hubungan antara debit air yang dikeluarkan pompa dan debit air yang masuk ke tangki dituliskan pada Persamaan 2.3.

$$\frac{Q_{inT}}{Q_{outP}} = e^{-\theta s} \quad (2.3)$$

2.2.3 Pemodelan Tangki

Komponen utama dalam pemodelan tangki adalah tangki itu sendiri dan *control valve*. Tangki diasumsikan seperti kapasitor dikarenakan sifatnya yang mirip. Ketika proses pengisian air seperti proses pengisian muatan sedangkan proses pembuangan air seperti proses pembuangan muatan pada kapasitor. Sehingga Kapasitansi tangki dapat didekati dengan luas alas tangki yang berupa lingkaran seperti Persamaan 2.4.

$$C = \frac{\text{Perubahan volume air } (cm^3)}{\text{Peruahan ketinggian air } (m)} \quad (2.4)$$

$$C = \text{Luas penampang tangki} = \frac{\pi d^2}{4} = A$$

Pemodelan tangki dimulai dari hukum kesetimbangan massa pada Persamaan 2.5 dimana debit air didalam merupakan selisih antara debit air yang masuk ke dalam tangki dan debit air yang keluar dari tangki.

$$Q_{in} - Q_{out} = Q \quad (2.5)$$

Hubungan antara debit dengan level air dalam tangki dirumuskan pada Persamaan 2.6.

$$Q = Av = A\frac{dh}{dt} \quad (2.6)$$

Substitusikan Persamaan 2.4 dan 2.6 pada Persamaan 2.5 sehingga hubungan antara debit air yang masuk dan yang keluar dituliskan pada Persamaan 2.7.

$$Q_{in} - Q_{out} = C\frac{dh}{dt} \tag{2.7}$$

Control valve diasumsikan seperti resistansi atau hambatan pada saat proses pembuangan air. Hal tersebut dikarenakan debit air yang keluar bergantung pada bukaan dari *control valve*, sehingga dapat dirumuskan pada Persamaan 2.8. Perumusan ini dilakukan pada saat kondisi titik kerja sehingga $h = h_0$ dan $Q_{out} = Q_0$.

$$R = \frac{\text{Perubahan ketinggian air } (cm)}{\text{Perubahan debit air yang keluar } (cm^3/\text{s})} \tag{2.8}$$

Debit air yang melewati *control valve* adalah laminar karena tidak terdapat perubahan diameter dari pipa antara pompa dan *control valve* ditunjukkan oleh Persamaan 2.9.

$$R = \frac{h_0}{Q_0} \tag{2.9}$$

$$Q_0 = \frac{h_0}{R} \tag{2.10}$$

Substitusi hubungan antara debit air yang keluar dengan bukaan *valve* pada Persamaan 2.10 ke Persamaan 2.7, sehingga rumus kesetimbangan massa menjadi Persamaan 2.11.

$$Q_{in} - \frac{h_0}{R} = C\frac{dh_0}{dt} \tag{2.11}$$

$$Q_{in}R - h_0 = RC\frac{dh_0}{dt} \tag{2.12}$$

Dengan menggunakan transformasi *Laplace* pada Persamaan 2.12 maka diperoleh fungsi alih sistem pada Persamaan 2.15.

$$Q_{in}(s)R - H(s) = RCsH(s) \quad (2.13)$$

$$Q_{in}(s)R = H(s)(1 + RCs) \quad (2.14)$$

$$\frac{H(s)}{Q_{in}(s)} = \frac{R}{RCs+1} \quad (2.15)$$

2.2.4 Pemodelan Sensor

Pemodelan sensor bertujuan untuk mendapatkan nilai penguatan yang dilambangkan dengan K_2. Nilai K_2 dirumuskan dengan rentang kerja dari pompa terhadap masukan berupa tegangan seperti Persamaan 2.16.

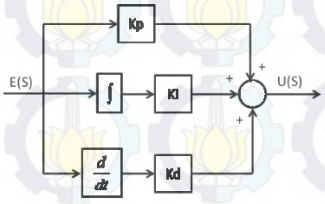
$$K2 = \frac{\text{Rentang tegangan keluaran } (V)}{\text{Rentang ketinggian air dalam tangki } (cm)} \quad (2.16)$$

Pemodelan keseluruhan plant dapat dihitung dengan rumus seperti Persamaan 2.17.

$$\frac{H(s)_{total}}{Q_{in}(s)_{total}} = K1 * e^{-\theta s} * \frac{H(s)}{Q_{in}(s)} * K2 \quad (2.17)$$

2.3 Kontroler PID [8]

Kontroler PID memiliki tiga komponen penyusun utama yaitu proporsional atau dilambangkan dengan P, Integral yang dilambangkan dengan I, dan differensial atau D. Pada Gambar 2.6 merupakan struktur kontroler PID. E(s) adalah *kesalahan* yang terjadi akibat selisih nilai aktual dan nilai yang diinginkan, sedangkan U(s) adalah sinyal kontrol yang dihasilkan oleh kontroler PID.



Gambar 2.6 Diagram blok kontroler PID

Kontroler PID menggabungkan keunggulan dari masing-masing kontroler penyusunnya untuk dijadikan sebagai sebuah kelebihan ketika disatukan. Kontroler PID merupakan kontroler dengan struktur paling mudah untuk diaplikasikan.

2.3.1 Kontroler Proporsional

Kontroler proporsional memberikan aksi yang proporsional terhadap kesalahan yang terjadi sekarang. Kontroler proporsional terdiri dari konstanta pengali atau penguatan yang ditunjukkan pada Persamaan 2.18. Kontroler proporsional berfungsi untuk mempercepat respon tetapi semakin besar nilainya maka semakin besar kesalahan yang terjadi dan juga akan menyebabkan sistem berosilasi.

$$u(t) = K_p e(t) = K_p (r(t) - y(t)), \tag{2.18}$$

Nilai K_p adalah penguatan proporsional. Jika kontroler ini diterapkan maka kesalahan yang sekarang akan terus meningkat sehingga dapat menimbulkan osilasi. Dengan melakukan transformasi *Laplace* pada Persamaan 2.18 maka fungsi alih dari kontroler proporsional dapat dituliskan menjadi Persamaan 2.19.

$$\frac{U(s)}{E(s)} = K_p \tag{2.19}$$

Pada aplikasinya di dunia industri penguatan proporsional biasanya digantikan dengan *Proportional Band* (PB) yang merupakan rentang kerja kesalahan yang dapat menyebabkan terjadinya perubahan rentang dari variabel kontrol pada Persamaan 2.20.

$$PB = \frac{100}{K_p} \tag{2.20}$$

2.3.2 Kontroler Integral

Kontroler integral memberikan aksi proporsional terhadap integral dari kesalahan. Kontroler ini dapat menghilangkan kesalahan yang terjadi namun mengakibatkan respon sistem menjadi lambat. Pada kontroler integral terdapat komponen τ_i yang merupakan waktu integral.

Hubungan antara kesalahan dan aksi integral dapat dilihat pada Persamaan 2.21.

$$u(t) = K_i \int_0^t e(\tau) d\tau \tag{2.21}$$

Nilai K_i adalah penguatan integral. Aksi kontrol ini muncul berdasarkan nilai kesalahan waktu sebelumnya pada Persamaan 2.22.

$$\frac{U(s)}{E(s)} = \frac{K_i}{s} \tag{2.22}$$

Kekurangan dari kontroler integral adalah memiliki *pole* pada origin seperti ditunjukkan pada Persamaan 2.22 sehingga menyebabkan sistem menjadi tidak stabil. Kehadiran *pole* pada titik origin membuat pengurangan pada kesalahan keadaan tunak ketika sinyal uji *step* diberikan.

2.3.3 Kontroler Differensial

Kontroler proporsional memberikan aksi berdasarkan kesalahan saat ini dan kontroler integral memberikan aksi berdasarkan nilai kesalahan sebelumnya. Kontroler differensial memberikan aksi berdasarkan prediksi nilai kesalahan yang akan datang seperti dirumuskan pada Persamaan 2.23

$$u(t) = K_d \frac{de(t)}{dt} \tag{2.23}$$

Nilai K_d adalah penguatan differensial, sehingga ketika ditransformasi *Laplace* fungsi alih dari kontroler differensial dapat dituliskan dengan rumus yang lebih sederhana pada Persamaan 2.24.

$$\frac{U(s)}{E(s)} = K_d s \tag{2.24}$$

Keuntungan dari kontroler differensial adalah memiliki *zero* pada origin sehingga jika ditambahkan, sistem bisa menjadi lebih stabil.

Kontroler differensial tidak bisa bediri sendiri dikarenakan jika diaplikasikan akan menghasilkan nilai nol, jika kesalahan yang terjadi konstan dan akan memperbesar sinyal kontrol yang diberikan. *Noise* pada frekuensi tinggi akan diperbesar, oleh karena itu bentuk kontroler differensial diubah menjadi Persamaan 2.25.

$$\frac{U(s)}{E(s)} = \frac{K_d s}{\frac{K_d s}{N} + 1} \tag{2.25}$$

2.3.4 Kontroler PID

Kontroler PID terdiri dari tiga komponen utama yaitu kontroler proporsional, kontroler integral, dan kontroler diferensial. Metode perancangan yang digunakan pada penelitian kali ini berdasarkan pada tuning *Chien Hrones Reswick* (CHR) [9]. Alasan memilih metode CHR karena metode tersebut dapat dirancang menggunakan kriteria *disturbance rejection* yang dibutuhkan dalam sistem pengaturan level.

$$G_p(s) = \frac{K}{Ts+1} e^{-\theta s} \tag{2.26}$$

Persamaan 2.26 mempunyai parameter antara lain K adalah penguatan *plant,* T *= time constant,* θ adalah waktu tunda. Parameter PI untuk tuning dengan metode CHR dapat dilihat pada Tabel 2.4.

Tabel 2.4 Parameter PI dengan metode CHR *disturbance rejection*

Tipe	Parameter	Kontroler
0 % *overshoot*	T_I	$2{,}4L$
	K_P	$\frac{0{,}95T}{KL}$
	T_d	$0{,}42L$
20 % *overshoot*	T_I	$2L$
	K_P	$\frac{1{,}2T}{KL}$
	T_d	$0{,}42L$

Masing-masing spesifikasi mempunyai kelemahan dan kekurangan yaitu ketika menggunakan spesifikasi 0 % *overshoot* maka respon sistem akan menjadi lambat dibandingkan dengan 20 % *overshoot*.

2.4 Indeks Performansi [10]

Kontrol optimal tidak dapat didefinisikan dengan tepat. Solusi yang menurut sebuah masalah adalah kondisi optimal, mungkin bukan nilai yang optimal bagi permasalahan lain. Indeks performansi banyak digunakan oleh kalangan praktisi dan juga akademisi untuk membantu dalam menentukan kualitas dari sebuah sistem. Indeks performansi sendiri sebenarnya adalah fungsi hubungan dimana beberapa karakteristik sistem seperti kondisi optimal dari sistem didefinisikan. Indeks performansi secara umum dapat dirumuskan pada Persamaan 2.27 dimana J adalah indeks performansi dan e adalah kesalahan.

$$J = \int_0^\infty f(e)dt \tag{2.27}$$

Nilai indeks performansi ini mengindikasikan seberapa bagus performansi dari suatu sistem. Pada penjelasan selanjutnya akan dibahas mengenai beberapa indeks performansi untuk kriteria kesalahan yang biasa digunakan dalam perancangan kontroler.

2.4.1 *Mean Square Error* (MSE)

Indeks performansi MSE didefinisikan pada Persamaan 2.28. Indeks performansi tersebut secara matematis cocok digunakan untuk mendesain sistem dengan orde rendah.

$$J = \lim_{\tau\to\infty} \frac{1}{2\tau} \int_{-\tau}^{\tau} e^2(t)dt \tag{2.28}$$

Kelebihan dari indeks performansi MSE antara lain mudah dioperasikan secara matematis, biasanya diaplikasikan untuk masukkan berupa *unit step*. Beberapa kekurangan jika meminimumkan nilai MSE dapat mengakibatkan sedikit redaman pada sistem orde tinggi dengan nilai waktu tunak dan *overshoot* yang tinggi. Sistem dengan indeks performansi ini cenderung tidak sensitif terhadap variasi parameter.

2.4.2 *Integral Square Error* (ISE)

Indeks performansi ISE ditunjukkan pada Persamaan 2.29. Indeks performansi ini sangat dekat hubungannya dengan MSE.

$$J = \int_0^\infty e^2(t)dt \tag{2.29}$$

Indeks performansi ISE memiliki karakteristik yang hampir sama dengan indeks performansi MSE. ISE selain digunakan untuk mendesain sebuah kontroler, juga digunakan sebagai kriteria analisa mengenai seberapa baik sebuah sistem dengan kontroler dalam mengatasi pembebanan.

2.4.3 *Integral of Time Multiplied Square Error* (ITSE)

Indeks performansi ITSE ditunjukkan pada Persamaan 2.30. Indeks performansi ITSE secara umum cocok digunakan untuk sistem dengan sinyal uji berupa sinyal *step*.

$$J = \int_0^\infty te^2(t)dt \tag{2.30}$$

Indeks ini memiliki karakteristik untuk menghilangkan kesalahan yang besar dengan cepat seperi kesalahan yang dihasilkan pada fase *transient* sebuah sistem. Perbedaan dengan kriteria ISE hanya terdapat tambahan bobot waktu pada kriteria ITSE.

2.4.4 *Integral of Squared Time Multiplied by Squared Error* (ISTSE)

Indeks performansi ISTSE ditunjukkan pada Persamaan 2.31. ITSE dan ISTSE sangat cocok digunakan pada respon sinyal *step*.

$$J = \int_0^\infty t^2e^2(t)dt \tag{2.31}$$

ISTSE menuju nilai tunak lebih cepat dibandingkan indeks performansi lainnya hal tersebut dikarenakan faktor pengali waktu yang ada padanya. Tetapi ITSE dan ISTSE lebih sensitif terhadap variasi parameter ketika dibandingkan dengan MSE dan ISE.

2.4.5 *Integral of Absolute Value of Kesalahan* (IAE)

Indeks performansi IAE ditunjukkan pada Persamaan 2.32. IAE memiliki struktur kriteria yang paling mudah diaplikasikan karena secara matematis mudah dioperasikan jika dibandingkan dengan kriteria kesalahan lainnya.

$$J = \int_0^\infty |e(t)|dt \tag{2.32}$$

Kriteria ini dapat menghilangkan kesalahan yang besar dan kesalahan yang kecil jika dibandingkan dengan ISE. Tetapi masih kurang dalam hal sensitivitas terhadap variasi parameter dan kecepatan dari respon yang dihasilkan jika dibandingkan dengan ITSE dan ISTSE. Sistem optimal berdasarkan kriteria ini mempunyai nilai redaman dan fase *transient* yang sangat baik.

2.4.6 *Integral of Time Multiplied by Absolute Value of Kesalahan* (ITAE)

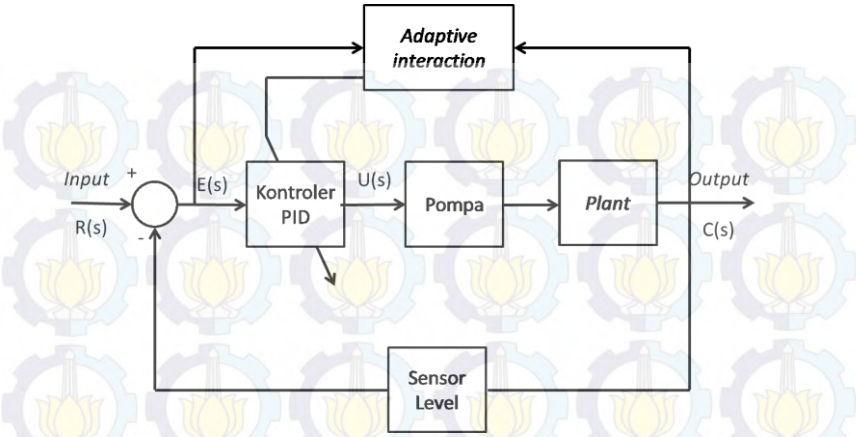
Indeks performansi dari ITAE ditunjukkan pada Persamaan 2.33. Bobot pengali waktu pada ITAE membuatnya lebih cepat untuk meminimalkan kesalahan dibandingkan IAE.

$$J = \int_0^\infty t|e(t)|dt \tag{2.33}$$

Jika dibandingkan dengan ITSE dan ISTSE tetapi bukan dalam hal matematis, ITAE memiliki nilai kesalahan awal yang besar dan kemudian kesalahan menghilang secara perlahan.

2.5 Kontroler *Self-Tuning* PID

Proses *tuning* diperlukan dalam memaksimalkan fungsi kontroler. Proses tuning dapat dibedakan menjadi dua yaitu *offline tuning* dan *online tuning*. Pada penelitian ini akan diterapkan metode *online tuning* seperti Gambar 2.7. Pemilihan metode *online tuning* dikarenakan dapat berubah secara otomatis tanpa campur tangan manusia. Mekanisme perhitungan pada *tuning function* digunakan untuk menemukan nilai yang tepat untuk melakukan tuning pada parameter kontroler utama.



Gambar 2.7 Diagram blok *self-tuning* PID

Pendekatan interaksi adaptif pada penelitian ini akan digunakan sebagai *tuning function*. Interaksi adaptif akan melakukan perhitungan yang digunakan untuk *tuning* terhadap parameter PID yaitu Kp, Ki, dan Kd. Kontroler *self-tuning* dengan pendekatan interaksi adaptif diharapkan dapat membuat kontroler PID lebih tahan terhadap pembebanan.

2.6 Interaksi Adaptif [5]

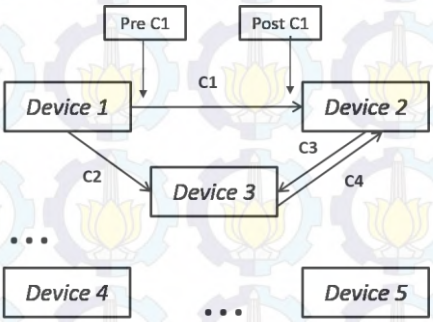
Interaksi adaptif diperkenalkan pertama kali oleh Robert D Brandt dan Feng Lin. Metode ini berawal dari hubungan antara masukan dan keluaran pada Persamaan 2.34. Fungsi waktu (t) dihilangkan untuk memudahkan perhitungan seperti pada Persamaan 2.35.

$$y_n(t) = f_n\left[u_n(t) + \sum_{c \in I_n} \alpha_c y_{prec}(t) \right], n \in N, \quad (2.34)$$

$$y_n = f_n\left[u_n + \sum_{c \in I_n} \alpha_c y_{prec} \right], n \in N, \quad (2.35)$$

Tujuan dari sistem ini adalah menngadaptasikan nilai α_c (*connection weight*) sehingga indeks performansi $E(y_1,\ldots,y_n,u_1,\ldots u_n)$

sebagai fungsi eksternal dari masukan dan keluaran dapat diminiminalkan. Jika α_c diadaptasi dari Persamaan 2.26 dan mempunyai solusi yang unik, maka indeks performansi E akan menurun sesuai waktu. Sehingga dapat ditulis seperti Persamaan 2.36. Interaksi antar komponen penyusun ditunjukkan pada Gambar 2.8.



Gambar 2.8 Interaksi antara komponen keluaran dan masukan

Gambar 2.8 terdiri dari komponen masukkan dan keluaran yang dilambangkan dengan *device*, dimana setiap interaksi antar komponennya memiliki bobot koneksi. Bobot koneksi tersebut antara lain C1, C2, C3, dan C4. Bobot koneksi inilah yang diadaptasi untuk menghasilkan respon yang sesuai. Terdapat beberapa istilah yang perlu dipahami diantaranya *pre* dan *post. Pre* merupakan istilah untuk nilai yang keluar dari sebuah *device* dan *post* adalah nilai yang masuk menuju *device*. Bobot koneksi dirumuskan pada Persaman 2.36 dan 2.37.

$$\dot{\alpha}_c = \left(\sum_{c \in O_n} \alpha_s \dot{\alpha}_s A - \gamma \frac{\partial E}{\partial y_{postc}} \right) \circ F'_{postc}[x_{postc}] \circ y_{prec}, \tag{2.36}$$

$$A = \frac{\frac{dE}{dy_{posts}} \circ F'_{postd}[x_{posts}]}{\frac{dE}{dy_{posts}} \circ F'_{postd}[x_{posts}] \circ y_{postc}} \tag{2.37}$$

Dengan melakukan penyederhanaan sehingga didapatkan solusi yang unik pada Persamaan 2.38.

$$\dot{\alpha}_c = -\gamma \frac{\partial E}{\partial \alpha_c} \tag{2.38}$$

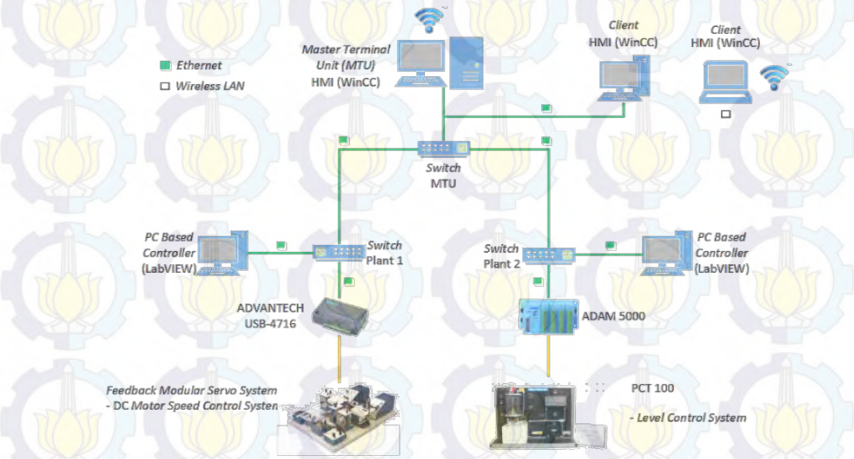
dimana Persamaan 2.38 didekati dengan persamaan rantai sehingga didapatkan pendekatan pada Persamaan 2.39

$$\dot{\alpha}_c = -\gamma \frac{\partial E}{\partial y_{postc}} \circ F'_{postc}[x_{postc}] \circ y_{prec}, \tag{2.39}$$

BAB 3
PERANCANGAN SISTEM

3.1 Gambaran Umum Sistem

Pada penelitian ini dirancang sebuah sistem *Supervisory Control And Data Acquisition* (SCADA) dengan arsitektur seperti pada Gambar 3.1.

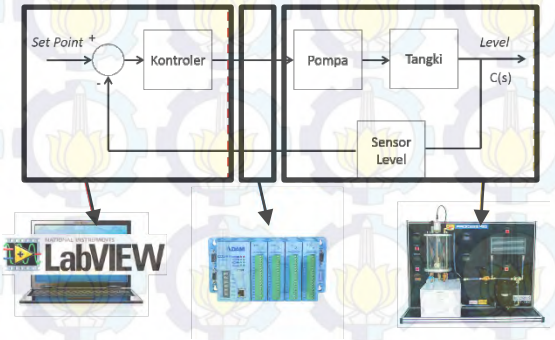


Gambar 3.1 Arsitektur sistem SCADA

Arsitektur sistem SCADA yang dirancang terdiri dari 3 buah komponen utama yaitu *Remote Terminal Unit* (RTU), *Master Terminal Unit* (MTU), dan jaringan komunikasi. Masing-masing RTU mengontrol satu buah *plant*. *Plant* yang dikontrol adalah PCT-100 dan motor DC. Hubungan antara RTU dan plant dilakukan melalui ADAM 5000 L/TCP dan Advantech USB 4716 sebagai akuisisi data. Pemilihan perangkat juga berpengaruh terhadap sistem yang akan dirancang. ADAM 5000 L/TCP memiliki *sampling rate* sebesar 10 S/detik oleh karena itu sesuai digunakan pada PCT-100 karena karakteristik sistemnya yang cenderung lama. Advantech USB 4716 memiliki *sampling rate* 200kS/detik, sehingga sesuai digunakan pada sistem pengaturan kecepatan motor DC yang memiliki karakteristik respon yang cepat. Fokus peneletian saya berada pada sistem pengaturan level PCT-100.

Gambar 3.2 menunjukkan diagram blok sistem pengaturan level yang akan dirancang. Diagram blok yang disusun hampir sama dengan sistem pengaturan level pada umumnya hanya saja terdapat metode *tuning* otomatis terhadap kontroler PID. *Tuning* otomatis tersebut menggunakan prinsip dari interaksi adaptif yang diusulkan oleh Feng Lin dan Robert D. Brandt [5].

Sistem yang akan diimplementasikan seperti Gambar 3.2 terdiri dari beberapa komponen penyusun diantaranya kontroler berupa algoritma program yang dibuat pada LabVIEW. Komputer berisikan LabVIEW akan dihubungkan dengan PCT-100 menggunakan *OLE for Process Control* (OPC). OPC yang digunakan adalah *KepserverEx*. Kepserver akan dihubungkan pada ADAM 5000 L/TCP. ADAM 5000 akan memberikan sinyal kontrol berupada tegangan 0 -10 V kepada pompa dan membaca sinyal analog dari sensor level berupa tegangan 0 – 10 V. Pompa akan bekerja sesuai dengan sinyal kontrol yang diberikan. Level air yang ada didalam tangki proses akan dibaca oleh sensor level dan diumpan balikkan kepada referensi sebagai pembanding. Jika data sensor level masih belum sesuai dengan referensi maka akan dilakukan koreksi oleh kontroler. Selain itu, pada penelitian ini dilakukan pengujian beban yang direpresentasikan dengan *control valve* untuk menguji kinerja dari kontroler yang dibuat.



Gambar 3.2 Diagram blok sistem pengaturan level secara umum

3.2 Pemodelan *Plant*

Pemodelan *plant* dilakukan mengacu pada Gambar 2.5 terdiri dari pemodelan pompa, pipa, tangki dan sensor

3.2.1 Pemodelan Pompa

Mengacu pada Persamaan 2.1 dengan memasukkan rentang debit air yang keluar dari pompa antara 0 - 58.33 cm^3/s. Rentang tegangan masukan antara 0 – 10 V didapatkan hasil pada Persamaan 3.1.

$$K_1 = \frac{58.33 - 0}{10 - 0} = 5.83\, cm^3/sV \tag{3.1}$$

3.2.2 Pemodelan Pipa

Mengacu pada Persaman 2.2 dengan memasukkan data diameter pipa sebesar 1 cm, panjang pipa sebesar 90 cm dan dengan debit air yang masuk 16.67 cm^3/s didapatkan Persamaan 3.2.

$$\theta = \frac{70.65}{16.67} = 4.23s \tag{3.2}$$

Jika dihubungkan dengan sistem pengaturan level maka waktu tunda yang terjadi dapat dituliskan pada Persamaan 3.3.

$$e^{-\theta s} = e^{-4.23s} \tag{3.3}$$

3.2.3 Pemodelan Tangki

Nilai Q_0 didapatkan dengan melakukan uji pengosongan tangki. Diambil titik kerja ketika nilai h_0 sebesar 7.5 cm didapatkan nilai Q_0 seperti pada Persamaan 3.4.

$$Q_0 = \frac{V}{t} = \frac{1507.2}{110} = 13.701 cm^3/s \tag{3.4}$$

V adalah *volume* air yang keluar dan t adalah waktu yang dibutuhkan untuk mengosongkan tangki. Mengacu pada Persamaan 2.4 dan Persamaan 2.9 didapatkan nilai R dan C pada Persamaan 3.5 dan 3.6.

$$R = \frac{7.5}{13.701} = 0.547 s/cm^2 \tag{3.5}$$

$$C = 200.96 cm^2 \tag{3.6}$$

Fungsi alih pada Persamaan 3.7 didapatkan melalui substitusi Persamaan 3.5 dan Persamaan 3.6 pada Persamaan 2.15.

$$\frac{H(s)}{Q_{in}(s)} = \frac{0.547}{109.92s+1} \tag{3.7}$$

3.2.4 Pemodelan Sensor

Mengacu pada Persamaan 2.16 dengan memasukkan rentang tegangan keluaran antara 0 – 10 V dan rentang ketinggian air dalam tangki antara 0 – 18 *cm* data sesuai dengan *user manual* PCT-100 , maka penguatan sensor dirumuskan pada Persamaan 3.8.

$$K_2 = \frac{10-0}{18-0} = 0.55\,\text{V/}cm \tag{3.8}$$

Mengacu pada Persamaan 2.17 dapat dihitung fungsi alih keseluruhan dari plant dengan cara mengalikan semua komponen pemodelan seperti pada Persamaan 3.9.

$$\frac{H(s)_{total}}{Q_{in}(s)_{total}} = 5.83 * e^{-4.23s} * \frac{0.547}{109.92s+1} * 0.55 \tag{3.9}$$

$$\frac{H(s)_{total}}{Q_{in}(s)_{total}} = \frac{1.753e^{-4.23s}}{109.92s+1} \tag{3.10}$$

3.3 Perancangan Kontroler PID

Sistem yang akan dirancang memiliki spesifikasi respon yang cepat dengan *overshoot* 0 %, dan mampu mengatasi pembebanan. Jika *plant* memiliki karakteristik seperti Persamaan 3.11.

$$G_p(s) = \frac{K}{Ts+1} e^{-\theta s} \tag{3.11}$$

Dimana K bernilai 1.753 dan T bernilai 109.92 dan θ bernilai 4.23. Jika perancangan kontroler menggunakan kriteria *overshoot* 0 %. Pemilihan tersebut dikarenakan sistem pengaturan proses membutuhkan kontroler yang bersifat *disturbance rejection*. Jika dimasukkan nilai pada masing-masing parameter pada Tabel 2.4 maka didapatkan nilai T_i pada

Persamaan 3.12, nilai K_p pada Persamaan 3.13, dan nilai T_d pada Persamaan 3.14.

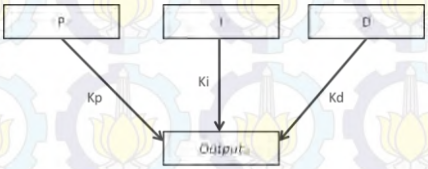
$$T_i = 10.152s \tag{3.12}$$

$$K_P = \frac{0.95*109.92}{1.753*4.23} = 14.17 \tag{3.13}$$

$$T_d = 1.77s \tag{3.14}$$

3.4 Perancangan Kontroler *Self-Tuning* PID dengan Interaksi Adaptif

Kontroler *self-tuning* PID dengan interaksi adaptif yang akan dirancang mengacu pada skema interaksi yang ditunjukkan pada Gambar 3.3. Terdapat tiga masukkan, satu keluaran, dan interaksi antara masukkan dan keluaran diberi bobot koneksi yang direpresentasikan dengan Kp, Ki, dan Kd. Bobot itulah yang menjadi tujuan utama adaptasi.



Gambar 3.3 Interaksi antar komponen masukkan dan keluaran

Penyederhanaan dari Persamaan 2.42 jika mengacu pada Gambar 3.3 menjadi tiga buah persamaan yaitu Persamaan 3.15 untuk penguatan proporsional, Persamaan 3.16 untuk penguatan integral, dan Persamaan 3.17 untuk penguatan differensial.

$$\dot{K}_p = -\gamma \frac{\partial E}{\partial y_{postc}} \circ F'_{postc}[x_{postc}] \circ y_{prec}, \tag{3.15}$$

$$\dot{K}_I = -\gamma \frac{\partial E}{\partial y_{postc}} \circ F'_{postc}[x_{postc}] \circ y_{prec}, \tag{3.16}$$

$$\dot{K}_d = -\gamma \frac{\partial E}{\partial y_{postc}} \circ F'_{postc}[x_{postc}] \circ y_{prec}, \tag{3.17}$$

Keluaran dari masing-masiing kontroler P,I, dan D berturut-turut adalah y_1, y_2 *dan* y_3. Desain kontroler *self-tuning* PID dengan interaksi adaptif digunakan tiga kriteria kesalahan ditunjukkan pada Persamaan 3.18 mewakili perancangan menggunakan indeks performansi SE, Persamaan 3.19 mewakili perancangan dengan indeks performansi TSE, dan Persamaan 3.20 mewakili perancangan dengan indeks performansi STSE.

$$E_1 = e^2 = (u - y_4)^2 \tag{3.18}$$

$$E_2 = te^2 = t(u - y_4)^2 \tag{3.19}$$

$$E_3 = t^2e^2 = t^2(u - y_4)^2 \tag{3.20}$$

Dari Persamaan 3.15 dapat dilakukan analisis bahwa bobot pada setiap komponen P, I, dan D nilainya bergantung pada kesalahan, *frechet derivative*, dan keluaran dari masing-masing komponen. Untuk menghitung *frechet derivative*, digunakan Persamaan 3.21 yang merupakan representasi dari turunan pengautan proporsional, intergral, dan differensial

$$F[x] = \int_0^t f(x(\tau), \tau) d\tau \tag{3.21}$$

Persamaan 3.21 dapat didekati dengan *Gateaux differential* [8]. Ketika $f = (f_1, f_2, \ldots, f_x)$ adalah turunan parsial kontinyu maka persamaan 3.21 dapat disubstitusikan ke Persamaan 3.22.

$$\delta F[x,h] = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} h_i \tag{3.22}$$

$$\delta F[x,h] = \int_0^t f_x(x(\tau), \tau) h(\tau) d\tau \tag{3.23}$$

Persamaan 3.23 diubah menjadi Persamaan 3.24 menggunakan pendekatan *Gateaux differential* [8].

$$F'[x] \circ h = \int_0^t f(x(\tau), \tau) h(\tau) d\tau \quad (3.24)$$

Sistem degan sifat *Linear Time Invariant* (LTI) dan fungsi alih G(s), sehingga dilakukan pendekatan dengan konvolusi pada Persamaan 3.25 dan didapatkan persamaan baru pada Persamaan 3.26.

$$F[x] = g(t) * x(t) = \int_0^t x(\tau) g(t-\tau) d\tau, \quad (3.25)$$

Frechet derivative dapat didekatai dengan β

$$F'[x] \circ h = \beta h \quad (3.26)$$

3.4.1 Berdasarkan *Squared Error* (SE)

Indeks performansi dari SE ditunjukkan oleh Persamaan 3.18. Substitusikan Persamaan 3.18 ke Persamaan 3.15 untuk bobot koneksi Kp, sehingga didapatkan Persamaan 3.30.

$$\dot{K}_p = -\gamma \frac{\partial e^2}{\partial y_4} \circ F'_4[x_4] \circ y_1, \quad (3.27)$$

$$\dot{K}_p = -\gamma \frac{\partial (u - y_4)^2}{\partial y_4} \circ F'_4[x_4] \circ y_1, \quad (3.28)$$

$$\dot{K}_p = 2\gamma (u - y_4) \circ F'_4[x_4] \circ y_1, \quad (3.29)$$

$$\dot{K}_p = 2\gamma e F'_4[x_4] \circ y_1, \quad (3.30)$$

Substitusikan Persamaan 3.26 ke Persamaan 3.30 sehingga ketergantungan dari *plant* dapat dihilangkan.

$$\dot{K}_p = 2\gamma e \beta y_1 \quad (3.31)$$

dengan memasukkan nilai 2β kedalam γ pada Persamaan 3.31.

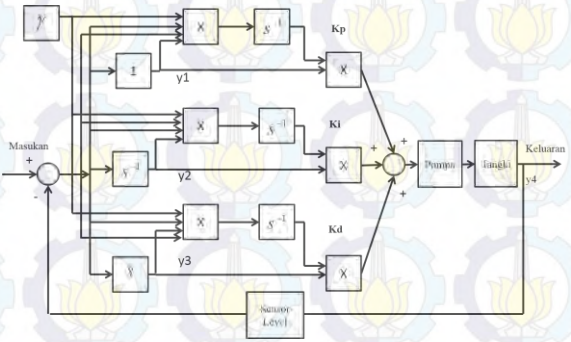
$$\dot{K}_p = \gamma e y_1 \tag{3.32}$$

Menerapkan penurunan rumus yang sama pada bobot koneksi Ki dan Kd di Persamaan 3.16 dan 3.17

$$\dot{K}_I = \gamma e y_2 \tag{3.33}$$

$$\dot{K}_d = \gamma e y_3 \tag{3.34}$$

Diagram blok sistem yang akan dirancang dengan indeks performansi SE ditunjukkan pada Gambar 3.4.



Gambar 3.4 Diagram blok sistem pengaturan level dengan kriteria SE

3.4.2 Berdasarkan *Time Squared Error* (TSE)

Indeks performansi dari TSE ditunjukkan oleh Persamaan 3.19. Substitusikan Persamaan 3.19 ke Persamaan 3.15 untuk bobot koneksi K_p, sehingga dapat dirumuskan pada Persamaan 3.38.

$$\dot{K}_p = -\gamma \frac{\partial e^2}{\partial y_4} \circ F'_4[x_4] \circ y_1, \tag{3.35}$$

$$\dot{K}_p = -\gamma \frac{\partial t(u - y_4)^2}{\partial y_4} \circ F'_4[x_4] \circ y_1, \tag{3.36}$$

$$\dot{K}_p = 2t\gamma(u - y_4) \circ F'_4[x_4] \circ y_1, \tag{3.37}$$

$$\dot{K}_p = 2t\gamma eF'_4[x_4] \circ y_1, \tag{3.38}$$

Substitusikan Persamaan 3.26 ke Persamaan 3.38 sehingga ketergantungan dari *plant* dapat dihilangkan.

$$\dot{K}_p = 2t\gamma e\beta y_1 \tag{3.39}$$

Dilakukan penyederhanaan dengan memasukkan nilai 2β kedalam γ pada Persamaan 3.39.

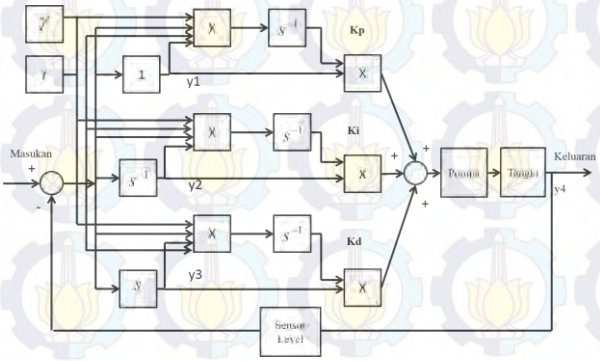
$$\dot{K}_p = t\gamma e y_1 \tag{3.40}$$

Dengan menerapkan penurunan rumus pada bobot koneksi dan sehingga persamaan bobot koneksi dapat disederhanakan menjadi Persamaan 3.41 untuk K_i dan 3.42 untuk K_d.

$$\dot{K}_I = t\gamma e y_2 \tag{3.41}$$

$$\dot{K}_d = t\gamma e y_3 \tag{3.42}$$

Dari penurunan rumus dapat digambarkan sistem yang akan dirancang dengan indeks performansi TSE ditunjukkan oleh Gambar 3.5.



Gambar 3.5 Diagram blok sistem pengaturan level dengan kriteria TSE

3.4.3 Berdasarkan *Squared Time Squared Error* (STSE)

Indeks performansi dari STSE ditunjukkan oleh Persamaan 3.20. Substitusikan Persamaan 3.20 ke Persamaan 3.15 untuk bobot koneksi K_p, sehingga dapat dirumuskan Persamaan 3.46.

$$\dot{K}_p = -\gamma \frac{\partial e^2}{\partial y_4} \circ F'_4[x_4] \circ y_1, \tag{3.43}$$

$$\dot{K}_p = -\gamma \frac{\partial t^2 (u - y_4)^2}{\partial y_4} \circ F'_4[x_4] \circ y_1, \tag{3.44}$$

$$\dot{K}_p = 2t\gamma(u - y_4) \circ F'_4[x_4] \circ y_1, \tag{3.45}$$

$$\dot{K}_p = 2t\gamma e F'_4[x_4] \circ y_1, \tag{3.46}$$

Substitusikan Persamaan 3.26 ke Persamaan 3.46 sehingga ketergantungan dari *plant* dapat dihilangkan dan metode ini dapat diterapkan pada banyak jenis sistem.

$$\dot{K}_p = 2t\gamma e\beta y_1 \tag{3.47}$$

Dilakukan penyederhanaan dengan memasukkan nilai 2β kedalam γ pada Persamaan 3.47 sehingga didapatkan Persamaan 3.48.

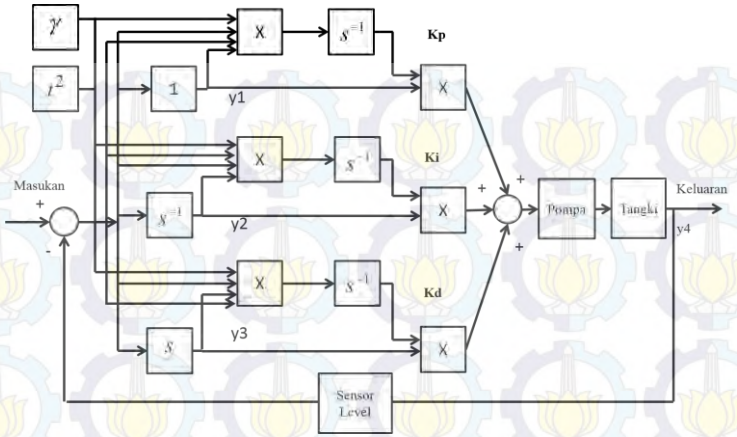
$$\dot{K}_p = t\gamma e y_1 \tag{3.48}$$

Dengan menerapkan penurunan rumus pada bobot koneksi dapat disederhanakan menjadi Persamaan 3.49 untuk K_i dan Persamaan 3.50 untuk K_d.

$$\dot{K}_I = t\gamma e y_2 \tag{3.49}$$

$$\dot{K}_d = t\gamma e y_3 \tag{3.50}$$

Dari penurunan rumus diatas dapat digambarkan sistem yang akan dirancang dengan indeks performansi STSE ditunjukkan pada Gambar 3.6.



Gambar 3.6 Diagram blok sistem pengaturan level dengan kriteria STSE

(Halaman ini sengaja dikosongkan)

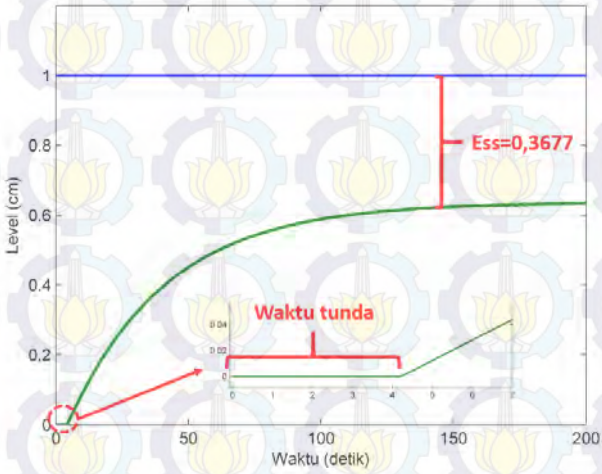
BAB 4
PENGUJIAN DAN ANALISIS

4.1 Pengujian Sistem Tanpa Kontroler

Berdasarkan hasil pemodelan sistem pada Bab 3 didapatkan model akhir *plant* seperti pada Persamaan 4.1

$$\frac{H(s)_{total}}{Q_{in}(s)_{total}} = \frac{1.753e^{-4.23s}}{109.92s+1} \tag{4.1}$$

Dilakukan pengujian sistem umpan balik tertutup menggunakan sinyal *unit step* pada Persamaan 4.1 didapatkan hasil respon pada Gambar 4.1.



Gambar 4.1 Respon *closed loop* sistem pengaturan level tanpa kontroler

Berdasarkan hasil pengujian sistem umpan balik tertutup tanpa kontroler masih terdapat waktu tunda yang disebabkan oleh panjang pipa untuk mengalirkan air dari pompa menuju tangki proses. Waktu tunda berbanding terbalik dengan debit air yang masuk dalam pipa.

Waktu tunda dapat mengakibatkan keterlambatan kontroler dalam memberikan sinyal kontrol sehingga aktuator akan menerima sinyal yang bukan seharusnya. Dalam dunia industri hal tersebut mengakibatkan aktuator bekerja terlambat yang biasanya dapat berdampak terhadap seluruh *line* produksi.

Selain adanya waktu tunda yang terjadi akibat pipa dari pompa ke tangki proses, terdapat kesalahan keadaan tunak sebesar 0,3677. Kesalahan keadaan tunak tersebut yang menjadikan dasar perlunya kontroler pada sistem pengaturan level PCT-100. Spesifikasi yang diinginkan adalah mampu menghilangkan kesalahan keadaan tunak dan berkarakteristik *disturbance rejection*.

4.2 Pengujian Sistem Dengan Kontroler PID

Pengujian sistem dengan kontroler PID bertujuan untuk mengetahui seberapa mampu kontroler PID untuk mengatasi berbegai permasalahan yang biasa ada di dunia industri. Pengujian sistem dilakukan menggunakan sinyal *step* dengan menggunakan parameter kontroler PID yang sudah dirancang pada Bab 3. Diberikan *set point* berupa sinyal *unit step* pada masing-masing variasi pengujian. Terdapat lima buah simulasi pengujian yang dilakukan antara lain.

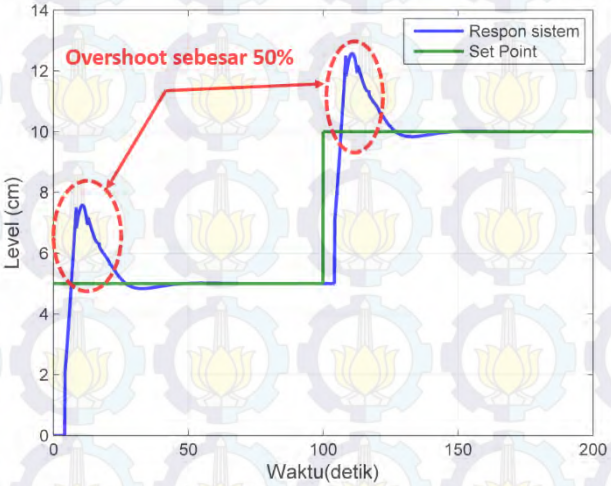
- Pengujian dengan perubahan *set point*
- Pengujian dengan variasi nilai Kp
- Pengujian dengan variasi nilai Ti
- Pengujian dengan variasi nilai Td
- Pengujian dengan pembebanan
- Pengujian dengan *noise*

4.2.1 Pengujian Dengan Perubahan *Set point*

Pengujian terhadap perubahan *set point* bertujuan untuk membuktikan kerja kontroler terhadap perubahan nilai *set point*. Simulasi pengujian dilakukan dengan cara memberikan sinyal uji *unit step*, pada detik ke 50 nilai *set point* diubah menjadi 5. Didapatkan hasil bahwa sistem dengan kontroler PID mampu mengikuti perubahan setiap nilai *set point* yang diberikan seperti Gambar 4.2. Nilai dari *time constant* dan *rise time* dari setiap *set point* juga tidak mengalami perubahan karena jika kontroler PID di-*tuning* dengan parameter sama, maka akan menghasilkan spesifiaksi sama pada setiap perubahan *set point*. Gambar 4.2 menunjukkan bahwa sistem memiliki kesalahan keadaan tunak =0

untuk semua nilai *set point* sehingga sistem tersebut bisa dikatakan stabil. Sistem masih memiliki *overshoot* sebesar 50 % untuk setiap perubahan nilai *set point*. *Overshoot* tersebut terjadi karena *closed loop* sistem setelah diberi kontroler menjadi sistem orde dua berkarakteristik *underdamped* dengan nilai $\zeta = 0.67$.

Ketika terjadi perubahan nilai *set point* parameter dari *plant* tidak ada yang berubah seperti nilai τ yang tetap sesuai hasil desain yaitu pada 10 detik.



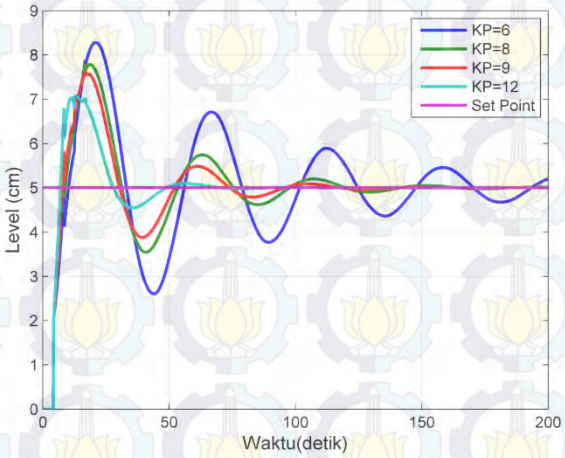
Gambar 4.2 Respon sistem dengan kontroler PID saat terjadi perubahan *set point*

4.2.2 Pengujian Dengan Variasi Nilai K_p

Pengujian terhadap perubahan nilai K_p bertujuan untuk membuktikan dan memahami pengaruh nilai K_p terhadap kecepatan respon dan kesalahan keadaan tunak pada sistem. Dalam Penelitian ini digunakan sistem dengan kontroler adaptif. Salah satu parameter yang diadaptasi adalah K_p sehingga bisa dipastikan nilainya tidak konstan atau tetap pada satu nilai saja melainkan bergerak mengikuti perubahan

yang terjadi pada parameter *plant*. Simulasi pengujian dilakukan dengan cara memberikan sinyal *step* sebesar 5 kemudian dilakukan variasi nilai K_p ditunjukkan oleh Gambar 4.3. Pada Gambar 4.3 menunjukkan bahwa sistem memiliki kesalahan keadaan tunak hampir mendekati 0 namun dengan waktu pencapaian yang berbeda-beda.

Semakin besar nilai K_p maka nilai τ akan semakin cepat. Pada Gambar 4.3, ketika nilai K_p bernilai 12 maka respon cepat menuju nilai keadaan tunaknya, sedangkan dengan K_p bernilai 6 cenderung lambat pada detik ke 200 belum mencapai nilai keadaan tunaknya. Namun jika nilai K_p terlalu besar maka yang terjadi adalah respon akan berosilasi. Hal tersebut dikarenakan nilai kesalahan akan terus dikalikan sehinga menuju nilai yang besar dan berosilasi.

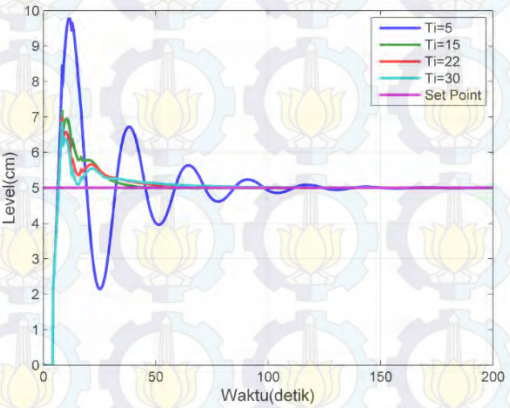


Gambar 4.3 Respon ssstem dengan kontroler PID untuk variasi K_p

4.2.3 Pengujian Dengan Variasi Nilai T_i

Pengujian terhadap perubahan nilai T_i bertujuan untuk membuktikan dan memahami pengaruh nilai T_i terhadap kemampuannya

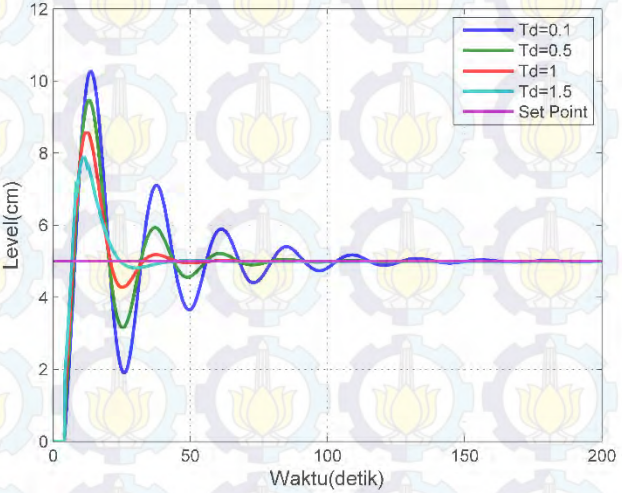
dalam menghilangkan kesalahan keadaan tunak. Simulasi pengujian dilakukan dengan cara memberikan sinyal uji berupa*t step* sebesar 5 kemudian divariasikan nilai T_i ditunjukkan oleh Gambar 4.4. Gambar 4.4 menunjukkan bahwa sistem memiliki kesalahan keadaan tunak hampir mendekati 0 namun dengan waktu pencapaian yang berbeda-beda. Ketika nilai T_i kita perbesar maka kesalahan akan dapat dihilangkan namun efeknya adalah respon menjadi lambat untuk menuju keadaan tunak. Ketika kita beri nilai T_i yang kecil respon akan terdapat *overshoot*. Ada tidaknya *overshoot* dapat dibuktikan secara matematis. Letak T_i yang berada pada denumerator, menyebabkan sistem loop tertutup menjadi orde dua. Karakteristik respon sistem orde dua dibedakan menjadi tiga yaitu *overdamped, critically damped*, dan *underdamped*. Ketika nilai $\zeta > 0$ maka respon akan cenderung lambat namun tanpa adanya overshoot, hal ini dikarenakan nilai T_i yang terlalu besar. Ketika nilai T_i terlalu kecil maka sistem akan memiliki karakteristik underdamped. Sistem underdamped memiliki overshoot yang besarnya sebanding dengan nilai ζ. Ketika diberi nilai T_i sebesar 30 maka sistem akan lebih cepat dalam menghilangkan kesalahan keadaan tunak, sedangkan ketika nilai T_i 5 respon sistem berosilasi.



Gambar 4.4 Respon sistem dengan kontroler PID untuk variasi T_i

4.2.4 Pengujian Dengan Variasi Nilai T_d

Pengujian sistem dengan variasi nilai T_d bertujuan untuk mengetahui peran kontroler differensial dalam sebuah sistem. Kontroler differensial memiliki fungsi untuk mengurangi *overshoot* yang terjadi. Pengujian dilakukan dengan cara memberikan sinyal uji berupa step sebesar 5 dengan nilai K_p dan T_i tetap, sedangkan nilai T_d divariasikan dari 0.1 - 1.5 seperti Gambar 4.5. Pada Gambar 4.5 dapat disimpulkan bahwa semakin besar nilai T_d maka semakin sedikit *overshoot* yang terjadi. Berkurangnya *overshoot* karena kontroler differensial memiliki *zero* pada origin sehingga sistem bisa menjadi lebih stabil.

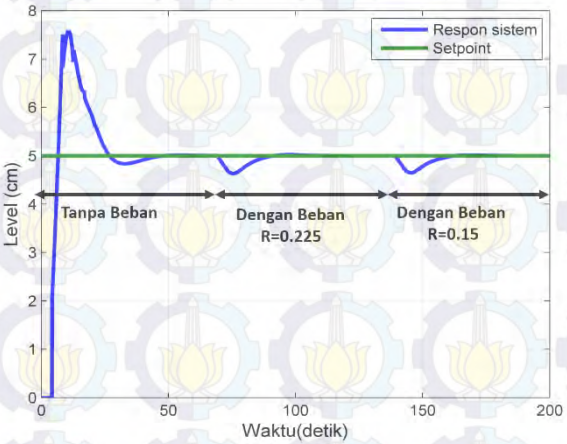


Gambar 4. 5 Respon sistem dengan kontroler PID untuk variasi T_d

4.2.5 Pengujian Dengan Pembebanan

Pemberian beban bertujuan menguji seberapa cepat kontroler mampu kembali menuju kondisi awal ketika diberikan gangguan. Selain kecepatan, besarnya lonjakan dan penurunan nilai yang terjadi akibat pembebanan juga menjadi kriteria yang dianalisis. Pembebanan dilakukan dengan mengubah nilai R atau bukaan *control valve* yang sudah

dimodelkan pada Bab 3. Semakin besar bukaan *control valve* maka semakin besar pula debit air yang keluar tangki, sehingga nilai R semakin kecil. Dari Gambar 4.6 bisa dilihat terjadi penurunan nilai saat dilakukan pembebanan detik ke 65 dan 135. Setelah adanya penurunan nilai sistem dapat kembali menuju kondisi semula dengan waktu 45,5 detik pada pembebanan pertama dan 46,6 detik pada pembebanan kedua. *Recovery time* yang dihasilkan masih terlalu lama dan penurunan nilai yang dihasilkan masih terlalu besar. Dari pengujian pembebanan pada sistem dengan kontroler PID didapatkan nilai RMSE yang masih terlalu besar yaitu 13,88 %. Besarnya nilai RMSE diakibatkan kontroler PID belum mampu beradaptasi terhadap perubahan parameter *plant* yang diakibatkan oleh pembebanan. Hasil simulasi ketika diberi gangguan berupa pembebanan dapat dilihat pada Gambar 4.6.

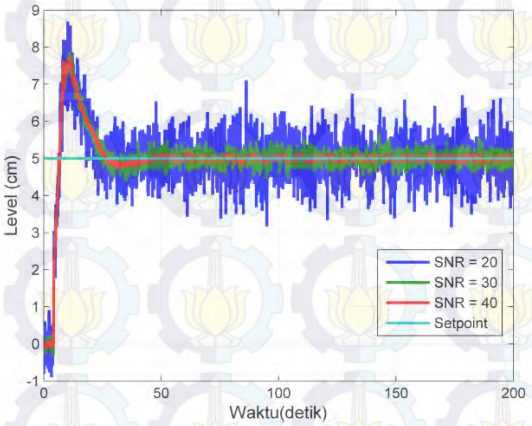


Gambar 4.6 Respon saat diberikan gangguan berupa efek pembebanan

4.2.6 Pengujian Dengan Menggunakan *Noise*

Pengujian *noise* dalam sistem bertujuan untuk mengukur seberapa sensitif sistem dengan sebuah kontroler terhadap perubahan kecil. Selain itu pengujian ini merepresentasikan kondisi pada *real plant*. *Real plant* pada dunia industri memiliki *measurement noise* yang mengakibatkan kesalahan dalam pembacaan sensor. Simulasi yang dilakukan dengan cara

menambahkan *noise white gaussian* terhadap sistem. Pada Gambar 4.7 dapat dilihat bahwa semakin besar nilai *Signal Noise Ratio* (SNR) dari *noise* yang diberikan maka semakin dia tidak berpengaruh. Hal tersebut dapat juga dibuktikan dari nilai RMSE yang dihasilkan. Dari Gambar 4.7 didapatkan nilai RMSE sebesar 56.23 % untuk SNR 20, 18.63 % untuk SNR 30, dan 5.55 % untuk SNR 40.



Gambar 4.7 Respon saat diberikan gangguan *noise*

4.3 Pengujian Sistem Dengan Kontroler *Self-tuning* PID Interaksi Adaptif

Pengujian dilakukan menggunakan sinyal step dengan menggunakan desain kontroler yang sudah dirancang pada Bab 3. Terdapat empat buah simulasi pengujian yang dilakukan antara lain.

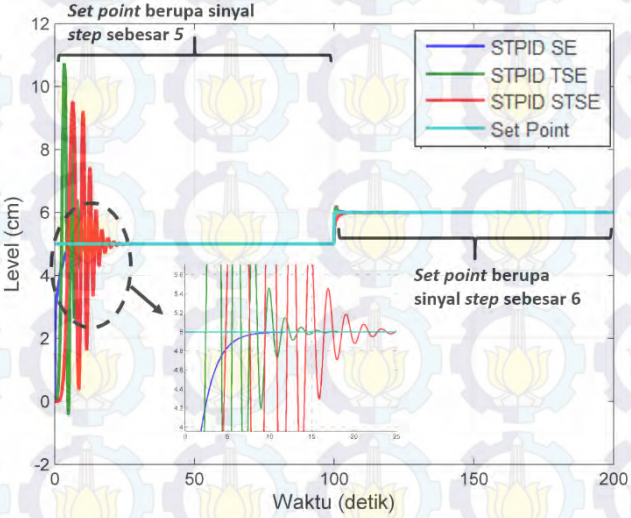
- Pengujian dengan perubahan *set point*
- Pengujian dengan variasi nilai gamma
- Pengujian dengan pembebanan
- Pengujian dengan *noise*

4.3.1 Pengujian Dengan Perubahan *Set point*

Pengujian dengan perubahan *set point* bertujuan untuk mengetahui kerja proses adaptasi yang dilakukan oleh kontroler adaptif untuk menghadapi perubahan *set point* yang terjadi. Simulasi dilakukan dengan

menggunakan sinyal uji berupa *step* sebesar 5 pada detik ke 0 – 100 dan diubah menjadi sinyal *step* sebesar 6 pada detik ke 100 -200. Dari Gambar 4.8 dapat dilihat bahwa sistem mampu beradaptasi untuk menuju nilai *set point*. Konstanta adaptif yang digunakan untuk SE sebesar 0,05, TSE sebesar 0.005, sedangkan untuk STSE sebesar 0.0001. Perbedaan tersebut dikarenakan STSE bernilai tak hingga ketika diberi konstanta adaptif yang terlalu besar. Hasil pengujian ditunjukkan pada Gambar 4.8.

Dimana untuk kriteria STSE lebih lambat menghilangkan kesalahan dibandingkan dengan kriteria yang lainnya dikarenakan nilai konstanta adaptifnya yang leih kecil dibandingkan yang lainnya. Tetapi kontroler dengan kriteria STSE masih mampu untuk mengikuti perubahan *set point* yang terjadi. Jika kita bandingakan kecepatan antara SE dan TSE maka jelas TSE lebih unggul karena terdapat bobot pengali waktu pada proses adaptasinya.

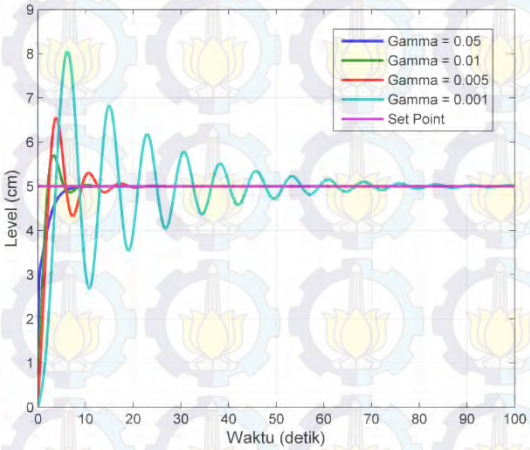


Gambar 4.8 Respon sistem dengan perubahan nila *set point*

4.3.2 Pengujian Dengan Perubahan Nilai Gamma

Pengujian dengan variasi gamma dilakukan untuk melihat pengaruh nilai gamma terhadap cepat lambatnya sistem dalam beradaptasi. Hasil simulasi dapat dilihat pada Gambar 4.9. Pengujian

dilakukan dengan cara memberikan variasi nilai gamma dengan *set point* yang sama. Variasi gamma yang digunakan adalah 0.05, 0.01, 0.005, dan 0.001. Semakin besar nilai gamma, maka semakin cepat proses adaptasi dan semakin cepat pula respon sistem untuk menuju nilai keadaan tunak. Selain terhadap kecepatan besarnya nilai gamma juga berpengaruh kepada kemampuan system dalam menghilangkan kesalahan keadaan tunak. Jika nilai gamma terlalu besar maka akan menimbulkan *overshoot* pada sistem, tetapi ketika nilai gamma terlalu kecil akan menyebabkan sistem tidak mampu mencapau nilai *set point* atau bisa dikatakan masih terdapat kesalahan keadaan tunak. Penentuan nilai gamma pada penelitian ini masih dilakukan dengan cara trial and kesalahan. Metode penentuan otomatis nilai gamma masih belum diterapkan. Sehingga pada setiap pengujian masih digunakan nilai gamma yang berbeda-beda sesuai dengan kebutuhan pengujian.

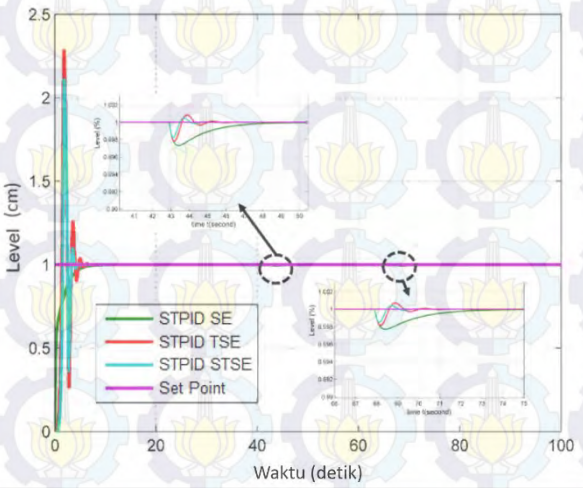


Gambar 4.9 Respon sistem dengan variasi nilai gamma

4.3.3 Pengujian Dengan Pembebanan

Dilakukan pengujian sistem dengan kontroler *self-tuning* PID interaksi adaptif melalui pemberian beban ketika respon sudah mencapai keadaan tunaknya. Dari Gambar 4.10 dapat dilihat pada saat detik ke 43 dan 68 sistem diberi gangguan berupa pembebanan dengan cara

mengubah parameter *plant* yaitu R. Ketika Sistem diberikan beban tetapi dapat kembali ke kondisi semula berarti sistem dapat dikatakan stabil. Hal tersebut dikarenakan sistem mampu beradaptasi dengan penurunan nilai yang terjadi. Contoh penerapan nyata dari pembebanan pada system pengaturan level adalah adanya air yang dikeluarkan dari *drain valve*. Dalam hal ini bukaan dari *drain valve* merepresentasikan nilai R. Semakin besar *drain valve* dibuka maka air yang mengalir semakin cepat dan dianggap semakin besar pula nilai R. Kontroler dengan kriteria STSE terbukti mampu lebih cepat dalam mengatasi beban yang diberikan dengan waktu 1.7 detik pada pembebanan pertama dan 2.2 detik pada pembebanan kedua. Sedangkan kriteria TSE sebesar 2.8 detik pada pembebanan pertama dan 3.5 detik pada pembebanan kedua. Serta kriteria SE sebesar 6.6 detik pada pembebanan pertama dan 6.4 detik pada pembebanan kedua. Tentu saja semakin kecepatan respon tersebut akan menimbulkan *overshoot* pada respon seperti terlihat pada Gambar 4.10.



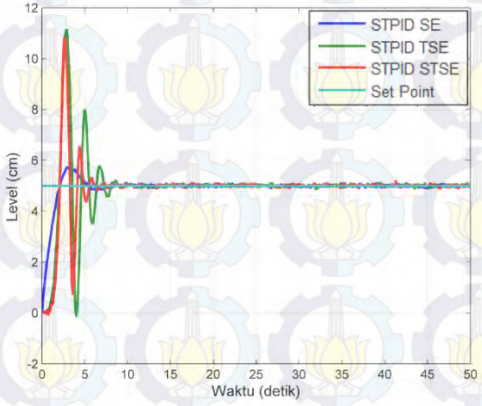
Gambar 4.10 Respon sistem ketika diberi gangguan

Kriteria STSE lebih cepat namun terdapat *overshoot* sedangkan kriteria SE lebih lambar tapi tanpa ada *overshoot*. Dari hasil pengujian ini telah terbukti bahwa kontroler *self-tuning* PID interaksi adaptif memiliki

karakteristik *disturbance rejection* atau mengatasi gangguan. Seperti apa sistem dalam mengatasi pembebanan bisa didesain melalui ketiga kriteria *kesalahan* yaitu SE, TSE, dan STSE. Jika diinginkan penanganan yang cepat maka dipilih STSE. Jika diinginkan tanpa adanya *overshoot* maka dipilih kriteria SE.

4.3.4 Pengujian Dengan *Noise*

Pengujian dengan *noise* digunakan untuk menerapkan sistem seakan-akan dalam *real* sistem. Pada *real* sistem pasti terdapat *measurement noise* yang mengakibatkan perubahan pembacaan pada sensor. Pengujian sistem degan kontroler *self-tuning* PID interaksi adaptif melalui pemberian *noise* pengukuran pada nilai keluaran. *Noise* yang diberikan berupa *additive white gaussian noise* atau yang sering disebut *noise* yang terjadi saat pengukuran. Pengujian dilakukan dengan menggunakan sinyal *step* sebesar 5 dan konstanta adaptif sebesar 1. Hasil dari simulasi pengujian dapat dilihat pada Gambar 4.11. Dari Gambar 4.11 dapat dilihat bahwa sistem tidak terlalu terpengaruh dengan adanya *noise* karena kesalahan keadaan tunak yang terjadi tidak lebih dari 6%. Respon sistem dengan *noise* hampir sama dengan respon sistem tanpa *noise*. Hal tersebut dibuktikan dengan menghitung nilai RMSE. Didapatkan hasil RMSE untuk kriteria SE sebesar 5,0023 %, untuk kriteria TSE sebesar 5.056 %, dan untuk kriteria STSE sebesar 5.024 %.



Gambar 4.11 Respon sistem ketika diberikan *noise* berupa *white gaussian*

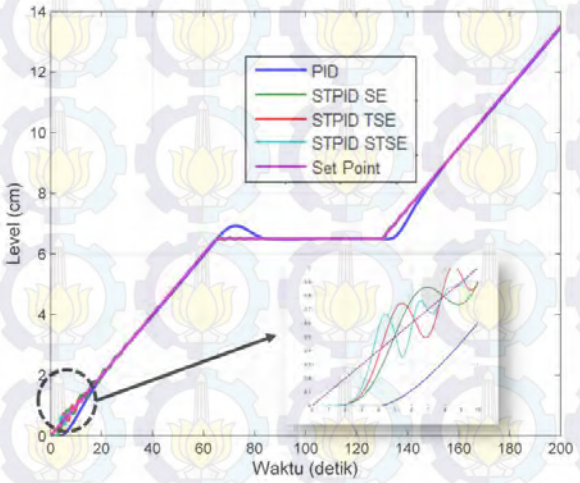
4.4 Perbandingan Antara Sistem Dengan Kontroler PI Dan Kontroler *Self-tuning* PID Interaksi Adaptif

Perbandingan dilakukan menggunakan dua buah simulasi pengujian yang dilakukan antara lain

- Pengujian dengan *tracking* sinyal *ramp*
- Pengujian dengan pembebanan

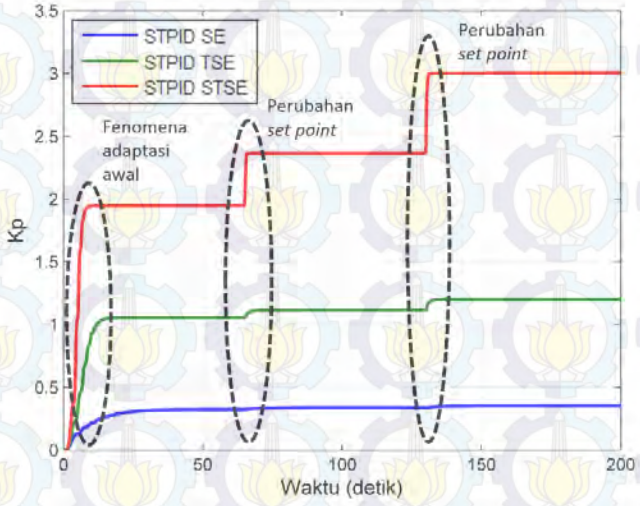
4.4.1 Pengujian Dengan Tracking Sinyal Ramp

Pengujian ini berfungsi untuk melihat seberapa baik peformansi respon *transient* suatu sistem terhadap beberapa kriteria kesalahan yang digunakan dalam Perancangan kontroler. Ketika dilakukan perbandingan pada Gambar 4.12 terlihat sistem dengan kontroler PID dan *self-tuning* PID interaksi adaptif mampu mengikuti *set point* berupa sinyal *ramp*. Dari Gambar 4.12 terlihat bahwa sistem dengan kontroler *self-tuning* PID interaksi adaptif dengan kriteria STSE mampu lebih cepat untuk mengurangi kesalahan dibandingkan dengan kriteria lainnya. Tetapi dengan *overshoot* awal yang besar. Dilihat dari nilai RMSE yang didapat untuk PID sebesar 18.78 %, untuk kontroler dengan kriteria SE sebesar 5.83 %, kriteria TSE sebesar 4.57 %, dan kriteria STSE sebesar 3.64 %.



Gambar 4.12 Perbandingan Kontroler dengan sinyal *ramp*

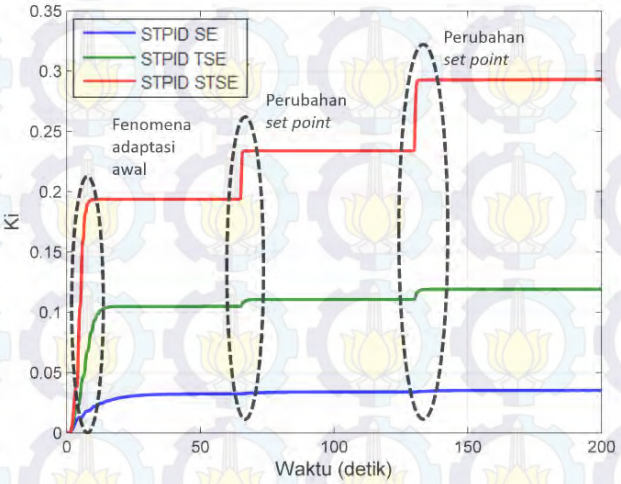
Dari perhitungan RMSE dapat disimpulkan bahwa sistem dengan kontroler *self-tuning* PID interaksi adaptif dengan kriteria STSE lebih baik jika digunakan untuk menghilangkan kesalahan. Berbeda halnya dengan pengujian yang dilakukan sebelumnya dimana SE memiliki nilai RMSE yang lebih kecil. STSE masih lebih unggul dibandingkan dengan yang lainnya. Hasil simulasi perubahan parameter kontroler dapat dilihat pada Gambar 4.13 – 4.15. Perbandingan nilai K_p dapat dilihat pada Gambar 4.13, terlihat ada beberapa fenomena yang dapat diamati yaitu adaptasi nilai K_p awal untuk menuju keadaan tunak dan adaptasi saat terjadi perubahan *set point*. Parameter K_p dari kontroler PID yang tidak dapat berubah menyebabkan sistem memiliki RMSE yang lebih tinggi dibandingkan dengan kontroler adaptif.



Gambar 4.13 Perubahan nilai Kp ketika *tracking* sinyal *ramp*

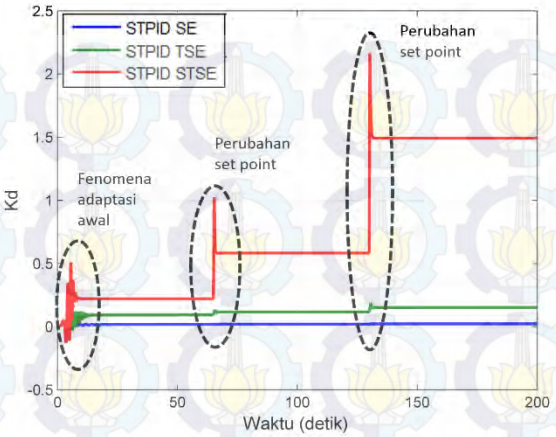
Semakin besar nilai K_p maka semakin cepat respon sistem yang dihasilkan. Hal tersebut ditunjukkan oleh kontroler adaptif dengan kriteria STSE dengan nilai K_p, K_i, dan K_d paling tinggi. Jika dilihat dari

perubahan K_i ditunjukan pada Gambar 4.14, Sistem dengan kontroler *self-tuning* PID interaksi adaptif nilainya mampu beradaptasi sesuai dengan kondisi pada sistem tersebut. Adaptasi pertama dilakukan untuk menemukan nilai K_p yang cocok untuk sistem karena tidak diberikan inisialisasi nilai K_p. Proses selanjutnya sistem dengn kontroler *self-tuning* PID interaksi adaptif akan menyesuaikan dengan perubahan *set point* ataupun perubahan parameter pada plant yang diakibatkan oleh pembebanan. Perubahan yang terjadi untuk setiap fenomena tidak begitu besar, tetapi proses adaptasi dilakukan terus menerus sehingga perubahan yang terjadi setiap waktu seperti pada sinyal *ramp*, mampu diatasi dengan baik. Berbeda dengan kontroler PID yang hanya didesain untuk kondisi-kondisi tertentu.



Gambar 4.14 Perubahan nilai Ki ketika *tracking* sinyal *ramp*

Jika ditinjau dari perubahan nilai K_d yang ditunjukkan pada Gambar 4.15, nilai K_d paling besar ketika kontroler dirancang dengan kriteria STSE. Dengan kriteria tersebut kontroler mampu menghilangkan kesalahan yang terjadi dengan cepat.

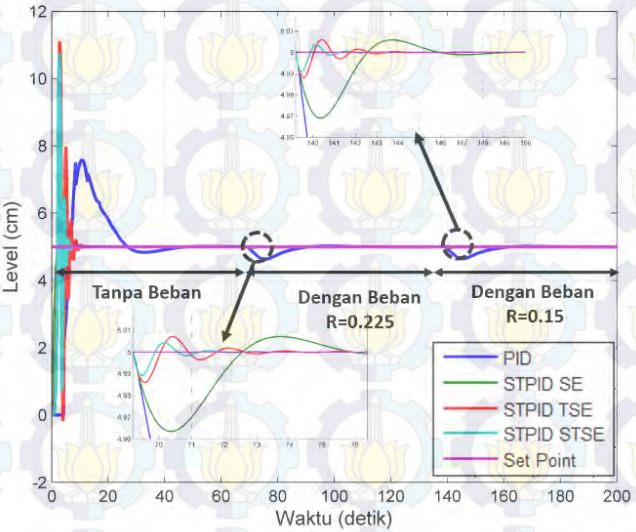


Gambar 4.15 Perubahan nilai Kd ketika *tracking* sinyal *ramp*

4.4.2 Pengujian Dengan Pembebanan

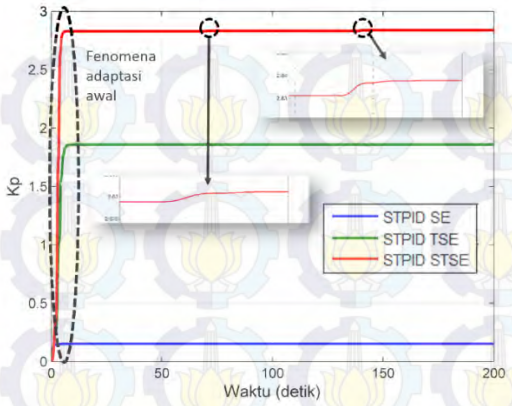
Pengujian dengan pembebanan ini bertujuan untuk membandingkan performa dari kontroler PID dengan kontroler *self-tuning* PID interaksi adaptif. Pengujian dilakukan dengan memberikan dua kali pembebanan kepada sistem dengan kontroler yang berbeda-beda. Hasil simualsi dapat dilihat pada Gambar 4.16. Dari Gambar 4.106 dapat dilihat pada saat detik ke 65 dan 135 sistem diberi gangguan berupa pembebanan dengan cara mengubah parameter *plant* yaitu R. Ketika Sistem diberikan beban tetapi dapat kembali ke kondisi semula berarti sistem dapat dikatakan stabil. Hal tersebut dikarenakan sistem mampu beradaptasi dengan penurunan nilai yang terjadi. Contoh penerapan nyata dari pembebanan pada system pengaturan level adalah adanya air yang dikeluarkan dari *drain valve*. Dalam hal ini bukaan dari *drain valve* merepresentasikan nilai R. Semakin besar *drain valve* dibuka maka air yang mengalir semakin cepat dan dianggap semakin besar pula nilai R. Kontroler PID mampu kembali pad kondisi keadaan tunak setelah pembebanan sebesar 45.5 detik pada pembebanan pertama dan 46,6 detik pada pembebanan kedua. Kontroler dengan kriteria STSE terbukti mampu lebih cepat dalam mengatasi beban yang diberikan dengan waktu 2 detik pada pembebanan pertama dan 2.4 detik pada pembebanan kedua. Sedangkan kriteria TSE sebesar 4.1 detik pada pembebanan pertama dan

4.1 detik pada pembebanan kedua. Serta kriteria SE sebesar 9.4 detik pada pembebanan pertama dan 9.2 detik pada pembebanan kedua. Tentu saja semakin kecepatan respon tersebut akan menimbulkan *overshoot* pada respon. Kriteria STSE lebih cepat namun terdapat *overshoot* sedangkan kriteria SE lebih lambat tapi tanpa ada *overshoot*.



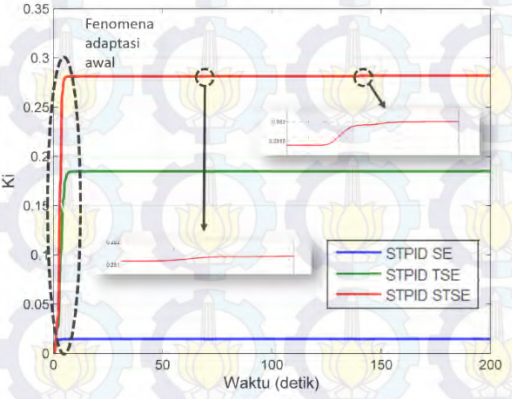
Gambar 4.16 Perbandingan sistem ketika pembebanan

Dari hasil pengujian ini telah terbukti bahwa kontroler *self-tuning* PID interaksi adaptif memiliki karakteristik *disturbance rejection* atau mengatasi gangguan. Seperti apa sistem dalam mengatasi pembebanan bisa didesain melalui ketiga kriteria kesalahan yaitu SE, TSE, dan STSE. Jika diinginkan penanganan yang cepat maka dipilih STSE. Jika diinginkan tanpa adanya *overshoot* maka dipilih kriteria SE. Aksi *disturbance rejection* itu diakibatkan oleh peranan adaptasi parameter kontroler yang ditunjukkan pada Gambar 4.17 untuk adaptasi nilai K_p, Gambar 4.18 untuk adaptasi nilai K_i, dan Gambar 4.19 untuk adaptasi nilai K_d.



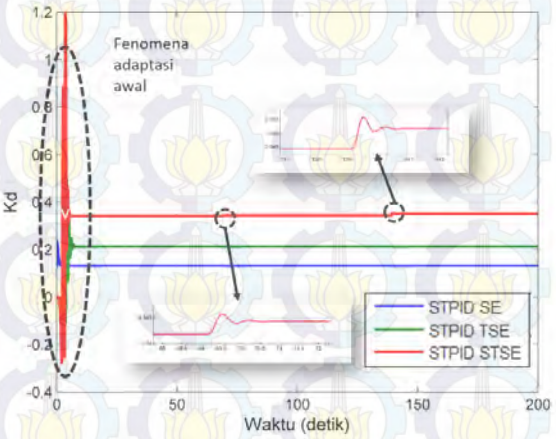
Gambar 4.17 Perubahan nilai K_p saat pembebanan

Dari Gambar 4.18 sama seperti hasil sebelumnya kontroler *self-tuning* PID interaksi adaptif melakukan adaptasi pada 5 detik pertama untuk menemukan parameter yang sesuai dengan kondisi sistem. Begitu juga ketika kita lihat dari perubahan nilai K_i pada Gambar 4.18.



Gambar 4.18 Perubahan nilai K_i ketika pembebanan

Pada nilai K_d proses adaptasi yang dilakukan lebih terlihat dibandingkan parameter kontrol yang lain. Kontroler differensial mampu mengurangi terjadinya osilasi akibat tingginya nilai K_d. Selain pada Gambar 4.19 nilai K_d terbesar dimiliki oleh kontroler dengan kriteria STSE yang berarti bahwa respon sistem dengan kontroler tersebut menjadi lebih cepat dari sebelumnya. Sama seperti pada parameter K_p, perubahan yang terjadi ketika pembebanan yaitu detik ke 65 dan 135 cenderung kecil. Hal tersebut tidak berarti bahwa kontroler *self-tuning* PID interaksi adaptif tidak mampu beradaptasi. Sebagai bukti pada Gambar 4.16 sistem dengan kontroler *self-tuning* PID interaksi adaptif mampu mengatasi penurunan nilai akibat pembebanan dengan cepat jika dibandingkan dengan kontroler PID. SE memiliki karakteristik lambat namun tanpa *overshoot*, sedangkan TSE dan STSE memiliki karakteristik yang cepat namun dengan *overshoot*.



Gambar 4.19 Perubahan nilai K_d ketika pembebanan

4.5 Implementasi Kontroler PID dan *Self-Tuning* PID interaksi adaptif pada PCT-100

Implementasi dilakukan dengan tiga buah pengujian. Pada implementasi didapatkan hasil yang berbeda dengan simulasi diantaranya

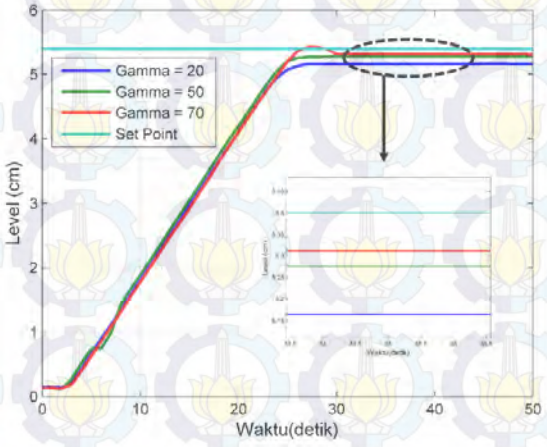
nilai K_p, K_i, dan K_d yang beradaptasi hingga mencapai ribuan nilainya.

Ketiga pengujian yang dilakukan antara lain.

- Pengujian dengan variasi nilai gamma
- Pengujian dengan perubahan *set point*
- Pengjian dengan pembebanan

4.5.1 Pengujian Variasi Nilai Gamma

Pengujian ini bertujuan untuk melihat pengaruh nilai gamma yang dipilih terhadap sistem dengan kontroler *self-tuning PID* interaksi adaptif dalam kemampuannya menghilangkan kesalahan keadaan tunak. Hasil implementasi dapat dilihat pada Gambar 4.20, dari gambar itu didapatkan hasil semakin besar nilai gamma semakin sistem tersebut mampu mengurangi kesalahan keadaan tunak. Tetapi diperoleh hasil yang tidak sesuai akibat batasan sinyal kontrol yang diberikan pada pompa. Sinyal kontrol yang terbaca pada program sebesar 300 V, tetapi karena adanya batasan maka sinyal kontrol yang dikeluarkan oleh kontroler hanya 10 V. Akibat adanya pembatasan nilai tersebut mengakibatkan adanya kesalahan keadaan tunak. Jika pada *real* sistem rentang tegangan pompa yang digunakan akan lebih besar sehingga sinyal kontrol yang diberikan bisa lebih besar.

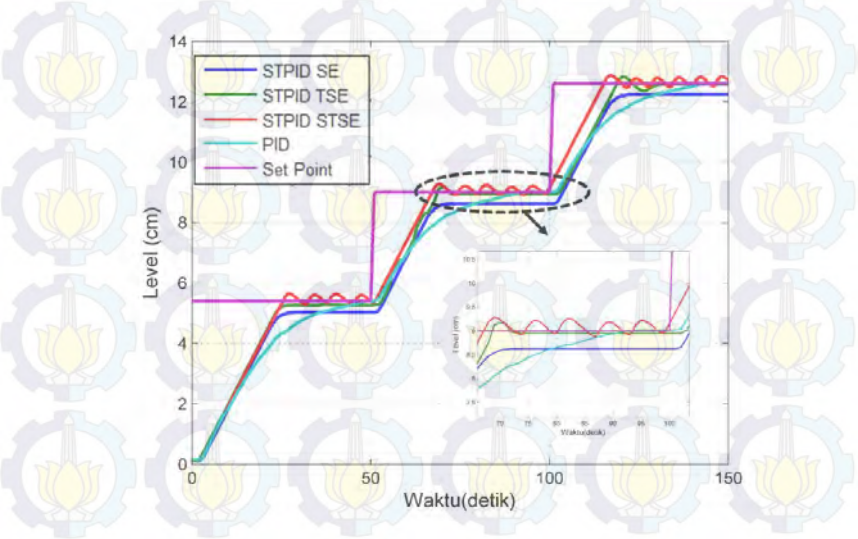


Gambar 4.20 Respon sistem dengan variasi nilai gamma

4.5.2 Pengujian Dengan Perubahan *Set Point*

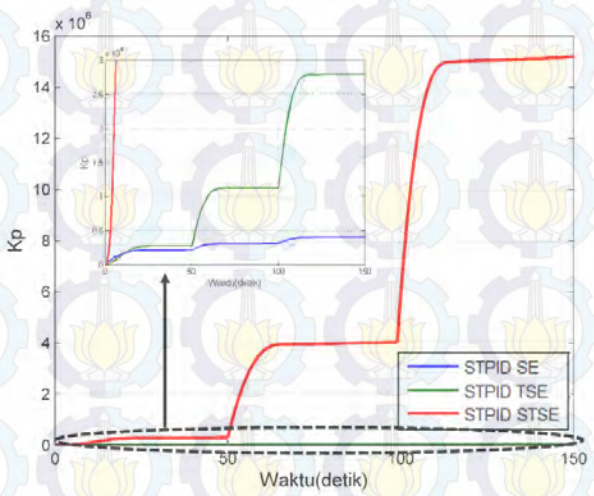
Pengujian dengan perubahan *set point* dilakukan bertujuan untuk membandingkan antara kontroler PID dengan *self-tuning* PID interaksi adaptif ketika digunakan untuk mengikuti *set point*. Gambar 4.21 menunjukkan hasil implementasi yang dilakukan terhadap PCT-100, terlihat bahwa masih terdapat kesalahan keadaan tunak untuk kontroler *self-tuning* PID dengan kriteria SE. Kesalahan keadaan tunak tersebut diakibatkan oleh pembatasan sinyal kontrol yang dihasilkan oleh kontroler, seharusnya pada detik tersebut dengan sinyal kontrol tanpa dibatasi sudah bisa mencapai nilai *set point*. Batasan yang diberikan adalah 0 – 10 V yang berarti jika terdapat sinyal kontrol dibawah nilai 0 maka akan tetap bernilai 0. Sedangkan jika sinyal kontrol bernilai diatas 10 akan tetap bernilai 10.

Dalam implementasi hanya diuji dengan menaikkan nilai set point tanpa menguji dengan penurunan set point. Pengujian tersebut dilakukan karena karakteristik antara *drain valve* dan pompa berbeda. Pompa dapat bekerja antara rentang 0 – 10 V sedangkan *drain valve* hanya bisa bekerja secara nyala-mati. Ketika diberikan tegangan 5 V maka akan terdapat bunyi seperti alarm pengingat.



Gambar 4.21 Perbandingan implementasi dengan perubahan *set point*

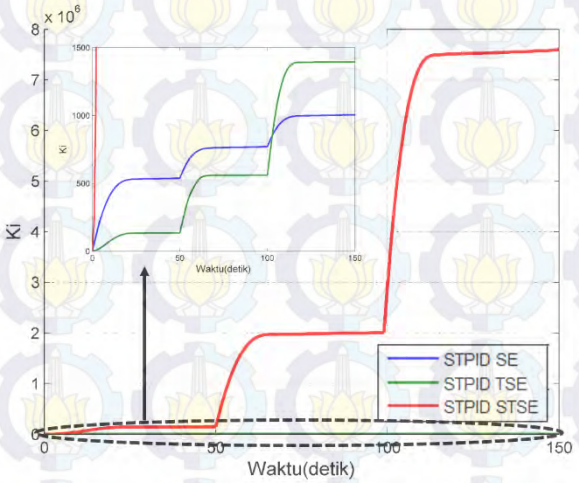
Sistem dengan kontroler *self-tuning* PID kriteria STSE nilainya berosilasi dikarenakan nilai gamma yang diberikan terlalu besar sedangkan karakteristik *drain valve* yang ada tidak bisa digunakan untuk nilai analog. Ketika dianalisa dengan nilai RMSE didapatkan bahwa kontroler PID lebih baik diantara kontroler lainnya. Kontroler *self-tuning* PID dengan kriteria SE mempunyai nilai RMSE sebesar 37.7 % ketika *set point* 5.4, 37.63 % ketika *set point* 9, dan 37.26 % ketika *set point* 12.6. Sedangkan kriteria TSE mempunyai nilai RMSE sebesar 13.06 % ketika *set point* 5.4, 5.01 % ketika *set point* 9, dan 8.96 ketika *set point* 12.6. kriteria STSE mempunyai nilai RMSE sebesar 78.89 % ketika *set point* 5.4, 61.37 % ketika *set point* 9, dan 13.68 ketika *set point* 12.6. Kontroler PID mempunyai nilai RMSE sebesar 4.5 % ketika *set point* 5.4, 1.66 % ketika *set point* 9, dan 0.88 ketika *set point* 12.6. Dalam hal kecepatan respon kontroler PID masih terlalu tertinggal dengan kontroler *self-tuning* PID interaksi adaptif. Tentunya kecepatan kontroler *self-tuning* PID interaksi adaptif mampu menghasilkan respon yang cepat dikarenakan adaptasi nilai K_p, K_i, dan K_d yang ditunjukkan pada Gambar 4.22 menunjukkan perubahan nilai K_p ketika terjadi perubahan *set point*.



Gambar 4.22 Perubahan nilai K_p ketika terjadi perubahan *set point*

Nilai K_p yang didapatkan bernilai sangat besar dikarenakan pada krieria TSE dan STSE dikalikan bobot pengali waktu, jika waktu semakin besar maka nilai K_p yang didapatkan akan semakin besar pula. Besarnya nilai K_p itulah yang membuat kontroler *self-tuning* PID dengan kriteria STSE mampu lebih cepat menuju *set point* namun menghasilkan *overshoot* diakarenakan sinyal kontrol yang diberikan terlalu besar.

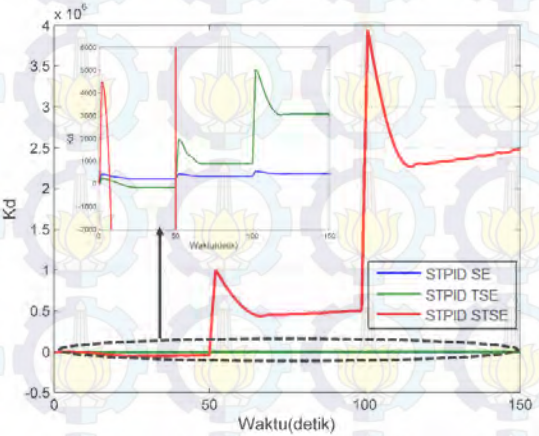
Hal yang sama juga terjadi pada nilai K_i seperti Gambar 4.23. Nilai K_i adalah penyebab hilangnya kesalahan keadaan tunak. Kontroler *self-tuning* PID dengan kriteria STSE masih memiliki nilai yang paling besar yang diakibatkan oleh bobot pengali waktu. Besarnya nilai gamma juga berpengaruh terhadap niali K_i yang diadaptasi. Karena batasan yang diberikan terhadap pompa menyebabkan respon sistem dengan kontroler *self-tuning* PID interaksi adaptif serupa semua. Jika tidak adanya batasan maka sudah pasti kriteria STSE yang paling cepat menuju *set point*.



Gambar 4.23 Perubahan nilai K_i ketika perubahan *set point*

Selain nilai K_i, nilai K_d juga terjadi hal yang sama pada Gambar 4.24. Setiap kali ada perubahan *set point* maka aktuator akan langsung

bekerja. Ketika *set point* dinaikkan maka pompa akan langsung bekerja. Ketika fase *starting* dan fase *shutdown* dari pompa terdapat bunyi perubahan kontak dari *relay* secara terus menerus dari pompa. Perubahan yang terus menerus pada *relay* akan mengakibatkan pompa seakan-akan bekerja nyala-mati. Dalam *real* sistem ketika terjadi hal itu maka dapat menimbulkan *bouncing* yang bisa memicu terjadinya percikan api. Selain menurunkan kinerja dari pompa, fenomena *bouncing* dapat membuat *lifetime* dari peralatan menjadi berkurang. Fenomena tersebut masih bisa diatasi jika pemberian parameter kontroler sesuai.

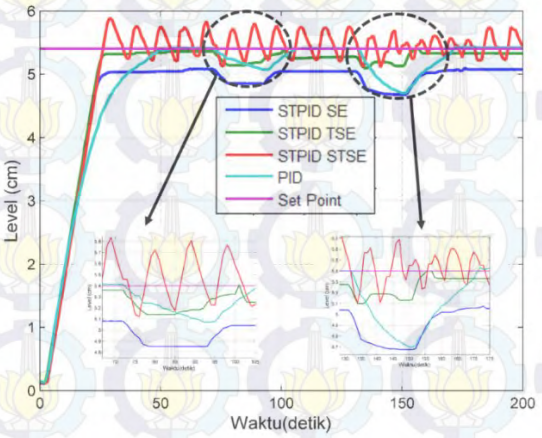


Gambar 4.24 Perubahan nilai Kd ketika perubahan *set point*

4.5.3 Pengujian Dengan Pembebanan

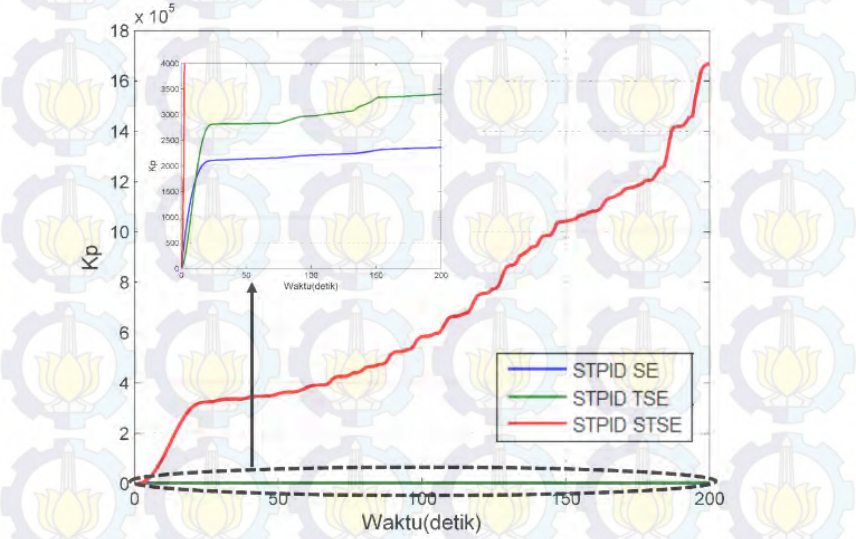
Pengujian ini dilakukan dengan tujuan untuk mempertahankan nilai keluaran sistem pada level. Pengujian dilakukan dengan memberikan dua kali pembebanan. Pembebanan pertama dilakukan pada detik ke 70 dan detik ke 130 diberikan lagi beban. Hasil implementasi dapat dilihat pada Gambar 4.25. Dari Gambar 4.25 didpatkan nilai RMSE untuk sistem dengan kontroler PID sebesar 25,23 %. Kontroler *self-tuning* PID interaksi adaptif dengan kriteria STSE sebesar 20,26 %, kriteria TSE sebesar 13,74 %, dan kriteria SE sebesar 43,25 %. Jika ditinjau dari nilai RMSE kontroler *self-tuning* PID interaksi adaptif dengan kriteria TSE lebih baik dan cocok dengan nilai gamma 50. Penurunan nilai yang terjadi

juga tidak terlalu besar jika dibandingkan dengan kontroler PID. Selain itu analisa dilakukan terhadap *recovery time* dari masing-masing kontroler ketika terjadi pembebanan. Kontroler PID menghasilkan *recovery time* sebesar 35 detik pada pembebanan pertama dan 46 detik pada pembebanan kedua. Kontroler *self-tuning* PID interaksi adaptif kriteria STSE menghasilkan *recovery time* sebesar 27 detik ketika pembebanan pertama dan 29 detik ketika pembebanan kedua, kriteria TSE menghasilkan nilai 31 detik ketika pembebanan pertama dan 27 detik ketika pembebanan kedua, sedangkan kriteria SE menghasilkan 27 detik pada pembebanan pertama dan 29 detik pada pembebanan kedua. Sistem dengan kontroler *self-tuning* PID interaksi adaptif dengan semua kriteria masih lebih cepat jika dibandingkan dengan kontroler PID. Pada kriteria STSE tidak cocok digunakan nilai gamma 50 dikarenakan respon yang dihasilkan berosilalasi yang menyebabkan pompa bekerja nyala-mati. Kerja pompa yang dipaksa dalam keadaan nyala-mati dapat menyebabkan *lifetime* dari pompa akan berkurang dan juga dapat memicu timbulnya percikan api. Percikan api berasal dari fenomena *bouncing* yang dipicu oleh kerja *relay* atau kontaktor yang terus menerus berubah dari kondisi nyala ke mati atau nyala-mati . Perubahan nilai secara terus menerus akibat tidak sesuainya parameter kontroler yang diberikan sehingga sinyal kontrol yang diberikan berosilasi.



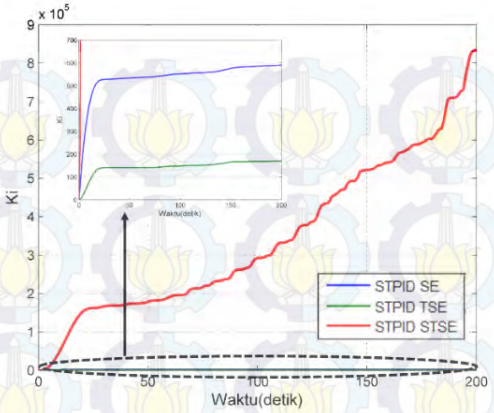
Gambar 4.25 Perbandingan implementasi kontroler dengan pembebanan

Fenomena lain juga muncul ketika implementasi yaitu pembacaan sensor yang tidak dimulai dari 0. Fenomena itu terjadi akibat sensor belum dikalibrasi dengan benar. Dalam dunia industri kesalahan pembacaan sensor menjadi masalah yang sangat serius karena dapat berdampak pada sinyal kontrol yang diberikan. Kemampuan *disturbance rejection* dari kontroler self tuning PID interaksi adaptif tidak lepas dari peran adaptasi nilai K_p, K_i, dan K_d yang ditunjukkan pada Gambar 4.26 merupakan adaptasi nilai K_d ketika terjadi pembebanan.



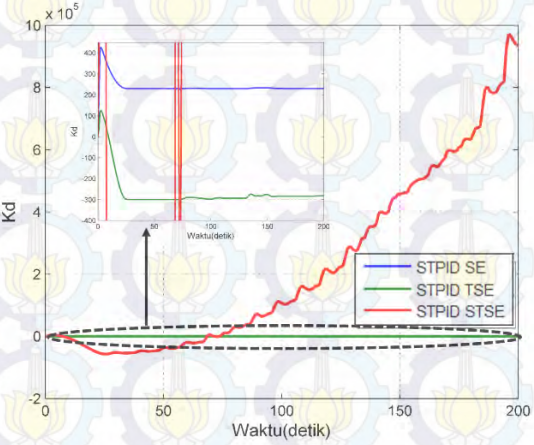
Gambar 4.26 Adaptasi nilai K_p ketika pembebanan

Nilai K_p yang dihasilkan relatif besar pada kontroler *self-tuning* PID dengan kriteria STSE. Hal yang sama juga terjadi pada nilai K_i seperti Gambar 4.27. Nilai K_i adalah penyebab hilangnya kesalahan keadaan tunak. Kontroler *self-tuning* PID dengan kriteria STSE masih memiliki nilai yang paling besar yang diakibatkan oleh bobot pengali waktu.



Gambar 4.27 Adaptasi nilai K_i ketika pembebanan

Selain itu adaptasi juga dilakukan pada nilai K_d. K_d bernilai negatif diakrenakan nilai kesalahan sebelumnya lebih besar daripada kesalahan sekarang. Adaptasi nilai K_d dapat diliat pada Gambar 4.28.



Gambar 4.28 Perubahan nila K_d ketika pembebanan

Dari semua pengujian, masing-masing parameter untuk setiap pengujian seperti RMSE dan *recovery time* ditunjukkan pada Tabel 4.1.

Tabel 4. 1 Data hasil pengujian

Kontroler	SIMULASI				IMPLEMENTASI			
	Perubahan *set point*	Pembebanan			Perubahan *set point*	Pembebanan		
	RMSE (%)	RMSE (%)	*Settling time* (s)	Penurunan nilai (%)	RMSE (%)	RMSE (%)	*Settling time* (s)	Penurunan nilai (%)
PID	18,78	13,88	46,05	7,2	2,34	25,23	40,5	9,48
STPID SE	5,83	0,623	9,3	0,69	37,53	43,25	28	4,71
STPID TSE	4,57	0,143	4,1	0,28	9,01	13,74	29	5,18
STPID STSE	3,64	0,093	2,2	0,03	51,31	20,26	29,5	5,27

BAB 5
PENUTUP

5.1 Kesimpulan

- Kecepatan adaptasi dari kontroler STPID interaksi adaptif dipengaruhi oleh besarnya nilai gamma
- Pada hasil simulasi, kontroler STPID kriteria STSE lebih unggul dalam hal perubahan *set point* dengan RMSE sebesar 3,64 %.
- Ketika disimulasikan, kontroler STPID kriteria STSE lebih unggul dalam hal pembebanan dengan RMSE, *recovery time,* dan penurunan nilai sebesar 0,093 %, 2,2 detik, dan 0,03 %.
- Pada implementasi, kontroler PID lebih unggul dalam hal perubahan *set point* dengan RMSE sebesar 2,34%.
- Ketika diimplementasikan, kontroler STPID kriteria TSE lebih unggul dalam hal pembebanan dengan RMSE, *recovery time*, dan penurunan nilai sebesar 13,74 %, 29 detik, dan 5,18 %.

5.2 Saran

Pada penelitian selanjutnya kontroler self tuning PID interaksi adaptif diharapkan

- Diimplementasikan menggunakan sistem yang lebih bervariasi dalam hal pembebanan seperti *real system* pada dunia industri.
- Diterapkan pada sistem nonlinear seperti pendulum . Bisa diterapkan sendiri ataupun digabungkan dengan *artificial intelligent* sehingga tidak memerlukan waktu yang lama untuk melakukan proses pembelajaran.
- Masih diperlukan optimalisasi cara penentuan nilai gamma secara otomatis yang digunakan agar sistem selalu dalam performa terbaiknya.

(Halaman ini sengaja dikosongkan)

LAMPIRAN

Penurunan rumus

$$\frac{\partial E}{\partial \alpha_c} = \frac{\partial E}{\partial y_{postc}} \circ \frac{\partial y_{postc}}{\partial x_{postc}} \circ \frac{\partial x_{postc}}{\partial \alpha_c} \tag{6.1}$$

$$\frac{\partial E}{\partial \alpha_c} = \frac{\partial E}{\partial y_{postc}} \circ F'_{postc}[x_{postc}] \circ y_{preC} \tag{6.2}$$

$$\dot{\alpha}_c = -\gamma \frac{\partial E}{\partial y_{postc}} \circ F'_{postc}[x_{postc}] \circ y_{prec}, \tag{6.3}$$

Hal tersebut menunjukkan persamaan ini mempunyai nilasi solusi yang unik oleh karena itu nilai E akan turun secara monoton berdasarkan waktu. Kemudian nilai $O_{postc} = O_4 = \text{himpunan kosong}$, dikarenakan tidak adanya interaksi antar *output* .sehingga didapatkan Persamaan 6.3.

$$\dot{\alpha}_c = \left(0 - \gamma \frac{\partial E}{\partial y_{postc}} \right) \circ F'_{postc}[x_{postc}] \circ y_{prec}, \tag{6.4}$$

$$\dot{\alpha}_c = -\gamma \frac{\partial E}{\partial y_{postc}} \circ F'_{postc}[x_{postc}] \circ y_{prec}, \tag{6.5}$$

Persamaan 6.6 merupakan bentuk lebih luas dari Persamaan 6.5 untuk persamaan semua anggota N,

$$\frac{dE}{dy_n} = \frac{\partial E}{\partial y_n} + \sum_{C \in O_n} \frac{dE}{dy_{postc}} \circ \frac{dy_{postc}}{dy_n} \tag{6.6}$$

$$\frac{dE}{dy_n} = \frac{\partial E}{\partial y_n} + \sum_{C \in O_n} \alpha_c \frac{dE}{dy_{postc}} \circ F'_{postc}[x_{postc}] \tag{6.7}$$

$$\frac{dE}{dy_n} = \frac{\partial E}{\partial y_n} + \sum_{C \in O_n} \alpha_c \frac{dE}{d\alpha_c} \frac{\frac{dE}{dy_{postc}} \circ F'_{postc}[x_{postc}]}{\frac{dE}{dy_{postc}} \circ F'_{postc}[x_{postc}] \circ y_{pre_c}} \tag{6.8}$$

dengan substitusi Persamaan 2.38 ke Persamaan 6.8.

$$\frac{dE}{dy_n} = \frac{\partial E}{\partial y_n} + \sum_{C \in O_n} \alpha_c \left(-\frac{\dot{\alpha}_c}{\gamma} \right) \frac{\frac{dE}{dy_{postc}} \circ F'_{postc}[x_{postc}]}{\frac{dE}{dy_{postc}} \circ F'_{postc}[x_{postc}] \circ y_{pre_c}} \tag{6.9}$$

$$\frac{dE}{dy_n} = \frac{\partial E}{\partial y_n} - \frac{1}{\gamma} \sum_{C \in O_n} \alpha_c \dot{\alpha}_c \frac{\frac{dE}{dy_{postc}} \circ F'_{postc}[x_{postc}]}{\frac{dE}{dy_{postc}} \circ F'_{postc}[x_{postc}] \circ y_{pre_c}} \tag{6.10}$$

$$\dot{\alpha}_c = -\gamma \frac{\partial E}{\partial y_{postc}} \circ F'_{postc}[x_{postc}] \circ y_{preC} \tag{6.11}$$

$$\dot{\alpha}_c = -\gamma \left(\frac{\partial E}{\partial y_{postc}} - \frac{1}{\gamma} \sum_{C \in O_n} \alpha_c \dot{\alpha}_c A \right) \circ F'_{postc}[x_{postc}] \circ y_{preC} \tag{6.12}$$

$$\dot{\alpha}_c = \left(\sum_{C \in O_n} \alpha_c \dot{\alpha}_c A - \gamma \frac{\partial E}{\partial y_{postc}} \right) \circ F'_{postc}[x_{postc}] \circ y_{preC} \tag{6.13}$$

Ketika $y_{postc} = y_{prec}$ maka, Persamaan 6.13 dapat disederhanakan menjadi Persamaan 6.14.

$$\dot{\alpha}_c = \left(\sum_{C \in O_n} \alpha_c \dot{\alpha}_c A - \gamma \frac{\partial E}{\partial y_{postc}} \right) \circ F'_{postc}[x_{postc}] \circ y_{preC} \tag{6.14}$$

Listing Program Simulasi

PID Tanpa Gangguan

```
%.....Created by.....%
%.....Muhammad Zakki Ghufron.....%
%.....2212100064.....%
%.....Electrical Engineering Department.....%
%.....Faculty of Industrial Technology.....%
%.....Institut Teknologi Sepuluh Nopember.....%
clear all
clc

%% Inisialisasi waktu dan variasi parameter kontroler
t=0:0.1:200;              %Waktu simulasi dan banyaknya data yang
diambil
%Variasi nilai gain proporsional kontroler
Kp3= 6;
Kp4= 8;
Kp5= 9;
Kp6= 12;
%Variasi nilai waktu integral kontroler
ti3= 5;
Ki3= 14.17/5;
ti4= 15;
Ki4=14.17/15;
ti5= 22;
Ki5=14.17/22;
ti6= 30;
Ki6=14.17/30;
%Variasi nilai waktu differensial kontroler
td3= 0.1;
Kd3=14.17*0.1;
td4= 0.5;
Kd4=14.17*0.5;
td5= 1;
Kd5=14.17*1;
td6= 1.5;
Kd6=14.17*1.5;
```

```
%% Inisialisasi masukan
%Masukan=sin(2*pi*0.02*t);  %Masukan sinyal sinus dengan frekuensi
0.02 Hz
t1=0:0.01:6.5;            %Masukan sinyal ramp antara 0 - 3.5
t2=6.5:0.01:13.5;          %Masukan sinyal ramp antara 3.5 - 6.5
Masukan(1,1:651) =t1;      %Masukan sinyal ramp
Masukan(1,651:1301)=6.5;    %Masukan sinyal ramp
Masukan(1,1301:2001)=t2;    %Masukan sinyal ramp
Masukan1(1,1:1001) =5;      %Masukan sinyal uji perubahan set point
Masukan1(1,1001:2001)=10;    %Masukan sinyal uji perubahan set
point
Masukan2(1,1:2001)=5;      %Masukan sinyal uji unit step

%% Penyusunan fungsi alih masing-masing variasi
%Fungsi alih sistem dengan Kp =9.729 dan Ti =10.37
P=tf([1.753],[109.92 1],'inputdelay',4.23);       %fungsi alih plant
C8=pid(14.17,1.4,25.08,0);            %fungsi alih kontroler PI
B=tf(C8)
T = feedback(B*P,1);                 %fungsi alih loop tertutup

%Fungsi alih sistem dengan Variasi nilai KP
ti =10.152;
td =1.77

%Dengan KP=6
C3=pid(Kp3,1.4,25.08,0);              %fungsi alih kontroler PI
B3=tf(C3);
T3 = feedback(B3*P,1);               %fungsi alih loop tertutup

%Dengan KP=8
C4=pid(Kp4,1.4,25.08,0);              %fungsi alih kontroler PI
B4=tf(C4);
T4 = feedback(B4*P,1);               %fungsi alih loop tertutup

%Dengan KP=9
C5=pid(Kp5,1.4,25.08,0);              %fungsi alih kontroler PI
B5=tf(C5);
T5 = feedback(B5*P,1);               %fungsi alih loop tertutup
```

```
%Dengan KP=12
C6=pid(Kp6,1.4,25.08,0);                %fungsi alih kontroler PI
B6=tf(C6);
T6 = feedback(B6*P,1);                  %fungsi alih loop tertutup

%Fungsi alih sistem dengan Variasi nilai Ti
Kp= 14.17;
td= 1.77;

%Dengan Ti=10
C7=pid(Kp,Ki3,25.08,0);                 %fungsi alih kontroler PI
B7=tf(C7);
T7 = feedback(B7*P,1);                  %fungsi alih loop tertutup

%Dengan Ti=15
C8=pid(Kp,Ki4,25.08,0);                 %fungsi alih kontroler PI
B8=tf(C8);
T8 = feedback(B8*P,1);                  %fungsi alih loop tertutup

%Dengan Ti=19
C9=pid(Kp,Ki5,25.08,0);                 %fungsi alih kontroler PI
B9=tf(C9);
T9 = feedback(B9*P,1);                  %fungsi alih loop tertutup

%Dengan Ti=22
C10=pid(Kp,Ki6,25.08,0);                %fungsi alih kontroler PI
B10=tf(C10);
T10 = feedback(B10*P,1);                %fungsi alih loop tertutup

%Fungsi alih sistem dengan Variasi nilai Td
Kp= 14.17;
ti= 10.152;

%Dengan Td=10
C11=pid(Kp,1.4,Kd3,0);                  %fungsi alih kontroler PI
B11=tf(C11);
T11 = feedback(B11*P,1);                %fungsi alih loop tertutup
```

```
%Dengan Ti=15
C12=pid(Kp,1.4,Kd4,0);                  %fungsi alih kontroler PI
B12=tf(C12);
T12 = feedback(B12*P,1);                %fungsi alih loop tertutup

%Dengan Ti=19
C13=pid(Kp,1.4,Kd5,0);                  %fungsi alih kontroler PI
B13=tf(C13);
T13 = feedback(B13*P,1);                %fungsi alih loop tertutup

%Dengan Ti=22
C14=pid(Kp,1.4,Kd6,0);                  %fungsi alih kontroler PI
B14=tf(C14);
T14 = feedback(B14*P,1);                %fungsi alih loop tertutup
%% Simulasi
D1 = lsim(T,Masukan,t);    %Pengujian dengan perubahan set point
D2 = lsim(T,Masukan1,t);   %Pengujian dengan sinyal ramp
D3 = lsim(T3,Masukan2,t);  %Variasi Kp (Kp3= 6) Ti =10.152 td = 1.77
D4 = lsim(T4,Masukan2,t);  %Variasi Kp (Kp4= 8) Ti =10.152 td = 1.77
D5 = lsim(T5,Masukan2,t);  %Variasi Kp (Kp5= 9) Ti =10.152 td = 1.77
D6 = lsim(T6,Masukan2,t);  %Variasi Kp (Kp6= 12) Ti =10.152 td =
1.77
D7 = lsim(T7,Masukan2,t);  %Variasi Ti (ti3= 10) Kp =14.17 td = 1.77
D8 = lsim(T8,Masukan2,t);  %Variasi Ti (ti4= 15) Kp =14.17 td = 1.77
D9 = lsim(T9,Masukan2,t);  %Variasi Ti (ti5= 19) Kp =14.17 td = 1.77
D10 = lsim(T10,Masukan2,t);%Variasi Ti (ti6= 22) Kp =14.17 td = 1.77
D11 = lsim(T11,Masukan2,t);%Variasi Td (td3= 0.5) Kp =14.17 ti =
10.152
D12 = lsim(T12,Masukan2,t);%Variasi Td (td4= 3) Kp =14.17 ti =
10.152
D13 = lsim(T13,Masukan2,t);%Variasi Td (td5= 5) Kp =14.17 ti =
10.152
D14 = lsim(T14,Masukan2,t);%Variasi Td (td6= 10) Kp =14.17 ti =
10.152

%% Plotting Hasil
figure (1)
plot(t,D1,t,Masukan,'linewidth',2),grid on
```

```
title('Pengujian dengan Set Point sinyal
ramp','FontWeight','Bold','FontSize',14)
xlabel('Waktu(detik)','FontWeight','normal','FontSize',12)
ylabel('Level (cm)','FontWeight','normal','FontSize',12)
legend('Respon sistem','Set Point')

figure (2)
plot(t,D3(:,1),t,D4(:,1),t,D5(:,1),t,D6(:,1),t,Masukan2(1,:),'linewidth',2),
grid on
title('Pengujian dengan variasi KP','FontWeight','Bold','FontSize',14)
xlabel('Waktu(detik)','FontWeight','normal','FontSize',12)
ylabel('Level (cm)','FontWeight','normal','FontSize',12)
legend('KP=6','KP=8','KP=9','KP=12','Set Point')

figure (3)
plot(t,D7(:,1),t,D8(:,1),t,D9(:,1),t,D10(:,1),t,Masukan2(1,:),'linewidth',2)
,grid on
title('Pengujian dengan variasi Ti','FontWeight','Bold','FontSize',14)
xlabel('Waktu(detik)','FontWeight','normal','FontSize',12)
ylabel('Level(cm)','FontWeight','normal','FontSize',12)
legend('Ti=10','Ti=15','Ti=22','Ti=30','Set Point')

figure (4)
plot(t,D2,t,Masukan1,'linewidth',2),grid on
title('Pengujian dengan Perubahan Set
Point','FontWeight','Bold','FontSize',14)
xlabel('Waktu(detik)','FontWeight','normal','FontSize',12)
ylabel('Level (cm)','FontWeight','normal','FontSize',12)
legend('Respon sistem','Set Point')

figure (5)
plot(t,D11(:,1),t,D12(:,1),t,D13(:,1),t,D14(:,1),t,Masukan2(1,:),'linewidt
h',2),grid on
title('Pengujian dengan variasi Td','FontWeight','Bold','FontSize',14)
xlabel('Waktu(detik)','FontWeight','normal','FontSize',12)
ylabel('Level(cm)','FontWeight','normal','FontSize',12)
legend('Td=0.1','Td=0.5','Td=1','Td=1.5','Set Point')
```

PID Gangguan

```
%.....................................Created by.....................................%
%.................................Muhammad Zakki Ghufron.....................................%
%.....................................2212100064.....................................%
%.....................Electrical Engineering Department.....................%
%.....................Faculty of Industrial Technology.....................%
%.....................Institut Teknologi Sepuluh Nopember.....................%

clear all
clc

%% Inisialisasi waktu
t=0:0.1:200;      %Waktu Simulasi dan banyaknya data yang diambil
t1=0:0.1:65;      %Waktu simulasi tanpa beban
t2=65:0.1:135;     %Waktu simulasi pemberian beban 1
t3=135:0.1:200;    %Waktu simulasi pemberian beban 2

%% Inisialisasi Masukan
Masukan(1,1:2001)=5;        %Masukan berupa sinyal unit step
Masukan1(1,1:701)=5;        %Masukan saat pembebanan 1
Masukan2(1,1:651)=5;        %Masukan saat pembebanan 2

%% Penyusunan fungsi alih sistem
%tanpa beban
P=tf([1.753],[109.92 1],'inputdelay',4.23);     %fungsi alih plant
C=pid(14.17,1.4,25.08,0); %fungsi alih kontroler PI
B=tf(C);
T = feedback(B*P,1);              %fungsi alih loop tertutup

%Dengan beban=0.225
P1=tf([0.7214],[45.216 1],'inputdelay',4.23); %fungsi alih plant dengan
Beban=0.225
T1 = feedback(P1,B) ;             %fungsi alih loop tertutup

%Dengan beban=0.15
P2=tf([0.48],[30.144 1],'inputdelay',4.23);   %fungsi alih plant dengan
Beban=0.15
T2 = feedback(P2,B) ;             %fungsi alih loop tertutup
```

```
%Pemberian noise
n2a=awgn(Masukan,20,'Measured');n2a=n2a-Masukan;
n2b=awgn(Masukan,30,'Measured');n2b=n2b-Masukan;
n2c=awgn(Masukan,40,'Measured');n2c=n2c-Masukan;
T3 = feedback(1,B*P)  ;

%% Simulasi
%Pembebanan
for k = 1:3;
   for k=1;
      B11(1:651,1) = (5*step(T,t1));
   end
   for k=2;
      B11(651:1351,1) = (5*step(T,t2))-lsim(T1,Masukan1,t2);
   end
   for k=3
      B11(1351:2001,1) = (5*step(T,t3))-lsim(T2,Masukan2,t3);
   end
end

%Pemberian noise
B21(:,1) = (5*step(T,t))+lsim(T3,n2a,t);
B22(:,1) = (5*step(T,t))+lsim(T3,n2b,t);
B23(:,1) = (5*step(T,t))+lsim(T3,n2c,t);

%% Plotting Hasil
figure (1)
plot(t,B11,t,Masukan,'linewidth',2.5),grid on
title('Respon Sistem dengan
Pembebanan','FontWeight','Bold','FontSize',14)
ylabel('Level (cm)','FontWeight','normal','FontSize',12)
xlabel('Waktu(detik)','FontWeight','normal','FontSize',12)
legend('Respon sistem','Setpoint')

figure (2)
plot(t,B21,t,B22,t,B23,t,Masukan,'linewidth',2),grid on
title('Respon Sistem dengan Pemberian
noise','FontWeight','Bold','FontSize',14)
ylabel('Level (cm)','FontWeight','normal','FontSize',12)
```

```
xlabel('Waktu(detik)','FontWeight','normal','FontSize',12)
legend('SNR = 20','SNR = 30','SNR = 40','Setpoint')
```

Self Tuning PID SE Tanpa Gangguan

```
%.....Created by.....%
%.....Muhammad Zakki Ghufron.....%
%.....2212100064.....%
%.....Electrical Engineering Department.....%
%.....Faculty of Industrial Technology.....%
%.....Institut Teknologi Sepuluh Nopember.....%
clear all;
clc;

%% Inisialisasi Waktu
t=0:0.1:200;      %Waktu Simulasi dan banyaknya data yang diambil

%% Inisialisasi Masukan

%Pengujian dengan bermacam-macam set point
%Masukan = input('Set Point='); %Masukkan Set Point bebas

%Pengujian dengan sinyal ramp
%t1=0:0.01:6.5;            %Masukan sinyal ramp antara 0 - 6.5
%t2=6.5:0.01:13.5;         %Masukan sinyal ramp antara 6.5 - 13.5
%Masukan(1,1:651) =t1;     %Masukan sinyal ramp
%Masukan(1,651:1301)=6.5;  %Masukan sinyal ramp
%Masukan(1,1301:2001)=t2;  %Masukan sinyal ramp

%Pengujian perubahan set point
Masukan(1,1:1001) =5;        %Masukan sinyal uji perubahan set point
Masukan(1,1001:2001)=6;      %Masukan sinyal uji perubahan set
point

%Pengujian dengan sinyal unit step
%Masukan(1,1:1201)=1;        %Masukan sinyal uji unit step

%% Inisialisasi variabel
dt = 0.1;                              %sampling time
n = 2000;                              %iterasi program
```

```
gamma =input('konstanta adaptive=');                    %Masukkan gamma
KP_dot(1:n+1) = 0;KI_dot(1:n+1) = 0;KD_dot(1:n+1) = 0;
KP_SUM(1:n+1) = 0;KI_SUM(1:n+1) = 0;KD_SUM(1:n+1) = 0;
Kesalahan(1:n+1)=0;U(1:n+1)=0;Yout(1:n+1)=0;Yout(1:n+1)=0;
P(1:n+1)=0;I(1:n+1)=0;D(1:n+1)=0;

%% Perhitungan nilai kesalahan dan output pertama
%Checking kesalahan
Kesalahan(1) = Masukan(1)-0;

%Kontroler PID
Y1(1) = Kesalahan(1);
Y3(1)  = (Kesalahan(1) - 0)/dt;
Y2(1)  = (Kesalahan(1) + 0)*dt/2;

%adaptive interaction
KP_dot(1) = Y1(1)*Kesalahan(1)*gamma;
KI_dot(1) = Y2(1)*Kesalahan(1)*gamma;
KD_dot(1) = Y3(1)*Kesalahan(1)*gamma;
KP(1) = (KP_dot(1)+0)*dt/2;
KP_SUM(1) = (sum(KP));          %Perubahan nilai KP
KI(1) = (KI_dot(1)+0)*dt/2;
KI_SUM(1) = (sum(KI));          %Perubahan nilai KI
KD(1) = (KD_dot(1)+0)*dt/2;
KD_SUM(1) = (sum(KD));          %Perubahan nilai KD

%Tuning PID
P(1) = KP_SUM(1)*Y1(1);
I(1) = KI_SUM(1)*Y2(1);
D(1) = KD_SUM(1)*Y3(1);
PID(1)=P(1)+I(1)+D(1);
U(1)=sum(PID);                  %Sinyal kontrol

%Modelling PCT-100
Yout(2)=-0.06746*U(2)+0.06905*U(1);

%% Simulasi
for i=2:n;
  %Checking kesalahan
```

```
    Kesalahan(i) = Masukan(i) - Yout(i);

    %Kontroler PID
    Y1(i) = Kesalahan(i);
    Y3(i)  = (Kesalahan(i) - Kesalahan(i-1))/dt;
    Y2(i)  = (Kesalahan(i) + Kesalahan(i-1))*dt/2;

    %adaptive interaction
    KP_dot(i) = Y1(i)*Kesalahan(i)*gamma;
    KI_dot(i) = Y2(i)*Kesalahan(i)*gamma;
    KD_dot(i) = Y3(i)*Kesalahan(i)*gamma;
    KP(i) = (KP_dot(i)+KP_dot(i-1))*dt/2;
    KP_SUM(i) = (sum(KP));
    KI(i) = (KI_dot(i)+KI_dot(i-1))*dt/2;
    KI_SUM(i) = (sum(KI));
    KD(i) = (KD_dot(i)+KD_dot(i-1))*dt/2;
    KD_SUM(i) = (sum(KD));

    %Tuning PID
    P(i) = KP_SUM(i)*Y1(i);
    I(i) = KI_SUM(i)*Y2(i);
    D(i) = KD_SUM(i)*Y3(i);
    PID(i)=P(i)+I(i)+D(i);
    U(i)=sum(PID);

    %Modelling PCT-100
    Yout(i+1)=0.9991*Yout(i)-0.06746*U(i+1)+0.06905*U(i);
end

%% Transpose
Yiha=Yout';
Yiha1=KP_SUM';
Yiha2=KI_SUM';
Yiha3=KD_SUM';

%% Plotting Hasil
figure (1)
plot(t(1,1:2001),Yout(1,1:2001),t(1,1:2001),Masukan(1,1:2001),'linewid
th',2),grid on;
```

```
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Level (cm)','FontWeight','normal','FontSize',12);
legend('Respon Sistem','Set Point');

figure (2)
plot(t(1,1:2000),KP_SUM(1,1:2000),'linewidth',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Kp','FontWeight','normal','FontSize',12);
legend('Kp');

figure (3)
plot(t(1,1:2000),KI_SUM(1,1:2000),'linewidth',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Ki','FontWeight','normal','FontSize',12);
legend('Ki');

figure (4)
plot(t(1,1:2000),KD_SUM(1,1:2000),'linewidth',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Kd','FontWeight','normal','FontSize',12);
legend('Kd');
```

Self Tuning PID TSE Tanpa Gangguan

```
%.....Created by.....%
%.....Muhammad Zakki Ghufron.....%
%.....2212100064.....%
%.....Electrical Engineering Department.....%
%.....Faculty of Industrial Technology.....%
%.....Institut Teknologi Sepuluh Nopember.....%

clear all;
```

```
clc;

%% Inisialisasi Waktu
t=0:0.1:200;       %Waktu Simulasi dan banyaknya data yang diambil

%% Inisialisasi Masukan

%Pengujian dengan bermacam-macam set point
%Masukan = input('Set Point='); %Masukkan Set Point bebas

%Pengujian dengan sinyal ramp
t1=0:0.01:6.5;              %Masukan sinyal ramp antara 0 - 6.5
t2=6.5:0.01:13.5;           %Masukan sinyal ramp antara 6.5 - 13.5
Masukan(1,1:651) =t1;       %Masukan sinyal ramp
Masukan(1,651:1301)=6.5;    %Masukan sinyal ramp
Masukan(1,1301:2001)=t2;    %Masukan sinyal ramp

%Pengujian perubahan set point
%Masukan(1,1:1001) =5;          %Masukan sinyal uji perubahan set
point
%Masukan(1,1001:2001)=6;        %Masukan sinyal uji perubahan set
point

%Pengujian dengan sinyal unit step
%Masukan(1,1:1201)=1;          %Masukan sinyal uji unit step

%% Inisialisasi variabel
dt = 0.1;                                %sampling time
n = 2000;                                %iterasi program
gamma =input('konstanta adaptive=');              %Masukkan gamma
KP_dot(1:n+1) = 0;KI_dot(1:n+1) = 0;KD_dot(1:n+1) = 0;
KP_SUM(1:n+1) = 0;KI_SUM(1:n+1) = 0;KD_SUM(1:n+1) = 0;
Kesalahan(1:n+1)=0;U(1:n+1)=0;Yout(1:n+1)=0;Yout(1:n+1)=0;
P(1:n+1)=0;I(1:n+1)=0;D(1:n+1)=0;

%% Perhitungan nilai kesalahan dan output pertama
%Checking kesalahan
Kesalahan(1) = Masukan(1)-0;
```

```
%Kontroler PID
Y1(1) = Kesalahan(1);
Y3(1)  = (Kesalahan(1) - 0)/dt;
Y2(1)  = (Kesalahan(1) + 0)*dt/2;

%adaptive interaction
KP_dot(1) = Y1(1)*t(1)*Kesalahan(1)*gamma;
KI_dot(1) = Y2(1)*t(1)*Kesalahan(1)*gamma;
KD_dot(1) = Y3(1)*t(1)*Kesalahan(1)*gamma;
KP(1) = (KP_dot(1)+0)*dt/2;
KP_SUM(1) = (sum(KP));          %Perubahan nilai KP
KI(1) = (KI_dot(1)+0)*dt/2;
KI_SUM(1) = (sum(KI));          %Perubahan nilai KI
KD(1) = (KD_dot(1)+0)*dt/2;
KD_SUM(1) = (sum(KD));          %Perubahan nilai KD

%Tuning PID
P(1) = KP_SUM(1)*Y1(1);
I(1) = KI_SUM(1)*Y2(1);
D(1) = KD_SUM(1)*Y3(1);
PID(1)=P(1)+I(1)+D(1);
U(1)=sum(PID);                  %Sinyal kontrol

%Modelling PCT-100
Yout(2)=-0.06746*U(2)+0.06905*U(1);

%% Simulasi
for i=2:n;
  %Checking kesalahan
  Kesalahan(i) = Masukan(i) - Yout(i);

  %Kontroler PID
  Y1(i) = Kesalahan(i);
  Y3(i)  = (Kesalahan(i) - Kesalahan(i-1))/dt;
  Y2(i)  = (Kesalahan(i) + Kesalahan(i-1))*dt/2;

  %adaptive interaction
  KP_dot(i) = Y1(i)*t(i)*Kesalahan(i)*gamma;
  KI_dot(i) = Y2(i)*t(i)*Kesalahan(i)*gamma;
```

```
    KD_dot(i) = Y3(i)*t(i)*Kesalahan(i)*gamma;
    KP(i) = (KP_dot(i)+KP_dot(i-1))*dt/2;
    KP_SUM(i) = (sum(KP));
    KI(i) = (KI_dot(i)+KI_dot(i-1))*dt/2;
    KI_SUM(i) = (sum(KI));
    KD(i) = (KD_dot(i)+KD_dot(i-1))*dt/2;
    KD_SUM(i) = (sum(KD));

    %Tuning PID
    P(i) = KP_SUM(i)*Y1(i);
    I(i) = KI_SUM(i)*Y2(i);
    D(i) = KD_SUM(i)*Y3(i);
    PID(i)=P(i)+I(i)+D(i);
    U(i)=sum(PID);

    %Modelling PCT-100
    Yout(i+1)=0.9991*Yout(i)-0.06746*U(i+1)+0.06905*U(i);
end

%% Transpose
Yiha=Yout';
Yiha1=KP_SUM';
Yiha2=KI_SUM';
Yiha3=KD_SUM';

%% Plotting Hasil
figure (1)
plot(t(1,1:2001),Yout(1,1:2001),t(1,1:2001),Masukan(1,1:2001),'linewid
th',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Level (cm)','FontWeight','normal','FontSize',12);
legend('Respon Sistem','Set Point');

figure (2)
plot(t(1,1:2000),KP_SUM(1,1:2000),'linewidth',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
```

```
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Kp','FontWeight','normal','FontSize',12);
legend('Kp');

figure (3)
plot(t(1,1:2000),KI_SUM(1,1:2000),'linewidth',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Ki','FontWeight','normal','FontSize',12);
legend('Ki');

figure (4)
plot(t(1,1:2000),KD_SUM(1,1:2000),'linewidth',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Kd','FontWeight','normal','FontSize',12);
legend('Kd');
```

Self Tuning PID STSE Tanpa Gangguan

```
%.....................................Created by.....................................%
%.............................Muhammad Zakki Ghufron.............................%
%.....................................2212100064.....................................%
%.........................Electrical Engineering Department.....................%
%.........................Faculty of Industrial Technology.....................%
%.........................Institut Teknologi Sepuluh Nopember.....................%

clear all;
clc;

%% Inisialisasi Waktu
t=0:0.1:200;      %Waktu Simulasi dan banyaknya data yang diambil

%% Inisialisasi Masukan

%Pengujian dengan bermacam-macam set point
%Masukan = input('Set Point='); %Masukkan Set Point bebas
```

```
%Pengujian dengan sinyal ramp
t1=0:0.01:6.5;              %Masukan sinyal ramp antara 0 - 6.5
t2=6.5:0.01:13.5;           %Masukan sinyal ramp antara 6.5 - 13.5
Masukan(1,1:651) =t1;       %Masukan sinyal ramp
Masukan(1,651:1301)=6.5;    %Masukan sinyal ramp
Masukan(1,1301:2001)=t2;    %Masukan sinyal ramp

%Pengujian perubahan set point
%Masukan(1,1:601) =5;        %Masukan sinyal uji perubahan set point
%Masukan(1,601:2001)=6;      %Masukan sinyal uji perubahan set
point


%Pengujian dengan sinyal unit step
%Masukan(1,1:1201)=1;        %Masukan sinyal uji unit step

%% Inisialisasi variabel
dt = 0.1;                                %sampling time
n = 2000;                                %iterasi program
gamma =input('konstanta adaptive=');              %Masukkan gamma
KP_dot(1:n+1) = 0;KI_dot(1:n+1) = 0;KD_dot(1:n+1) = 0;
KP_SUM(1:n+1) = 0;KI_SUM(1:n+1) = 0;KD_SUM(1:n+1) = 0;
Kesalahan(1:n+1)=0;U(1:n+1)=0;Yout(1:n+1)=0;Yout(1:n+1)=0;
P(1:n+1)=0;I(1:n+1)=0;D(1:n+1)=0;

%% Perhitungan nilai kesalahan dan output pertama
%Checking kesalahan
Kesalahan(1) = Masukan(1)-0;

%Kontroler PID
Y1(1) = Kesalahan(1);
Y3(1)  = (Kesalahan(1) - 0)/dt;
Y2(1)  = (Kesalahan(1) + 0)*dt/2;

%adaptive interaction
KP_dot(1) = Y1(1)*t(1)*t(1)*Kesalahan(1)*gamma;
KI_dot(1) = Y2(1)*t(1)*t(1)*Kesalahan(1)*gamma;
KD_dot(1) = Y3(1)*t(1)*t(1)*Kesalahan(1)*gamma;
KP(1) = (KP_dot(1)+0)*dt/2;
```

```
KP_SUM(1) = (sum(KP));          %Perubahan nilai KP
KI(1) = (KI_dot(1)+0)*dt/2;
KI_SUM(1) = (sum(KI));          %Perubahan nilai KI
KD(1) = (KD_dot(1)+0)*dt/2;
KD_SUM(1) = (sum(KD));          %Perubahan nilai KD

%Tuning PID
P(1) = KP_SUM(1)*Y1(1);
I(1) = KI_SUM(1)*Y2(1);
D(1) = KD_SUM(1)*Y3(1);
PID(1)=P(1)+I(1)+D(1);
U(1)=sum(PID);                  %Sinyal kontrol

%Modelling PCT-100
Yout(2)=-0.06746*U(2)+0.06905*U(1);

%% Simulasi
for i=2:n;
  %Checking kesalahan
  Kesalahan(i) = Masukan(i) - Yout(i);

  %Kontroler PID
  Y1(i) = Kesalahan(i);
  Y3(i)  = (Kesalahan(i) - Kesalahan(i-1))/dt;
  Y2(i)  = (Kesalahan(i) + Kesalahan(i-1))*dt/2;

  %adaptive interaction
  KP_dot(i) = Y1(i)*t(i)*t(i)*Kesalahan(i)*gamma;
  KI_dot(i) = Y2(i)*t(i)*t(i)*Kesalahan(i)*gamma;
  KD_dot(i) = Y3(i)*t(i)*t(i)*Kesalahan(i)*gamma;
  KP(i) = (KP_dot(i)+KP_dot(i-1))*dt/2;
  KP_SUM(i) = (sum(KP));
  KI(i) = (KI_dot(i)+KI_dot(i-1))*dt/2;
  KI_SUM(i) = (sum(KI));
  KD(i) = (KD_dot(i)+KD_dot(i-1))*dt/2;
  KD_SUM(i) = (sum(KD));

  %Tuning PID
  P(i) = KP_SUM(i)*Y1(i);
```

```
    I(i) = KI_SUM(i)*Y2(i);
    D(i) = KD_SUM(i)*Y3(i);
    PID(i)=P(i)+I(i)+D(i);
    U(i)=sum(PID);


    %Modelling PCT-100
    Yout(i+1)=0.9991*Yout(i)-0.06746*U(i+1)+0.06905*U(i);
end

%% Transpose
Yiha=Yout';
Yiha1=KP_SUM';
Yiha2=KI_SUM';
Yiha3=KD_SUM';

%% Plotting Hasil
figure (1)
plot(t(1,1:2001),Yout(1,1:2001),t(1,1:2001),Masukan(1,1:2001),'linewid
th',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Level (%)','FontWeight','normal','FontSize',12);
legend('Respon Sistem','Set Point');

figure (2)
plot(t(1,1:2000),KP_SUM(1,1:2000),'linewidth',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Kp','FontWeight','normal','FontSize',12);
legend('Kp');

figure (3)
plot(t(1,1:2000),KI_SUM(1,1:2000),'linewidth',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
```

```
ylabel('Ki','FontWeight','normal','FontSize',12);
legend('Ki');

figure (4)
plot(t(1,1:2000),KD_SUM(1,1:2000),'linewidth',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Kd','FontWeight','normal','FontSize',12);
legend('Kd');
```

Self Tuning PID SE Gangguan

```
clear all;
clc;

%% Inisialisasi Waktu
t=0:0.1:200;      %Waktu Simulasi dan banyaknya data yang diambil
t1=0:0.1:65;      %Waktu simulasi tanpa beban
t2=65:0.1:130;     %Waktu simulasi pemberian beban 1
t3=130:0.1:200;    %Waktu simulasi pemberian beban 2

%% Inisialisasi Masukan

%Masukan2 = input('Set Point='); %Masukkan Set Point bebas
t1=0:0.01:3.5;               %Masukan sinyal ramp antara 0 - 3.5
t2=3.5:0.01:8.5;             %Masukan sinyal ramp antara 3.5 - 6.5
Masukan(1,1:351) =t1;        %Masukan sinyal ramp
Masukan(1,351:701)=3.5;      %Masukan sinyal ramp
Masukan(1,701:1201)=t2;      %Masukan sinyal ramp
Masukan1(1,1:601) =1;        %Masukan sinyal uji perubahan set point
Masukan1(1,601:1201)=5;      %Masukan sinyal uji perubahan set
point
Masukan2(1,1:2001)=5;        %Masukan sinyal uji unit step

%% Inisialisasi variabel
dt = 0.1;                              %sampling time
n = 2000;                              %iterasi program
gamma =input('konstanta adaptive=');          %Masukkan gamma
KP_dot(1:n+1) = 0;KI_dot(1:n+1) = 0;KD_dot(1:n+1) = 0;
```

```
KP_SUM(1:n+1) = 0;KI_SUM(1:n+1) = 0;KD_SUM(1:n+1) = 0;
Kesalahan(1:n+1)=0;U(1:n+1)=0;Yout(1:n+1)=0;Yout(1:n+1)=0;
P(1:n+1)=0;I(1:n+1)=0;D(1:n+1)=0;

%% Perhitungan nilai kesalahan dan output pertama
%Checking kesalahan
Kesalahan(1) = Masukan2(1)-0;

%Kontroler PID
Y1(1) = Kesalahan(1);
Y3(1)  = (Kesalahan(1) - 0)/dt;
Y2(1)  = (Kesalahan(1) + 0)*dt/2;

%adaptive interaction
KP_dot(1) = Y1(1)*Kesalahan(1)*gamma;
KI_dot(1) = Y2(1)*Kesalahan(1)*gamma;
KD_dot(1) = Y3(1)*Kesalahan(1)*gamma;
KP(1) = (KP_dot(1)+0)*dt/2;
KP_SUM(1) = (sum(KP));          %Perubahan nilai KP
KI(1) = (KI_dot(1)+0)*dt/2;
KI_SUM(1) = (sum(KI));          %Perubahan nilai KI
KD(1) = (KD_dot(1)+0)*dt/2;
KD_SUM(1) = (sum(KD));          %Perubahan nilai KD

%Tuning PID
P(1) = KP_SUM(1)*Y1(1);
I(1) = KI_SUM(1)*Y2(1);
D(1) = KD_SUM(1)*Y3(1);
PID(1)=P(1)+I(1)+D(1);
U(1)=sum(PID);                  %Sinyal kontrol

%Modelling PCT-100
Yout(2)=-0.06746*U(2)+0.06905*U(1);

%% Simulasi
for i=2:n;
  %Checking kesalahan
  Kesalahan(i) = Masukan2(i) - Yout(i);
```

```
    %Kontroler PID
    Y1(i) = Kesalahan(i);
    Y3(i)  = (Kesalahan(i) - Kesalahan(i-1))/dt;
    Y2(i)  = (Kesalahan(i) + Kesalahan(i-1))*dt/2;

    %adaptive interaction
    KP_dot(i) = Y1(i)*Kesalahan(i)*gamma;
    KI_dot(i) = Y2(i)*Kesalahan(i)*gamma;
    KD_dot(i) = Y3(i)*Kesalahan(i)*gamma;
    KP(i) = (KP_dot(i)+KP_dot(i-1))*dt/2;
    KP_SUM(i) = (sum(KP));
    KI(i) = (KI_dot(i)+KI_dot(i-1))*dt/2;
    KI_SUM(i) = (sum(KI));
    KD(i) = (KD_dot(i)+KD_dot(i-1))*dt/2;
    KD_SUM(i) = (sum(KD));

    %Tuning PID
    P(i) = KP_SUM(i)*Y1(i);
    I(i) = KI_SUM(i)*Y2(i);
    D(i) = KD_SUM(i)*Y3(i);
    PID(i)=P(i)+I(i)+D(i);
    U(i)=sum(PID);


    %Modelling PCT-100
    %Yout(i+1)=0.9991*Yout(i)-0.06746*U(i+1)+0.06905*U(i);

    %Pemberian beban
    if i<=692;
        Yout(i+1)=0.9991*Yout(i)-0.06746*U(i+1)+0.06905*U(i);
        Noise=awgn(Yout,25,'measured');
    elseif i>692 && i<=1392;
        Yout(i+1)=0.9978*Yout(i)-0.06749*U(i+1)+0.06908*U(i);
        Noise=awgn(Yout,25,'measured');
    else
        Yout(i+1)=0.9967*Yout(i)-0.06736*U(i+1)+0.0689*U(i);
        Noise=awgn(Yout,25,'measured');
    end
end
```

```
%Pemberian Noise
Yout(n+1)=Yout(n);
Noise=awgn(Yout,40,'measured');

%% Transpose
Yiha=Yout';
Yiha1=KP_SUM';
Yiha2=KI_SUM';
Yiha3=KD_SUM';
Yiha4=Noise';

%% Plotting Hasil
figure (1)
plot(t(1,1:2001),Yout(1,1:2001),t(1,1:2001),Masukan2(1,1:2001),'linewi
dth',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Level (%)','FontWeight','normal','FontSize',12);
legend('Respon Sistem','Set Point');

figure (2)
plot(t(1,1:1001),Noise(1,1:1001),t(1,1:1001),Masukan2(1,1:1001),'linew
idth',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Level (%)','FontWeight','normal','FontSize',12);
legend('Respon Sistem dengan Noise','Set Point');


figure (3)
plot(t(1,1:1000),KP_SUM(1,1:1000),'linewidth',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Kp','FontWeight','normal','FontSize',12);
legend('Kp');
```

```
figure (4)
plot(t(1,1:1000),KI_SUM(1,1:1000),'linewidth',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Ki','FontWeight','normal','FontSize',12);
legend('Ki');

figure (5)
plot(t(1,1:1000),KD_SUM(1,1:1000),'linewidth',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Kd','FontWeight','normal','FontSize',12);
legend('Kd');
```

Self Tuning PID TSE Gangguan

```
clear all;
clc;

%% Inisialisasi Waktu
t=0:0.1:200;      %Waktu Simulasi dan banyaknya data yang diambil
t1=0:0.1:65;      %Waktu simulasi tanpa beban
t2=65:0.1:130;     %Waktu simulasi pemberian beban 1
t3=130:0.1:200;    %Waktu simulasi pemberian beban 2

%% Inisialisasi Masukan

%Masukan2 = input('Set Point='); %Masukkan Set Point bebas
t1=0:0.01:3.5;              %Masukan sinyal ramp antara 0 - 3.5
t2=3.5:0.01:8.5;            %Masukan sinyal ramp antara 3.5 - 6.5
Masukan(1,1:351) =t1;         %Masukan sinyal ramp
Masukan(1,351:701)=3.5;       %Masukan sinyal ramp
Masukan(1,701:1201)=t2;       %Masukan sinyal ramp
Masukan1(1,1:601) =1;         %Masukan sinyal uji perubahan set point
Masukan1(1,601:1201)=5;       %Masukan sinyal uji perubahan set
point
```

```
Masukan2(1,1:2001)=5;          %Masukan sinyal uji unit step

%% Inisialisasi variabel
dt = 0.1;                                  %sampling time
n = 2000;                                  %iterasi program
gamma =input('konstanta adaptive=');              %Masukkan gamma
KP_dot(1:n+1) = 0;KI_dot(1:n+1) = 0;KD_dot(1:n+1) = 0;
KP_SUM(1:n+1) = 0;KI_SUM(1:n+1) = 0;KD_SUM(1:n+1) = 0;
Kesalahan(1:n+1)=0;U(1:n+1)=0;Yout(1:n+1)=0;Yout(1:n+1)=0;
P(1:n+1)=0;I(1:n+1)=0;D(1:n+1)=0;

%% Perhitungan nilai kesalahan dan output pertama
%Checking kesalahan
Kesalahan(1) = Masukan2(1)-0;

%Kontroler PID
Y1(1) = Kesalahan(1);
Y3(1)  = (Kesalahan(1) - 0)/dt;
Y2(1)  = (Kesalahan(1) + 0)*dt/2;

%adaptive interaction
KP_dot(1) = Y1(1)*t(1)*Kesalahan(1)*gamma;
KI_dot(1) = Y2(1)*t(1)*Kesalahan(1)*gamma;
KD_dot(1) = Y3(1)*t(1)*Kesalahan(1)*gamma;
KP(1) = (KP_dot(1)+0)*dt/2;
KP_SUM(1) = (sum(KP));         %Perubahan nilai KP
KI(1) = (KI_dot(1)+0)*dt/2;
KI_SUM(1) = (sum(KI));         %Perubahan nilai KI
KD(1) = (KD_dot(1)+0)*dt/2;
KD_SUM(1) = (sum(KD));         %Perubahan nilai KD

%Tuning PID
P(1) = KP_SUM(1)*Y1(1);
I(1) = KI_SUM(1)*Y2(1);
D(1) = KD_SUM(1)*Y3(1);
PID(1)=P(1)+I(1)+D(1);
U(1)=sum(PID);                 %Sinyal kontrol

%Modelling PCT-100
```

```
Yout(2)=-0.06746*U(2)+0.06905*U(1);

%% Simulasi
for i=2:n;
  %Checking kesalahan
  Kesalahan(i) = Masukan2(i) - Yout(i);

  %Kontroler PID
  Y1(i) = Kesalahan(i);
  Y3(i)  = (Kesalahan(i) - Kesalahan(i-1))/dt;
  Y2(i)  = (Kesalahan(i) + Kesalahan(i-1))*dt/2;

  %adaptive interaction
  KP_dot(i) = Y1(i)*t(i)*Kesalahan(i)*gamma;
  KI_dot(i) = Y2(i)*t(i)*Kesalahan(i)*gamma;
  KD_dot(i) = Y3(i)*t(i)*Kesalahan(i)*gamma;
  KP(i) = (KP_dot(i)+KP_dot(i-1))*dt/2;
  KP_SUM(i) = (sum(KP));
  KI(i) = (KI_dot(i)+KI_dot(i-1))*dt/2;
  KI_SUM(i) = (sum(KI));
  KD(i) = (KD_dot(i)+KD_dot(i-1))*dt/2;
  KD_SUM(i) = (sum(KD));

  %Tuning PID
  P(i) = KP_SUM(i)*Y1(i);
  I(i) = KI_SUM(i)*Y2(i);
  D(i) = KD_SUM(i)*Y3(i);
  PID(i)=P(i)+I(i)+D(i);
  U(i)=sum(PID);

  %Pemberian beban
  if i<=692;
      Yout(i+1)=0.9991*Yout(i)-0.06746*U(i+1)+0.06905*U(i);
      Noise=awgn(Yout,25,'measured');
  elseif i>692 && i<=1392;
      Yout(i+1)=0.9978*Yout(i)-0.06749*U(i+1)+0.06908*U(i);
      Noise=awgn(Yout,25,'measured');
  else
      Yout(i+1)=0.9967*Yout(i)-0.06736*U(i+1)+0.0689*U(i);
```

```
        Noise=awgn(Yout,25,'measured');
    end
end

%Pemberian Noise
Yout(n+1)=Yout(n);
Noise=awgn(Yout,40,'measured');

%% Transpose
Yiha=Yout';
Yiha1=KP_SUM';
Yiha2=KI_SUM';
Yiha3=KD_SUM';
Yiha4=Noise';

%% Plotting Hasil
figure (1)
plot(t(1,1:1001),Yout(1,1:1001),t(1,1:1001),Masukan2(1,1:1001),'linewi
dth',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Level (%)','FontWeight','normal','FontSize',12);
legend('Respon Sistem','Set Point');

figure (2)
plot(t(1,1:1001),Noise(1,1:1001),t(1,1:1001),Masukan2(1,1:1001),'linew
idth',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Level (%)','FontWeight','normal','FontSize',12);
legend('Respon Sistem dengan Noise','Set Point');


figure (3)
plot(t(1,1:1000),KP_SUM(1,1:1000),'linewidth',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
```

```
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Kp','FontWeight','normal','FontSize',12);
legend('Kp');

figure (4)
plot(t(1,1:1000),KI_SUM(1,1:1000),'linewidth',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Ki','FontWeight','normal','FontSize',12);
legend('Ki');

figure (5)
plot(t(1,1:1000),KD_SUM(1,1:1000),'linewidth',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Kd','FontWeight','normal','FontSize',12);
legend('Kd');
```

Self Tuning PID STSE Gangguan

```
clear all;
clc;

%% Inisialisasi Waktu
t=0:0.1:200;      %Waktu Simulasi dan banyaknya data yang diambil
t1=0:0.1:65;      %Waktu simulasi tanpa beban
t2=65:0.1:130;     %Waktu simulasi pemberian beban 1
t3=130:0.1:200;    %Waktu simulasi pemberian beban 2

%% Inisialisasi Masukan

%Masukan2 = input('Set Point='); %Masukkan Set Point bebas
t1=0:0.01:3.5;                %Masukan sinyal ramp antara 0 - 3.5
t2=3.5:0.01:8.5;              %Masukan sinyal ramp antara 3.5 - 6.5
Masukan(1,1:351) =t1;         %Masukan sinyal ramp
Masukan(1,351:701)=3.5;       %Masukan sinyal ramp
Masukan(1,701:1201)=t2;       %Masukan sinyal ramp
Masukan1(1,1:601) =1;         %Masukan sinyal uji perubahan set point
```

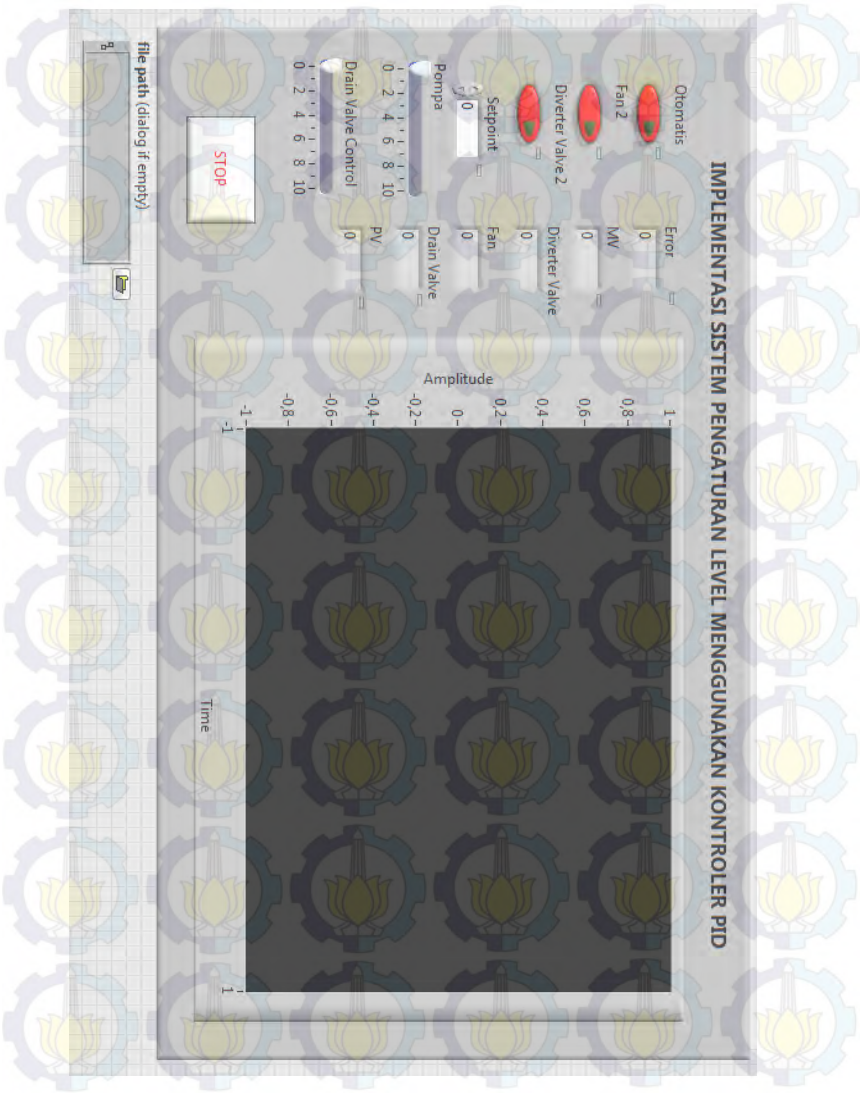
```
Masukan1(1,601:1201)=5;        %Masukan sinyal uji perubahan set
point
Masukan2(1,1:2001)=5;          %Masukan sinyal uji unit step

%% Inisialisasi variabel
dt = 0.1;                                %sampling time
n = 2000;                                 %iterasi program
gamma =input('konstanta adaptive=');              %Masukkan gamma
KP_dot(1:n+1) = 0;KI_dot(1:n+1) = 0;KD_dot(1:n+1) = 0;
KP_SUM(1:n+1) = 0;KI_SUM(1:n+1) = 0;KD_SUM(1:n+1) = 0;
Kesalahan(1:n+1)=0;U(1:n+1)=0;Yout(1:n+1)=0;Yout(1:n+1)=0;
P(1:n+1)=0;I(1:n+1)=0;D(1:n+1)=0;

%% Perhitungan nilai kesalahan dan output pertama
%Checking kesalahan
Kesalahan(1) = Masukan2(1)-0;

%Kontroler PID
Y1(1) = Kesalahan(1);
Y3(1)  = (Kesalahan(1) - 0)/dt;
Y2(1)  = (Kesalahan(1) + 0)*dt/2;

%adaptive interaction
KP_dot(1) = Y1(1)*t(1)*t(1)*Kesalahan(1)*gamma;
KI_dot(1) = Y2(1)*t(1)*t(1)*Kesalahan(1)*gamma;
KD_dot(1) = Y3(1)*t(1)*t(1)*t(1)*Kesalahan(1)*gamma;
KP(1) = (KP_dot(1)+0)*dt/2;
KP_SUM(1) = (sum(KP));         %Perubahan nilai KP
KI(1) = (KI_dot(1)+0)*dt/2;
KI_SUM(1) = (sum(KI));         %Perubahan nilai KI
KD(1) = (KD_dot(1)+0)*dt/2;
KD_SUM(1) = (sum(KD));         %Perubahan nilai KD

%Tuning PID
P(1) = KP_SUM(1)*Y1(1);
I(1) = KI_SUM(1)*Y2(1);
D(1) = KD_SUM(1)*Y3(1);
PID(1)=P(1)+I(1)+D(1);
U(1)=sum(PID);                 %Sinyal kontrol
```

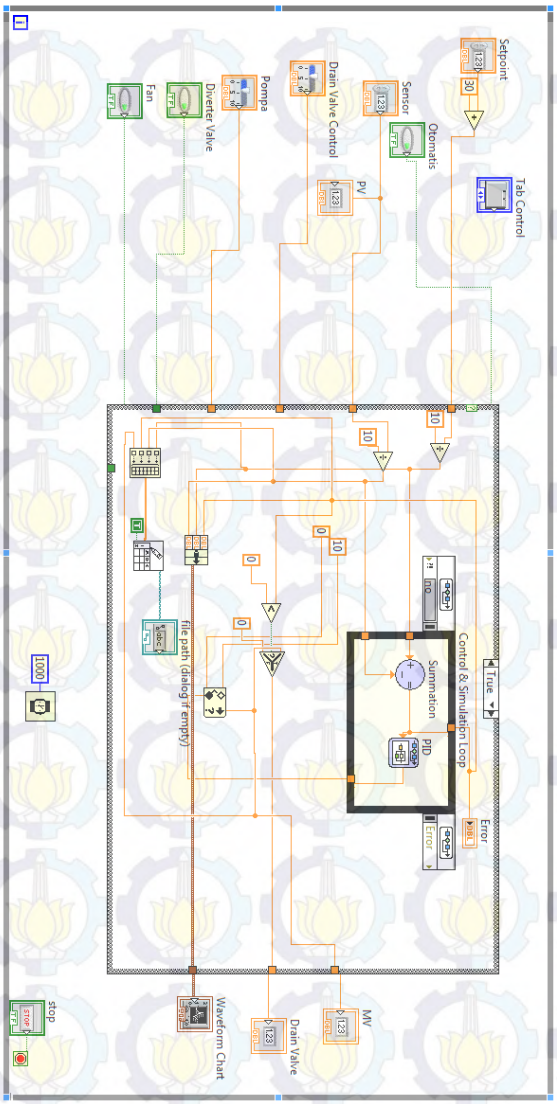
```
%Modelling PCT-100
Yout(2)=-0.06746*U(2)+0.06905*U(1);

%% Simulasi
for i=2:n;
  %Checking kesalahan
  Kesalahan(i) = Masukan2(i) - Yout(i);

  %Kontroler PID
  Y1(i) = Kesalahan(i);
  Y3(i)  = (Kesalahan(i) - Kesalahan(i-1))/dt;
  Y2(i)  = (Kesalahan(i) + Kesalahan(i-1))*dt/2;

  %adaptive interaction
  KP_dot(i) = Y1(i)*t(i)*t(i)*Kesalahan(i)*gamma;
  KI_dot(i) = Y2(i)*t(i)*t(i)*Kesalahan(i)*gamma;
  KD_dot(i) = Y3(i)*t(i)*t(i)*Kesalahan(i)*gamma;
  KP(i) = (KP_dot(i)+KP_dot(i-1))*dt/2;
  KP_SUM(i) = (sum(KP));
  KI(i) = (KI_dot(i)+KI_dot(i-1))*dt/2;
  KI_SUM(i) = (sum(KI));
  KD(i) = (KD_dot(i)+KD_dot(i-1))*dt/2;
  KD_SUM(i) = (sum(KD));

  %Tuning PID
  P(i) = KP_SUM(i)*Y1(i);
  I(i) = KI_SUM(i)*Y2(i);
  D(i) = KD_SUM(i)*Y3(i);
  PID(i)=P(i)+I(i)+D(i);
  U(i)=sum(PID);

  %Pemberian beban
  if i<=692;
     Yout(i+1)=0.9991*Yout(i)-0.06746*U(i+1)+0.06905*U(i);
     Noise=awgn(Yout,25,'measured');
  elseif i>692 && i<=1392;
     Yout(i+1)=0.9978*Yout(i)-0.06749*U(i+1)+0.06908*U(i);
     Noise=awgn(Yout,25,'measured');
```

```
  else
      Yout(i+1)=0.9967*Yout(i)-0.06736*U(i+1)+0.0689*U(i);
      Noise=awgn(Yout,25,'measured');
  end
end

%Pemberian Noise
Yout(n+1)=Yout(n);
Noise=awgn(Yout,40,'measured');

%% Transpose
Yiha=Yout';
Yiha1=KP_SUM';
Yiha2=KI_SUM';
Yiha3=KD_SUM';
Yiha4=Noise';

%% Plotting Hasil
figure (1)
plot(t(1,1:1001),Yout(1,1:1001),t(1,1:1001),Masukan2(1,1:1001),'linewi
dth',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Level (%)','FontWeight','normal','FontSize',12);
legend('Respon Sistem','Set Point');

figure (2)
plot(t(1,1:1001),Noise(1,1:1001),t(1,1:1001),Masukan2(1,1:1001),'linew
idth',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Level (%)','FontWeight','normal','FontSize',12);
legend('Respon Sistem dengan Noise','Set Point');


figure (3)
plot(t(1,1:1000),KP_SUM(1,1:1000),'linewidth',2),grid on;
```

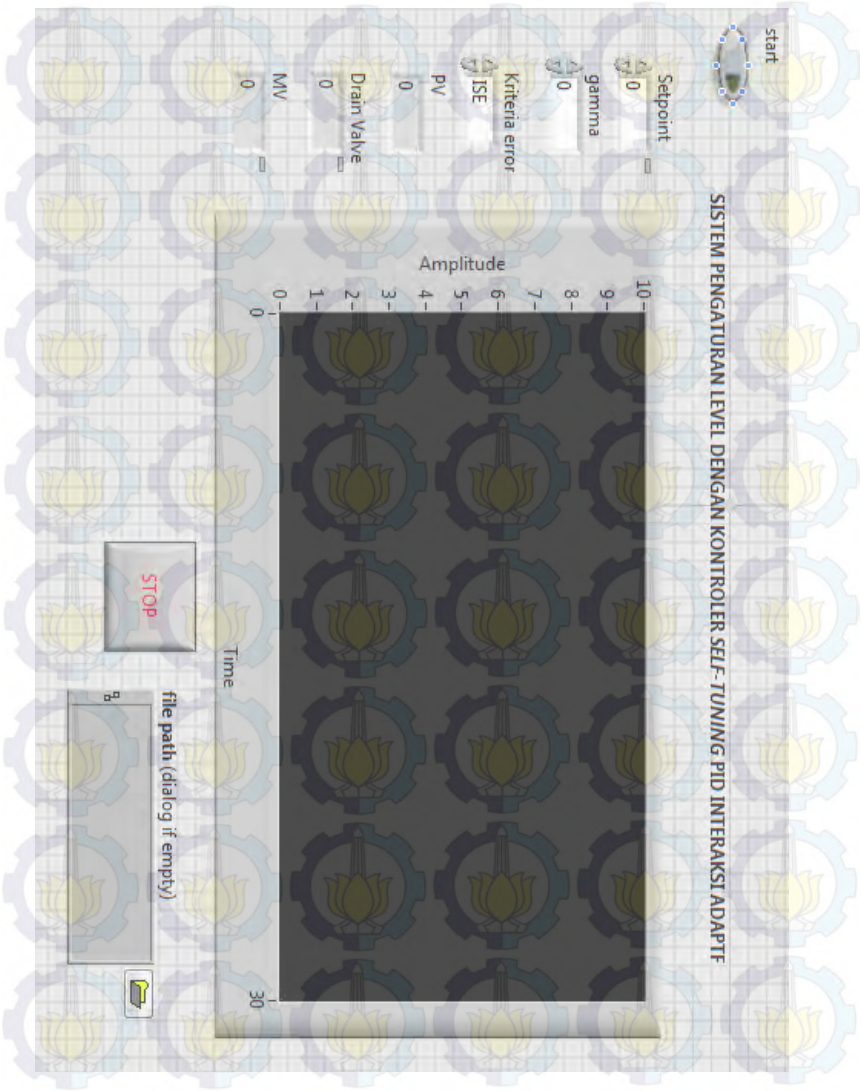
```
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Kp','FontWeight','normal','FontSize',12);
legend('Kp');

figure (4)
plot(t(1,1:1000),KI_SUM(1,1:1000),'linewidth',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Ki','FontWeight','normal','FontSize',12);
legend('Ki');

figure (5)
plot(t(1,1:1000),KD_SUM(1,1:1000),'linewidth',2),grid on;
title('Self Tuning PID by Adaptive
Interaction','FontWeight','Bold','FontSize',14);
xlabel('time t(second)','FontWeight','normal','FontSize',12);
ylabel('Kd','FontWeight','normal','FontSize',12);
legend('Kd');
```

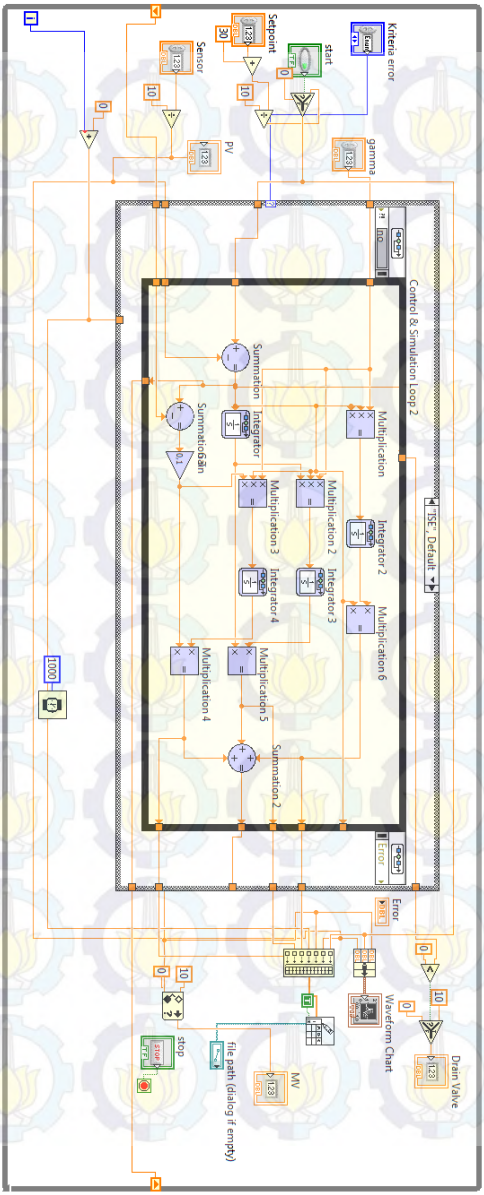
Program Implementasi Kontroler PID




Setpoint
30
Tab Control
Otomatis
Sensor
PV
Drain Valve Control
Pompa
Diverter Valve
Fan
10
0
Control & Simulation Loop
Summation
PID
True
Error
file path (dialog if empty)
1000
stop
STOP
Waveform Chart
Drain Valve
MV

Program Implementasi Kontroler STPID




Kritasia error
Setpoint
Sensor
start
PV
gamma
Control & Simulation Loop 2
Summation
Summation6
Integrator
Multiplication
Multiplication 2
Multiplication 3
Integrator 2
Integrator 3
Integrator 4
Multiplication 6
Multiplication 4
Multiplication 5
Summation 2
TSE: Default
Error
Waveform Chart
Drain Valve
MV
file path (dialog if empty)
stop
1000

DAFTAR PUSTAKA

[1] Pan, Yongping, dan Qinruo Wang,"*Research on a Stable Adaptive Fuzzy Control of Nonlinear Liquid Level System*", Sixth Internatioanl Conference on Hybrid Intelligent Systems (HIS), pp. 65, Rio de Janeiro, December 2006.

[2] Wu, Yifeng, dan Xiaoqi Lin,"*Reserach of Boiler Fault Diagnosis Based on Fuzzy Neural Network*", Internatioanl Conference on Information Science and Computer Application (ISCA).2013

[3] Chen, Liang, Cuizhu Wang, Yang Yu, dan Yawei Zhao, "*The research on Boiler Drum Water Level Control System Based on Self-Adaptive Fuzzy PID*", Control and Decision Conference (CCDC), pp. 1582 – 1584, Xuzhou, Chinese, May 2010.

[4] Ehsani, M.,"*Adaptive control by MRAC method",* Procedings OCEANS 2003, vol 1*,* San Diego, September, 2003

[5] Lin, Feng dan Robert D Brandt, and George Saikalis, "*Self-tuning od PIF Controllers by Adaptive Interaction*", Proceedings of the American Control Conference, pp. 3676 – 3681 vol.5, Chicago, Illonis, June 2000.

[6] Process Control Technology PCT -100, "*User Manual PCT-100*", Bytronic, 124 Anglesey Court, Towers Plaza, England.

[7] Yoyok Dwi Setyo Pambudi, '' *Penerapan Pemodelan dan Metode Kurva Reaksi Proses Untuk Mengindentifikasi Sistem DURESS*". Prosiding Pertemuan dan Presentasi Ilmiah Penelitian Dasar Ilmu Pengetahuan dan Teknologi Nuklir, Yogyakarta, 2011.

[8] Visoli, Antonia, "*Practical PID Control*", Springer Media, 2006.

[9] Xue, Dingyu, YangQuan Chen, dan Derek P.Atherton,"*Linear Feedback Control Analysis and Design with MATLAB*", SIAM, 2007.

[10] Bhaksi, V.U. dan U.A. Bhaksi, "*Control System*", First Edition, Technical Publications Pune, 2007.

[11] Gundogdu, Tayfun, Guven Komurgoz, "*Adaptive PID Controller Desogn by Using Adaptive Interaction Approach Theory*", 3rd International Conference Elecric Power and Energy Conversion Systems, pp. 1- 5 , Istanbul, Turkey, October 2013.

[12] Astrom, Karl J., and Tore Hagglund, "*PID Controllers: Theory, Design, and Tuning*", 2nd Edition, Instrument Society of America, United States of America, 1988.

[13] Ibrahim, Dogan, "*Microcontroller Based Applied Digital Control*", John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, England, 2006.
[14] Kuo, Benjamin C., and Farid Golnaraghi, " *Automatic Control System*", John Wiley & Sons, Ninth edition, 1991.
[15] Saikalis, George dan Feng Lin, "*Adaptive Neural Network Control by Adaptive Interaction*", Proceedings of the American Control Conference, pp. 1247 – 1252 vol.2, Arlington, 2001.
[16] Luenberger, David G, "*Optimization by Vector Space Methods*", John Wiley & Sons, Inc, Canada, 1969.

RIWAYAT PENULIS

Muhammad Zakki Ghufron dilahirkan di Malang, 3 September 1994. Merupakan anak pertama dari 3 bersaudara pasangan Bapak Dr. Muhadjir Anwar SE., MM. dan Ninik Widiastuti S.Pd. Penulis menamatkan pendidikan Sekolah Dasar di SDN Lesanpuro 04, SMPN 21 Malang, dan SMAN 5 Malang. Setelah menamatkan jenjang SMA, penulis melanjutkan kuliah S1 Teknik Elektro di Institut Teknologi Sepuluh Nopember melalui seleksi tulis masuk perguruan tinggi. Penulis mengambil konsentrasi Teknik Sistem Pengaturan dan aktif sebagai anggota research Automation and Industrial Robotics Research Group (AIRRG) AJ-104. Pada bulan Januari, penulis mengikuti ujian Tugas Akhir sebagai syarat untuk menyelesaikan masa studi dan memperoleh gelar Sarjana Teknik Elektro.