TA / SEKJUR TE / 2006 / 026

PENGENDALI PID DENGAN GUI
(*Graphical User Interface*)
MATLAB

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana
Teknik Elektro

Oleh:

Wisang Wiyoso

00 524 094

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
JOGJAKARTA
2006

PENGENDALI PID DENGAN GUI (*Graphical User Interface*) MATLAB

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana
Teknik Elektro

Oleh:

Wisang Wiyoso

00 524 094

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
JOGJAKARTA
2006

LEMBAR PENGESAHAN PEMBIMBING

PENGENDALI PID DENGAN GUI

(Graphical User Interface)

MATLAB

TUGAS AKHIR

Oleh :

Nama : *Wisang Wiyoso*

No.mhs : *00 524 094*

JOGJAKARTA, FEBRUARI 2006

Di Sahkan Oleh :

Pembimbing 1	**Pembimbing II**
(Ir.Hj.Budi Astuti,MT)	**(Dwi Ana Ratna Wati,ST)**

LEMBAR PENGESAHAN DOSEN PENGUJI

PENGENDALI PID DENGAN GUI

(Graphical User Interface)

MATLAB

TUGAS AKHIR

Disusun oleh :

Nama : ***Wisang Wiyoso***

No.mhs : ***00 524 094***

Telah Dipertahankan Di Depan Sidang Penguji Sebagai Salah Satu Syarat
Untuk Mempertahankan Gelar Sarjana Teknik Elektro
Fakultas Teknologi Islam Indonesia

JOGJAKARTA, MARET 2006

Tim Penguji :

Ketua

IR. H. Budi Astuti,MT _____

Anggota I

Tito Yuwono,ST,M.Sc _____

Anggota II

Dwi Ana Ratna Wati,ST _____

Mengetahui,
Dekan Fakultas Industri
Universitas Islam Indonesia

UNIVERSITAS ISLAM INDONESIA ★ YOGYAKARTA ★ FAKULTAS TEKNOLOGI INDUSTRI

IR.H.Bahrun Sutrisno,M.Sc

HALAMAN PERSEMBAHAN

Tingkah lakuku seringkali menyakiti hatimu
tak menghiraukan nasehatmu
pernah mengkhianati kepercayaanmu
Namun Kau slalu menyayangiku dan mencintaiku

Saat ku terjatuh Kau menolongku
Merasa sedih Kau menghiburku
Menghadapi cobaan Kau mendoakanku
Tak tau arah Kau membimbingku

Tiada lelah kau memberikan kasih dan sayangmu
Menaruh kepercayaan pada diriku
Mendoakan untuk kemenanganku
Mengorbankan sgalanya untukku

Kini aku sadar
Bahwa kaulah anugrah terindah dalam hidupku
Yang slalu menerangi gelapku
Mencintaiku

Terimakasih Bunda 'Dima Utami' Ayahanda 'Sudaryono'
Tercinta
Masku 'Iyon Kristiona Juli Prabowo' serta Adeku'Yudhie Bayu Adi '
Tersayang

Aku mencintaimu
Sampai akhir hayatku
Untuk cinta kasih dan sayang yang telah kau berikan
Kupersembahkan karya ini untukmu...

MOTTO

Semua yang kita jalani ada hikmahnya

Teruslah melangkah dan ambil maknanya

Jangan pernah menyerah dalam menjalani cobaan

Karena cobaan merupakan ujian

Suatu pengajaran yang akan mengantarkan kita menuju pendewasaan

Jangan pernah letih tuk melangkah

Mencari jalan keluar setiap persoalan

Karena setiap persoalan mempunyai jawaban

Dan pelajaran yang berarti bagi diri kita

Yang mengalaminya

Hidup itu indah

jadi jangan kau sia-siakan dengan hal menyesatkan

Karena itu kan membuat hati takan tenang

Jika kita mengamalkan amal perbuatan yang baik

Niscaya hati kita kan damai, tentram dan bahagia

Jangan pernah lelah tuk melangkah

Dan teruslah melangkah menghadapi setiap masalah

Selama hayat masih di kandung badan jangan kau menyerah

Majulah dan gapai semua cita dan cintamu

Demi kehidupan sekarang dan yang kan datang

NEVER GIVE UP

KATA PENGANTAR

بسم الله الرحمن الرحيم

Assalamu'alaikum Wr. Wb.

Puji Syukur kami panjatkan kehadirat Allah SWT atas limpahan rahmat dan hidayah-Nya dan tak lupa sholawat serta salam senantiasa kita limpahkan kepada junjungan kita Nabi Muhammad SAW beserta keluaga dan sahabat serta pengikutnya sampai akhir zaman sehingga penulis dapat menyelesaikan Tugas Akhir ini sebagai syarat akhir untuk meraih gelar Sarjana Teknik di Jurusan Teknik Elektro Fakultas Teknologi industri Universitas Islam Indonesia.

Dalam penyusunan tugas akhir ini penulis banyak mendapat bantuan dari berbagai pihak, sehingga penulis ingin menyampaikan ucapan terima kasih kepada ;

1. Bpk Ir. Bachrun Sutrisno M.Sc, selaku Dekan Fakultas Teknologi Industri (FTI) Universitas Islam Indonesia (UII)
2. Ibu Ir. Hj. Budi Astuti MT, selaku Kajur Teknik Elektro dan sebagai dosen pembimbing I.
3. Bpk Wahyudi Budi Pramono ST, selaku Sekjur Teknik Elektro.
4. Ibu Dwi Ana Ratna Wati ST, selaku dosen pembimbing II dan Ka.Lab pemrograman Matlab dan kontroler dan instrumentasi atas waktu, kebaikan, kesabaran dan ilmunya.
5. Dosen dan karyawan Fakultas Teknologi Industri UII, Ka.Lab dan laboran jurusan Teknik Elektro atas waktu, tempat dan ilmu yang diberikan.
6. Seluruh mahasiswa jurusan Teknik Elektro UII.
7. Seluruh pihak yang tidak dapat di sebutkan satu-persatu, yang telah memberikan *support* dan doa.

Penulis menyadari bahwa Tugas Akhir ini masih terdapat kesalahan dan kekurangan. Oleh karena itu, kritik dan saran yang membangun akan senantiasa penulis terima dengan senang hati. Akhirnya, harapan penulis semoga Tugas Akhir ini dapat bermanfaat bagi kita semua. Amiin...

Wassalamu'alaikum Wr.Wb

Jogjakarta, Februari 2006

Wisang Wiyoso

Wisang Thanks 2.

- **Allah Swt** atas semua nikmat dan rahmat serta cobaan yang telah engkau berikan sehingga membuatKu menjadi seperti ini.
- Kedua Orangtuaku Ibunda **DimaUtami**, Ayahanda **Sudaryono** Tercinta... dan Kk'ku **YonKristionoJuliPrabowo** serta Ad'ku **YudiBayuAdi** Tersayang Thank's 4 semua cinta kasih yang tlah kau berikan 4 **WiesangBagoesWiyoso** IuvU 4ever smoga kita kan menjadi kluarga mawadah, warohmah n Sakinah slalu amien...
- MyHoney '**HenyEndyartie**' diZogza & MyDear '**SiskaN.Chizs**' diTegal yang telah menyayangiku dan mencintaiku apa adanya in state of Sad&Happy Now I cannot chosen one of U, IluvUall SoMuch.
- **MbahMohBani** yg baik hati & mau mengangkat aku jadi anaknya, Thank's 4 All yo Mbah smoga sehat slalu. becouse U, W can ngirit duit, awae lemu, ngerjain TA & bisa Jln^2 with MyGirl. Jg tak lupa anae mbae **MbaVia**, **MbaNur**, **Mba3**, **MasRahmat**, **MasFacthur** putuneMbae **Ayu**, **Vio** & **Ana**.

- MyBestFriend **HusnieThamrin** yg suka ma Tipe-x n pengen jd musisi but g bs nyiptain lagu he... k-cian deh U, **TeguhHartono** Playboy from Tegal yang haus akan wanita, g tau deh yg di cari yg spt apa??? But sadarlah teman hidup bukan hanya utk wanita, **MalikEistein** si Prof Jail + PujanggaCinta yg baik hati & mo minjemin something 4 me (Duit,Print,Kmptr,dll) thank's ya Q n mg cpt klar TAnya. **BaGoEnk** yg namanya ky ikatan Basket NBA(**NurBudiAgoenk**) yg Chubby abis but fear with all women except Adik ma Ibunya, smoga ia kan sgr mnemukan kjantanannya n menjadi DaredevilBoy. **YonoCol (Suyono)** Yg anaknya lugu n culun abiz but pgn dapet Cewe Ckp, smg Tercapai Tman. **ArabFauzan** yang always with me Tyus setiap kuliah dari ospek smp skrng, naksir x ya but Ane duluan ya Pren, smg bisnis Nt berjalan lancar n succes slalu 4 U but jgn lupa beramal Okre (jo plit^2).
- Old Friend **Fahmie** yg pgn jd org Tegal jg ky W, **Ipoel** yg dah ngebet nikah ma **Yunie**, **Lutfie** yg junkis abis n pgn jd Gmuk but blm nemu obat yg co^{2}k. mendingan ngracik sendiri aza Pren.

- MyFriend elektro00All Pa **Romie** yg sbk abiz, dr pagi smp mlm n pagi trus mlm n pagi lagi wah pkknya sbk bgt deh, **Hadi**CoenterHp thank's ya, **Ju'pri**, **Yulie**Ustadz, **Bramd**Ot rival BaGoEnK piss donk pren, **Sapto**, **Heru**SuTimbul yg licik abiz, **Wahyu** lohan dll yg g bs **WisangBogoesWiyoso** sbutin 1/1.
- Bocah^2 Krawitan **Andi**, **Suranto**, **Sutri**, **Kechu**, **Heri**, **Yanto** pokoke pemuda panutan & warga krawitan smuanya deh.
- MyMotorcycle **G3788DP** & **G3006AP** yg mo nganterin **WisangBagoesWiyoso** ke penjuru Tegal,Zogza & Dunia & MyBlueComputer yg tlah membantu mbuat MyTa n memberikan hiburan2 4 me.
- MyPetCat **Cole&Ayes** yg slalu mnemaniku & menghiburku dikala aku berada di CosKrawitan meooonk^2...
- **Anyone person** who recognize with **WisangBagoesWiyoso**. Thankyu thankyu & thankyu... **Somuch**.

ABSTRAK

Pengendali PID adalah pengendali berumpan balik yang paling populer di dunia industri dan hal yang penting dalam desain pengendali PID ialah menentukan paramater dari P, I dan D agar didapatkan respon sistem *close loop* yang memenuhi kriteria performansi yang diinginkan. Masing-masing pengendali secara keseluruhan bertujuan untuk mempercepat reaksi sebuah sistem, menghilangkan *offset* dan menghasilkan perubahan awal yang besar.

Dengan menggunakan GUI (*Graphical User Interface*) Matlab pengaturan parameter pengendali PID menjadi lebih mudah dan efisien karena *user* tidak perlu mengetahui baris perintah yang di gunakan tinggal memasukan sistem dan nilai yang kita inginkan pada GUI Matlab, GUI Matlab akan melaksanakan perintah dan menampilkan hasil *step respon*nya pada tampilan *scope* yang ada pada *Simulink*.

Pengendali PID dengan GUI Matlab dapat mengendalikan sistem Orde Satu yang berupa mesin dc dengan nilai KP = 10, KI = 100 dan KD = 0, Orde Dua yang berupa sistem pegas dengan nilai KP = 100, KI = 190, KD = 15, dan Orde Tiga yang berupa sistem kendali batang pesawat dengan nilai KP = 8, KI = 5, KD = 10, dan menghasilkan respon sistem yang lebih baik dari natural respon, akan tetapi kendali PID susah untuk menyetabilkan sistem yang mempunyai Orde yang lebih besar dari Tiga. Kendali *Proportional* tidak bisa bekerja sendiri dalam pengendalian untuk meyetabilkan suatu sistem, penggunaannya harus dikombinasikan dengan kendali *Integral* dan *Derivative*. Kendali *Proportional* berfungsi untuk menambah waktu naik, *Derivative* untuk mengurangi *overshoot* dan kendali *Integral* untuk menghilangkan kesalahan keadaan tunak, namun pada kenyataannya mengubah salah satu variabel dapat mengubah karakteristik lainnya.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR GRAFIK

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Dalam era globalisasi sektor industri memegang peranan penting khususnya di Indonesia, banyak industri-industri yang berkembang dengan pesat. Di dalam industri sangat dibutuhkan sistem kontrol yang baik untuk dapat menunjang proses berjalannya proses industri tersebut dan untuk meningkatkan efisiensi dalam proses produksi maka sistem kontrol harus dapat dikendalikan dengan mudah dan dapat memberikan kontribusi dalam dunia industri.

Dan sampai sekarang pengendali PID dalam dunia industri masih banyak digunakan selama kurang lebih 50 tahun pengendali PID terbukti dapat memberikan performansi kontrol yang baik.

Maka dari itu pembuatan sebuah kontrol PID yang dapat dikendalikan melalui GUI Matlab akan sangat membantu, karena suatu pengontrolan dapat dikendalikan dengan mudah oleh seseorang dengan adanya suatu tampilan grafis.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan di atas maka dapat diambil rumusan masalah sebagai berikut, Bagaimana mengendalikan kontrol PID serta mengatur keluaran dari sistem dengan mengubah parameter kontrol PID melalui GUI Matlab oleh seseorang.

1.3 Batasan Masalah.

Agar permasalahan yang dibahas dalam laporan skripsi ini tidak menyimpang dari judul yang telah di tetapkan maka perlu di tetapkan pokok-pokok permasalahan yang dibatasi pada masalah :

1. Pembuatan simulasi menggunakan Matlab 7.0.4
2. Sistem berupa fungsi alih, dan nilainya telah ditetapkan dalam tiap-tiap Orde.
3. Orde yang digunakan Orde Satu, Orde Dua dan Orde Tiga.
4. GUI Matlab dapat mengatur kontrol P(*Proportional*), I (*Integral*) dan D(*Derivative*) yang ada pada *Simulink* dengan nilai 0 – 1000.
5. GUI Matlab dapat menjalankan simulasi pada Simulink.

1.4 Tujuan Tugas Akhir

Tujuan pembuatan tugas akhir adalah membuat suatu pengontrol PID yang dapat mengatur parameternya serta dapat memasukan nilai dari suatu sistem yang ada pada *Simulink* melalui GUI Matlab dan mahasiswa dapat mempelajari dan memperoleh pengetahuan mengenai karakteristik kontrol PID serta pemmrograman GUI Matlab.

1.5 Metodologi Penelitian

1. Pengumpulan Data

   Data diperoleh dari studi pustaka berupa buku, artikel, makalah dan *tutorial* yang tersedia pada *website* di internet.

2. Studi Pustaka

Pengumpulan data ini digunakan untuk mendapatkan informasi-informasi yang berkaitan dengan proses penyusunan tugas akhir, sehingga dapat digunakan sebagai acuan dalam proses pembuatan target aplikasi.

3. Pemecahan Masalah

Setelah semua data terkumpul, maka dilakukan perancangan sistem, pembutan simulasi sistem dan pengujian sistem.

1.6 Sistematika Penulisan

Sistematika penulisan Tugas Akhir ini terdiri dari 5 bab bagian isi laporan, dengan penjelasan bab sebagai berikut :

BAB I : PENDAHULUAN

Berisi tentang latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, metodelogi penelitian dan sistematika penulisan.

BAB II : LANDASAN TEORI

Memuat dasar-dasar teori yang berhubungan dengan penelitian dan yang berhubungan dengan sistem yang berkenaan dengannya.

BAB III : PERANCANGAN SISTEM.

Menjelaskan tentang pemrograman GUI MATLAB dan perancangan *Simulink* dan pengujian sistem yang telah dibuat, serta berisi lebih

terperinci tentang apa yang telah disampaikan pada proposal Tugas Akhir.

BAB IV : ANALISIS DAN PEMBAHASAN

Membahas hasil dan analisis simulasi sistem yang telah dijalankan oleh GUI Matlab.

BAB V : PENUTUP

Berisi kesimpulan dari proses simulasi sistem, keterbatasan-keterbatasan yang ditemukan dan juga asumsi-asumsi yang dibuat selama melakukan penelitian dan saran-saran guna perbaikan serta pengembangan simulasi ini.

BAB II

LANDASAN TEORI

2.1 Pengendali PID

Pengendali adalah komponen yang berfungsi meminimasi sinyal kesalahan. Tipe pengendali yang paling populer adalah pengendali PID. Pengendali PID adalah pengendali berumpan balik yang sering digunakan dalam industri–industri. Pengendali PID terbukti dapat memberikan performansi kontrol yang baik meski mempunyai algoritma yang sederhana dan mudah dipahami.

Pengendali PID merupakan kombinasi dari kontrol *Proportional*, *Integral* dan *derivative* atau turunan, kombinasi ini mempunyai keuntungan dibandingkan dengan masing-masing kontroler.

Persamaan tiga kombinasi ini adalah sebagai berikut:

$$u(t) = K_p e(t) + \mathrm{K_I} \int_0^t e(t)dt + K_D \frac{de(t)}{dt} \qquad (2.1)$$

Transformasi Laplace untuk persamaan 2.1 menghasilkan fungsi alih:

$$\mathrm{U(s)} = \left(\mathrm{Kp} + \frac{Ki}{s} + \mathrm{K_D\, s}\right) \mathrm{E_{(s)}} \qquad (2.2)$$

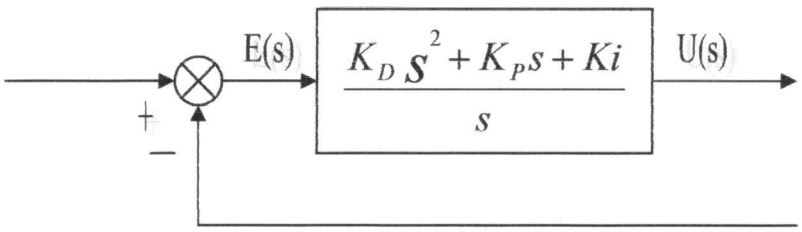
atau

$$\mathrm{Gc(s)} = \frac{U(s)}{E(s)} = \mathrm{Kp} + \frac{Ki}{s} + \mathrm{K_D\, s} \qquad (2.3)$$

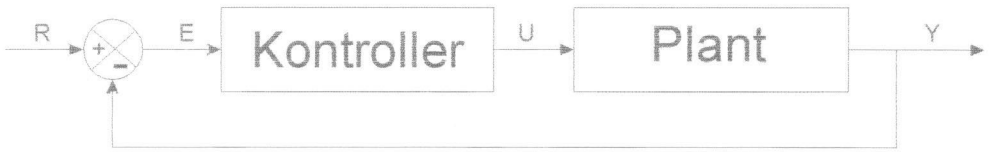
atau

$$\mathrm{Gc(s)} = \frac{U(s)}{E(s)} = \frac{K_D s^2 + K_P s + Ki}{s} \qquad (2.4)$$

- U(s) = Keluaran
- E(s) = Masukan
- s = *Frekuensi*
- G(s) = Perbandingan keluaran dan masukan
- Kp = *Proportional Gain*
- Ki = *Integral gain*
- K_D = *Derivative gain*



Gambar 2.1 Diagram blok kontrol PID

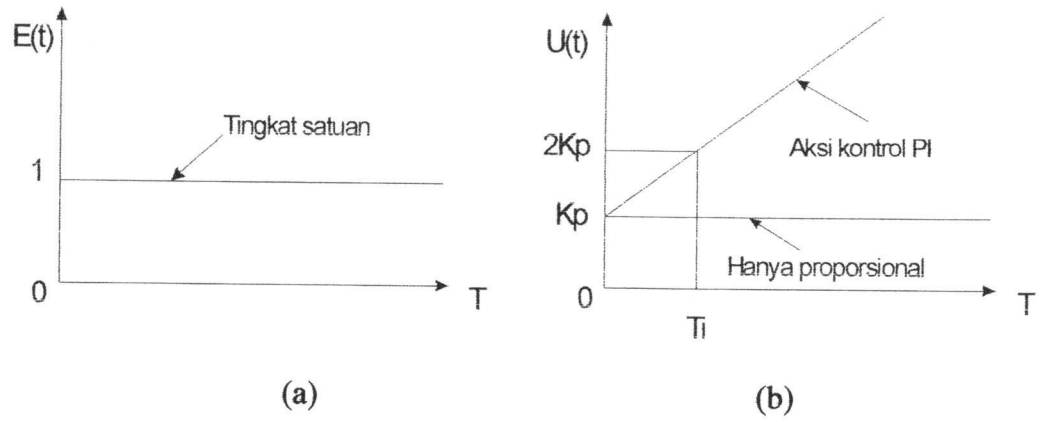


Gambar 2.2 Sistem *close loop*

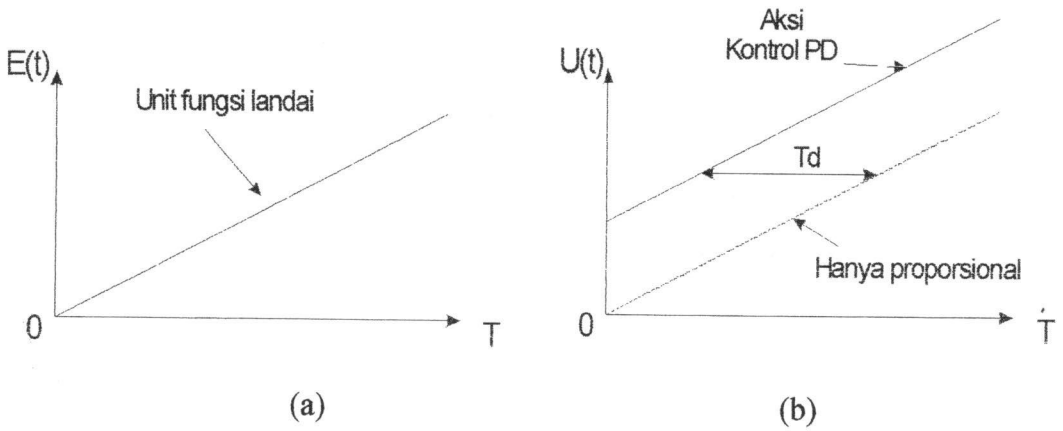
Cara kerja kontroler PID pada sistem *close loop* menggunakan skema yang terlihat pada gambar 2.2. Variabel (E) menggambarkan *tracking error*, nilai masukan yang berbeda (R), keluaran aktual (Y). *Signal error* ini akan dikirim ke PID kontroler, dan kontroler akan menghitung keseluruhan turunan dan *integral* dari *signal error* ini. Sinyal (U) yang telah melewati kontroler, sekarang sama dengan *proportional* penguatan (Kp) dikalikan ukuran kesalahannya ditambah penguatan *integral* (Ki) dikalikan ukuran kesalahan *integral*nya ditambah penguatan turunan (Kd) dikalikan kesalahan *derifasi*nya. sinyal (U) akan dikirim ke *plant,* dan akan mendapatkan keluaran baru (Y). keluaran baru (Y) ini akan

dikirim kembali ke sensor untuk mencari kesalahan sinyal baru (E). kontroler membawa kesalahan sinyal baru tersebut dan menghitung turunan-turunannya dan *integral-integral*nya lagi. Proses tersebut akan berjalan terus menerus seperti semula.

Elemen-elemen kontroler P, I dan D masing-masing secara keseluruhan bertujuan untuk mempercepat reaksi sebuah sistem, menghilangkan *offset* dan menghasilkan perubahan awal yang besar.

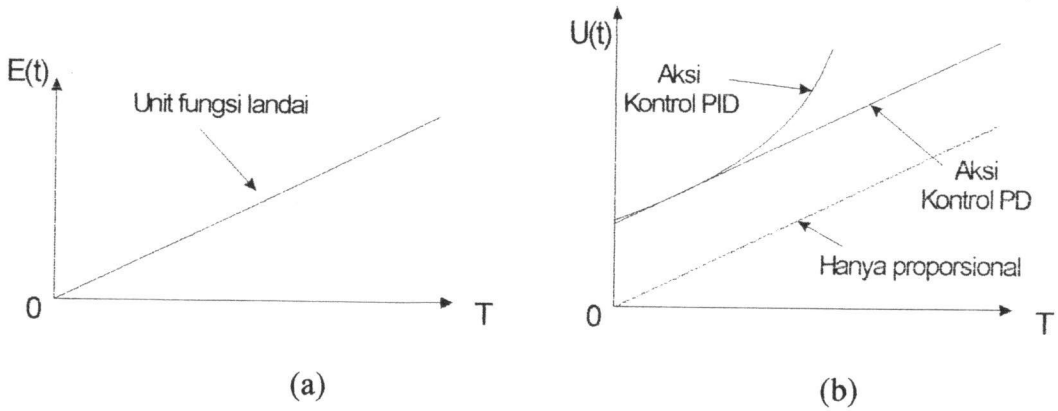


Gambar 2.3 (a) dan (b) diagram masukan langkah unit dan keluaran kontrol PI



Gambar 2.4

(a) dan (b) diagram unit masukan fungsi landai dan keluaran kontrol PD



(a) (b)

Gambar 2.5

(a) dan (b) Diagram unit masukan fungsi landai dan keluaran kontroler PID

Aspek yang sangat penting dalam desain kontrol PID ialah penentuan parameter kontrol PIDnya supaya sistem *close loop* memenuhi kriteria performansi yang diinginkan hal ini disebut juga dengan *tuning controller*.

2.1.1 Karakteristik PID Kontroler

Proportional kontroler (Kp) akan memberikan efek mengurangi waktu naik, tetapi tidak menghapus keadaan tunak. *Integral* kontroler (Ki) akan memberikan efek menghapus kesalahan keadaan tunak, tetapi memburuknya respon transient. *Derivative* kontroler akan memberikan efek meningkatnya stabilitas sistem, mengurangi *overshoot*, dan menaikan respon transfer. Efek dari setiap kontroler (Kp, Ki,Kd) dalam sistem loop tertutup dapat dilihat pada tabel 2.1.

Hubungan korelasi pada tabel mungkin tidak sepenuhnya akurat, karena Kp, Ki, dan Kd saling bebas karena pada kenyataannya, mengubah salah satu

variabel dapat mengubah dua yang lainnya. Karena alasan tersebut, tabel hanya digunakan sebagai referensi saat menentukan nilai Kp, Ki, dan Kd.

Respon Loop Tertutup	**Waktu Naik**	***Overshoot***	**Waktu Turun**	**Kesalahan Keadaan Tunak**
Kp	Menurun	Meningkat	Perubahan kecil	Menurun
Ki	Menurun	Meningkat	Meningkat	Hilang
Kd	Perubahan Kecil	Menurun	Menurun	Perubahan kecil

Tabel 2.1 Karakteristik kontroler PID

Suatu Sistem dikatakan stabil jika sistem tersebut akan tetap dalam keadan diam atau berhenti. Jika kontrol PID diaplikasikan pada GUI Matlab dan *Simulink* maka pengendalian suatu sistem agar stabil akan lebih mudah dilakukan oleh seseorang, dan juga akan mempermudah seseorang untuk mempelajari karakteristik kontrol PID tanpa harus mengetahui baris perintah yang digunakan.

2.2 *Simulink*

Simulink adalah *software* yang digunakan untuk memodelkan, melakukan simulasi dan analisa terhadap sistem dinamis. *Simulink* dapat digunakan untuk sistem linier maupun nonlinier, yang dimodelkan dalam waktu *continue*, waktu

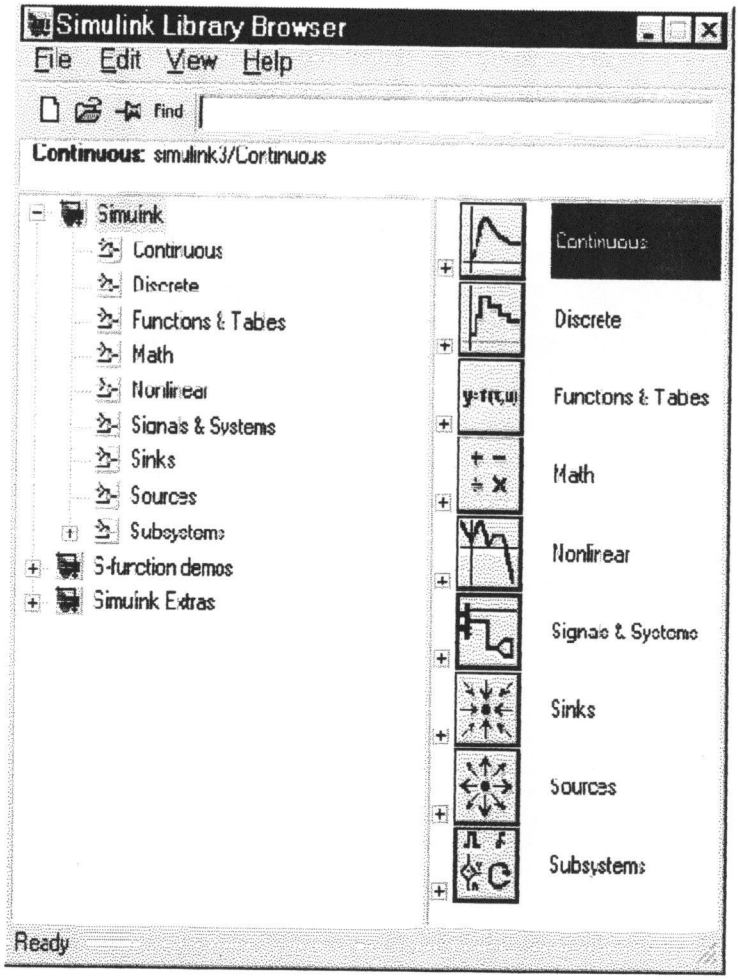
cuplik, atau gabungan dari keduanya. Sistem tersebut juga dapat bersifat *multirate*, yakni mempunyai bagian yang berbeda waktu cupliknya.

Untuk permodelan suatu sistem, *Simulink* menyediakan GUI (*Graphical User interface)* untuk membangun model sebagai diagram blok, dengan menggunakan operasi mouse *click-and-drag*. *Simulink* menyediakan *block library* yang lengkap meliputi *sinks* (keluaran), *sources* (masukan), komponen linier dan nonlinier dan konektor. *Simulink* juga menyediakan fasilitas untuk membuat blok yang didefinisikan sendiri oleh pengguna yakni melalui *s-function*.

Model bersifat hirarkis, sehingga kita dapat membangun model secara *top-down* maupun *bottom up*. Kita dapat melihat sistem pada level tinggi, lalu *double-click* pada blok untuk melihat isi di bawah blok tersebut secara lebih detail. Hal ini memungkinkan kita melihat bagaimana model itu diorganisasikan dan bagaimana setiap bagiannya saling berinteraksi.

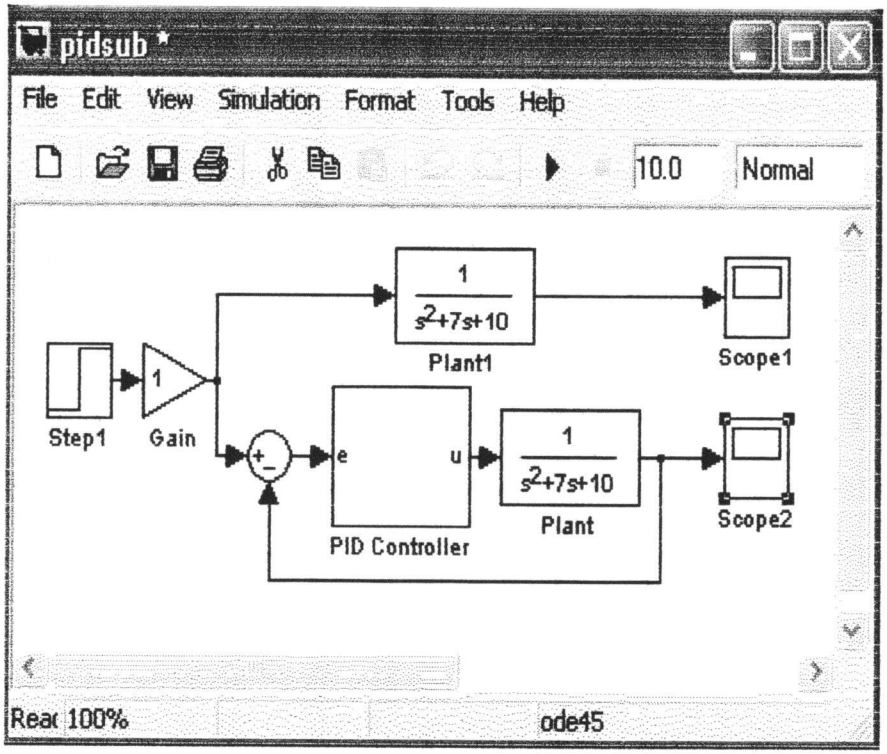
Setelah kita mendefinisikan suatu model, kita dapat melakukan simulasi terhadap model itu, menggunakan metode integrasi yang disediakan. Dengan *scope* atau blok *display* lainnya kita dapat melihat hasil simulasi saat simulasi sedang berjalan dan kita juga dapat mengubah parameter dan melihat pengaruhnya. Hasil simulasi juga dapat diletakkan di Matlab *workspace* untuk keperluan analisa berikutnya dan visualisasi karena Matlab dan *Simulink* terintegrasi maka kita dapat melakukan simulasi, analisa, dan revisi model pada berbagai lingkungan kerja Matlab.

Masuk ke Matlab *Simulink* dengan cara meng-klik tombol *Simulink* di *toolbar* atau ketik *Simulink* di *command window*, akan terbuka *editor Simulink* seperti pada gambar 2.6.



Gambar 2.6 Jendela *Simulink Library Browser*

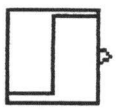
Untuk membuat model caranya pilih ***file, new, model***, kemudian akan terbuka sebuah *editor* untuk menempatkan blok-blok model seperti pada gambar 2.7 di bawah ini.



Gambar 2.7 Tampilan *editor Simulink* untuk membuat suatu model

Blok-blok *Simulink* yang digunakan adalah sebagai berikut :

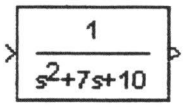
- *Step*



Step adalah suatu *step* masukan u(t), nilainya dapat di tentukan sendiri oleh *user*.

- *Gain*

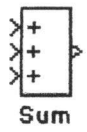
Gain adalah blok penguat yang memiliki nilai keluaran tetap

- *Transfer Fcn* 

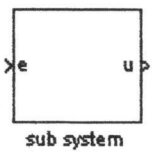
Transfer Fcn adalah sebuah sistem yang berupa fungsi alih terdiri dari numerator untuk pembilang dan denominator untuk penyebut.

- *Mux* 

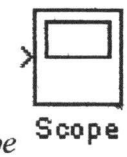
Mux blok ini mengkombinasikan beberapa *input* menjadi satu *output*, keluaran blok ini tergantung masukannya.

- *Sum* 

Sum adalah blok penjumlahan atau pengurangan *input* sesuai dengan tanda operasinya dan blok ini bisa mempunyai lebih dari satu *input* tetapi harus mempunyai dimensi yang sama.

- *Sub sistem* 

Sub sistem di dalamnya terdapat struktur jaringan yang di buat secara manual sesuai dengan keinginan kita

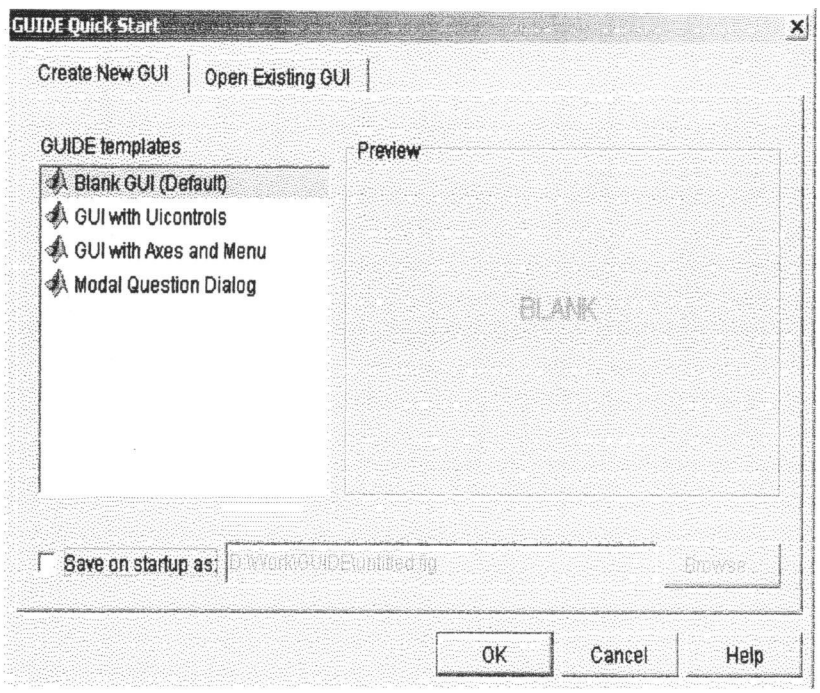
- *Scope* 

Scope merupakan tampilan keluaran yang dapat diamati oleh *user*.

2.3 *Graphical User Interface* (GUI)

GUI (*Graphical User Interface*) adalah sebuah *tool* yang disediakan di dalam Matlab. GUI adalah sebuah alat penghubung untuk membuat sebuah objek *grafis* / gambar di dalam Matlab, seperti tombol, bidang text, sliders, menu dan lain sebagainya. Penggunaan GUI untuk mengontrol suatu sistem lebih mudah dipelajari dan digunakan, karena penggunanya tidak perlu mengetahui baris perintah untuk mengatur / mengendalikan suatu sistem.

Matlab menyediakan *tool* untuk pembuatan GUI (*Graphical User Interface*). Fasilitas ini dinamakan *GUIDE* (GUI *Development Environment*). Untuk pembuatan GUI, ketikkan perintah *guide* di *command window*. Perintah ini akan menampilkan *GUIDE Quick Start.*

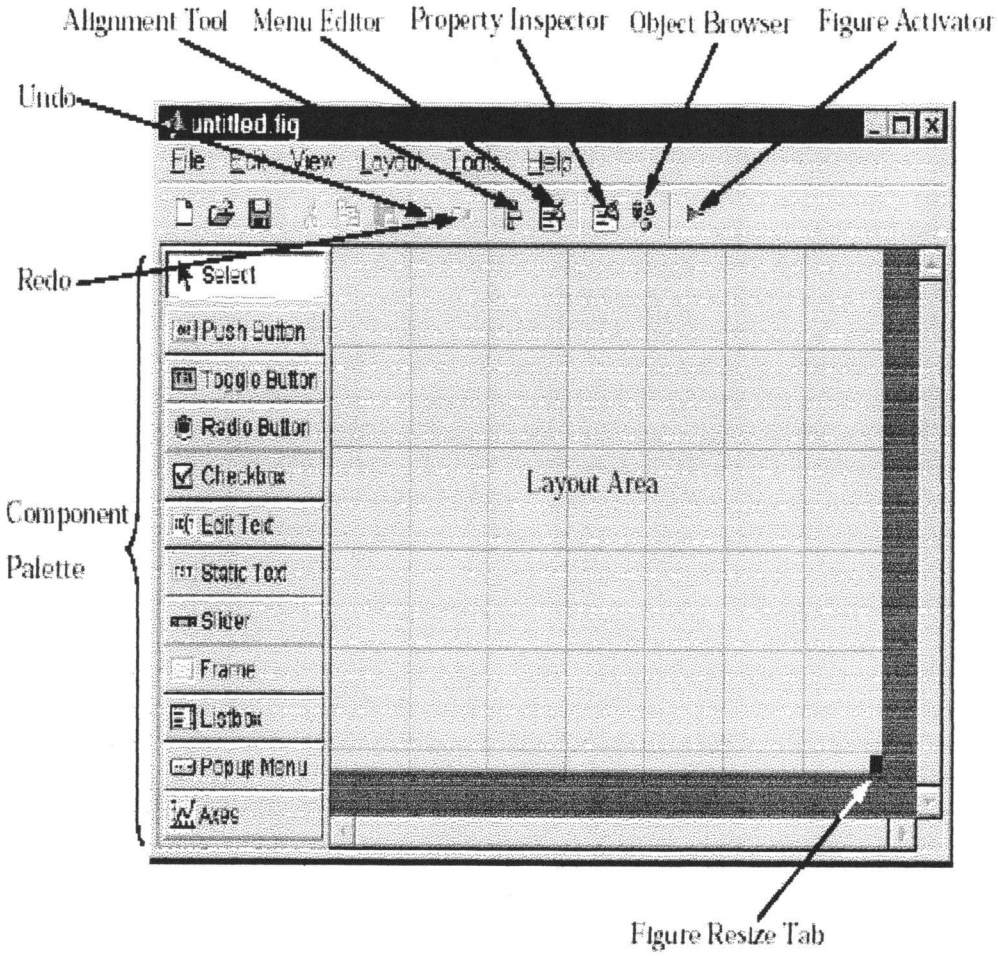


Gambar 2.8 Tampilan *GUIDE Quick Start*

Dari *GUIDE Quick Start* dialog kita dapat :

1. Membuat GUI baru dari *GUIDE templates*. GUI yang sudah tersedia yang dapat dimodifikasi untuk keperluan kita sendiri dapat dipilih melalui menu *Create new GUI.*
2. Membuka GUI yang ada atau GUI yang pernah di buat dapat dibuka kembali melalui *open existing* GUI.

Jika anda sudah memilih salah satu, klik 'OK' akan membuka GUI di *Layout Editor*. Pada layout area dapat diletakkan panel-panel yang sudah disediakan pada *component palette* dengan cara *drag and drop.*



Gambar 2.9 Jendela *Toolset*

GUIDE Toolset :

- *Layout Editor* : Tempat untuk menambah dan menyusun objek di jendela GUI.
- *Alignment Tools* : Untuk mengatur posisi masing-masing objek.
- *Component Pallet* : Komponen yang di sediakan di dalam GUI.
- *Property Inspector* : Memeriksa dan mengeset nilai-nilai (sifat) objek.
- *Object Browser* : Memantau hirarki objek-objek yang telah ada di *layout.*
- *Menu Editor* : Membuat menu dan submenu dari GUI.

Komponen *Palatte* GUI :

- *Push Button*

  Push Button menghasilkan suatu tindakan ketika di-klik. Sebagai contoh suatu tombol OK mungkin akan menutup suatu dialog kotak ketika tombol OK di tekan, tombol akan nampak ditekan dan *callback*nya akan melaksanakan perintah.

- *Toggle Button*

  Tombol *Toggle button* akan menghasilkan suatu tindakan ketika diklik suatu tombol kotak akan nampak tertekan, dan ketika diklik lagi akan melepaskan kotak kembali normal dan masing-masing mempunyai *callback.*

- *Radio Button*

  Kita dapat berkomunikasi dengan komponen *radio button* sama seperti *toggle button*. Jika komponen *radio button* diberi tanda maka nilai akan diset 1, sebaliknya jika tidak diberi tanda maka akan bernilai 0.

- *Check Boxes*

  Sama seperti komponen *radio button*.

- *Edit Text*

  Dalam penggunaannya, sebagian besar komponen e*dit text* banyak difungsikan sebagai komponen untuk menginputkan suatu data.

- *Slider*

  Komponen ini berfungsi memberikan *input* angka dengan suatu range tertentu. Properti *MAX-MIN* menunjukkan *range* yang digunakan oleh komponen *slider*.

- *List Boxes*

  List boxes lebih banyak digunakan untuk menampilkan suatu indeks data. Tiap baris dalam *list boxes* memiliki nilai dalam *property value*, sehingga kita dapat mengakses variabel di dalam *list boxes* menggunakan perintah *get*. Untuk menentukan range baris, di set menggunakan *MIN-MAX*.

- *Menu pop up*

  Kita dapat memprogram *menu pop-up callback* dengan mengambil indeks dari nilai yang dipilih. *Callback* ini akan memeriksa indeks dari item yang dipilih dan men-switch sesuai dengan nilai yang diperoleh.

- *Axes*

  Axes adalah komponen yang berfungsi sebagai *interface* penampil grafik baik berupa koordinat maupun gambar.

Dalam membuat GUI, proses yang dilakukan adalah :

- Perancangan tampilan GUI, akan lebih baik jika tampilan GUI digambar terlebih dahulu dalam kertas, penempatan dan pengaturan objek seperti *pushbutton, axis* dll.
- Pemrograman GUI, membuat program bagi objek-objek GUI di *M-file.*

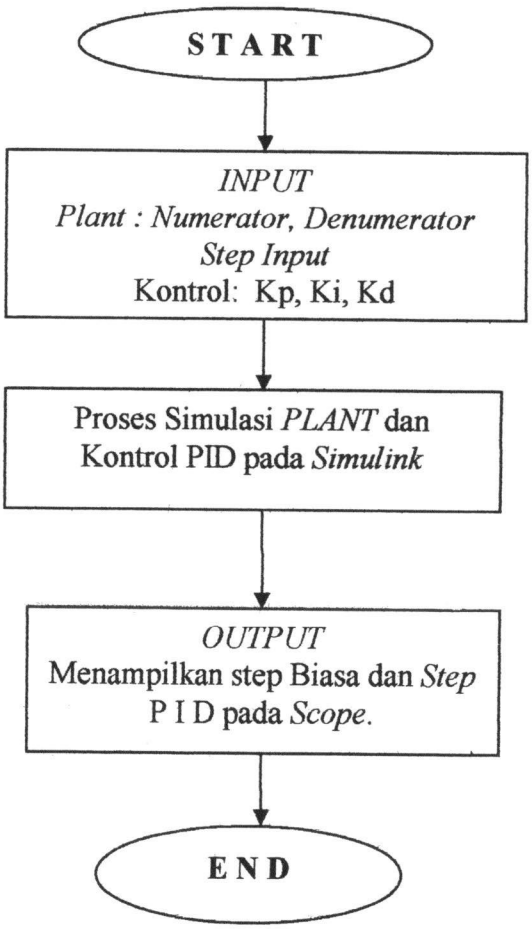
Ketika kita membuat *layout* GUI kemudian menyimpannya, maka Matlab akan membuat *M-file* secara otomatis yang nantinya akan digunakan untuk pemrograman. Nama *M-file* harus sama dengan nama *figure* dari GUI tersebut.

BAB III

PERANCANGAN SISTEM

Dalam bab III akan dibahas mengenai perancangan sistem pengendali PID dengan GUI Matlab dan program yang menghubungkan antara sistem tersebut dengan *Simulink* sehingga keduanya saling berinteraksi berdasarkan teori-teori yang telah dibahas pada bab sebelumnya.

Flowchart



Gambar 3.1 *Flowchart* jalannya program GUI

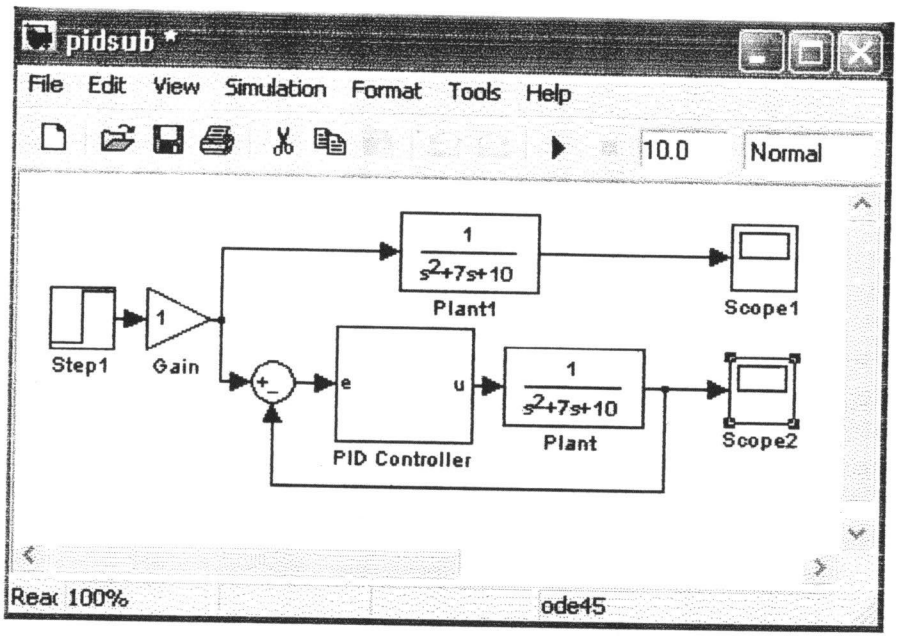
3.1 Perancangan Sistem pada *Simulink*

Membuat suatu sistem pengendali PID yang mengendalikan suatu *Plant* pada *Simulink* caranya pertama buka *software* Matlab kemudian pilih *file, new, model* setelah itu *model windows* akan terbuka untuk menempatkan blok-bok model.

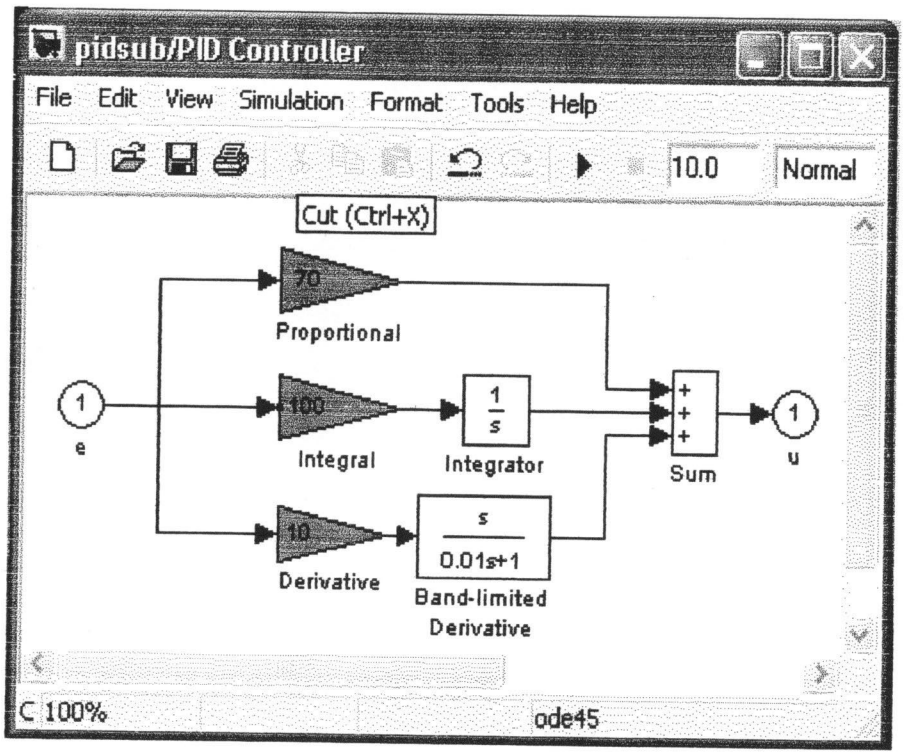
Untuk membuat sebuah sistem kita perlu mengkopi blok model yang kita butuhkan dari *Simulink Blocks Library*. Blok yang diperlukan adalah :

- *Step* sebagai input.
- *Sum* untuk melaksanakan penambahan atau pengurangan pada masukannya.
- *Transfer Fcn* sebagai *Plant*.
- *Scope* sebagai tampilan output.
- *Subsystem* untuk membuat suatu sistem pengendalian PID.
- *Gain, integrator Transfer Fcn* untuk membuat Pengendali PID.

Blok-blok ini kita peroleh dari *Library Browser*, jika tidak tahu letak bloknya maka baiknya dicari lewat kotak *search* atau menu *find*. Setelah blok-bloknya ditemukan maka blok tersebut didrag dari *Simulink Browser* ke model window kemudian port-portnya dihubungkan, setelah itu akan diperoleh gambar sebagai berikut.

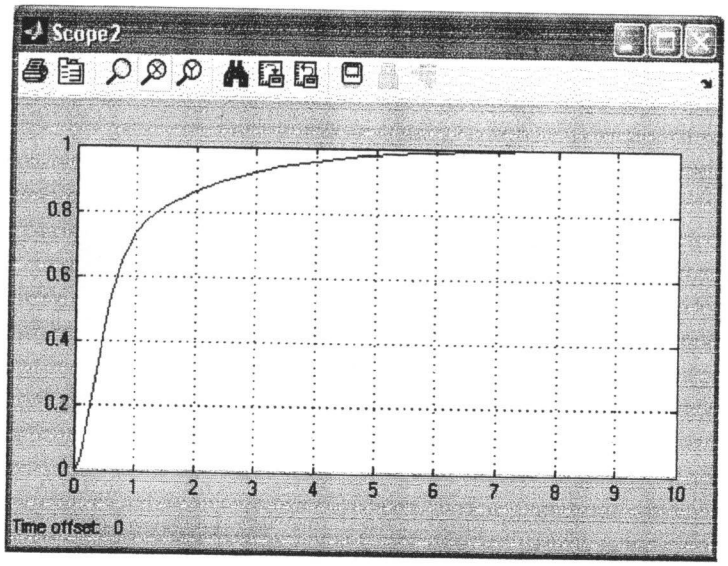


Gambar 3.2 Pidsub pada *Simulik*



Gambar 3.3 Sub sistem PID *Controller* pada Pidsub *Simulink*

Setelah semua konektor dihubungkan maka kita siap melakukan simulasi dengan cara menekan tombol *run* dan melihat hasilnya dengan cara mengklik dua kali pada blok *Scope* seperti pada Grafik 3.1 di bawah ini.



Grafik 3.1 Hasil simulasi pada *Scope*

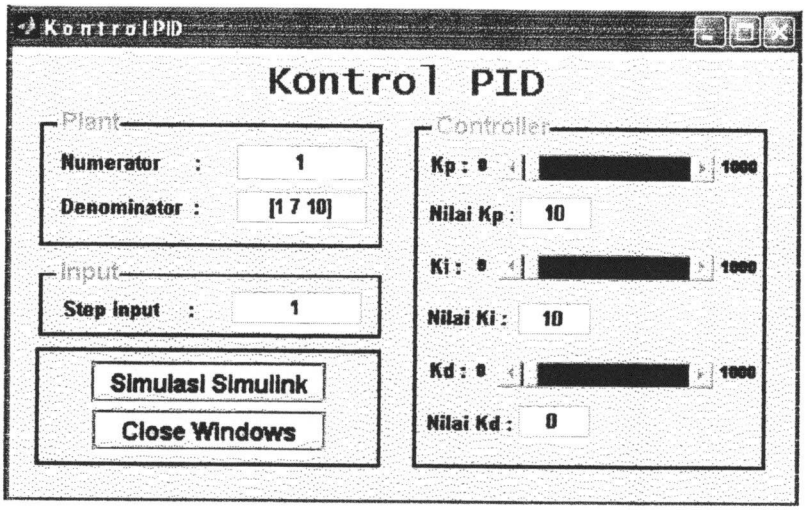
3.2 Perancangan GUI Matlab.

Disini akan dibuat suatu tampilan GUI yang komponennya dapat mengatur parameter PID, memasukan nilai input dan step input serta menjalankan simulasi pada *Simulink*.

Pada pengaturan pengendali PID yang terdiri dari Kp sebagi *Proportional*, Ki sebagai *Integral*, dan Kd sebagai *Derivative* atau Turunan dapat dilakukan dengan dua cara :

1. Mengetikan besarnya nilai Kp, Ki, Kd yang kita inginkan pada menu edit
2. Menggeser *slider* Kp, Ki dan Kd sesuai dengan yang kita inginkan.

Untuk memasukan parameter *plant*nya dilakukan dengan cara mengisikan nilai *Numerator* dan *Denominator*nya.



Gambar 3.4 Desain Kontrol PID pada GUI

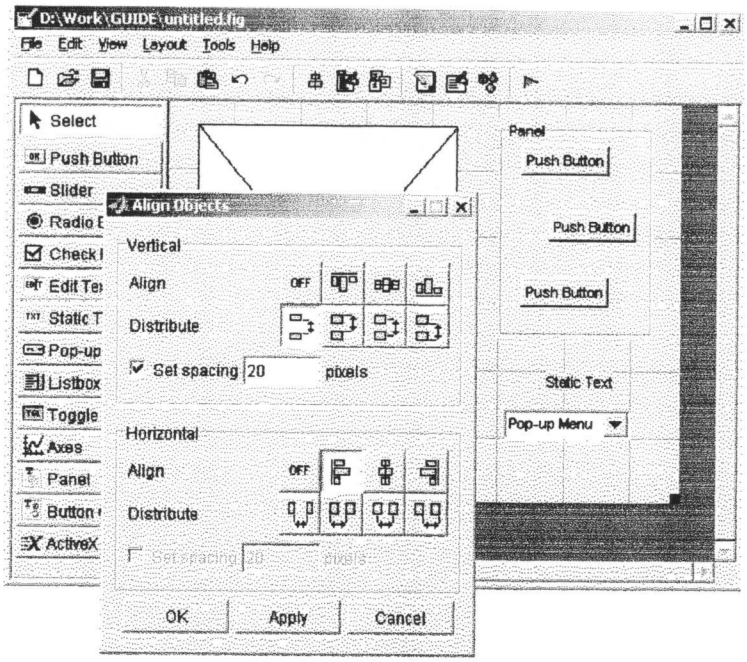
Tampilan GUI diatas dirancang agar dapat bekerja bersama dengan *Simulink* model, dan aturan option GUInya adalah sebagai berikut :

- *Resize behavior: Proportional.*
- *Command-line accessibility: Callback.*
- *M-file options selected : Generate callback function prototypes GUI allows only one instance to run.*

Komponen-komponen GUI yang digunakan adalah sebagai berikut :

- *Edit text* ada 6 : numw, denw, stepw, nilai Kpw, nilai Kiw, nilai Kdw.
- *Slider* ada 3 : *slider* Kpw, *slider* Kiw, *slider* Kdw.
- *Push button* ada 2 : *Simulasi Simulink* dan *Close Window*.
- *Group Panel* ada 3 : *Plant, Input* dan *Controller*.
- *Group Button* ada 1.

Untuk menambahkan komponen yang dibutuhkan ke *Layout Area* klik komponen yang di perlukan dari komponen *pallette*, kemudian *drag* dan letakan pada *Layout Area*. Setelah komponen-komponen tersebut disusun di dalam *Layout Area* kemudian kita rapikan penempatannya dengan *align object* dari *menu tools* dengan cara pilih komponen yang ingin dirapikan sambil menekan *Ctrl* kemudian lakukan pengaturan sesuai dengan yang dikehendaki.



Gambar 3.5 Tampilan *Align object*

Untuk mengeset *property* dan *Tag* dari tiap-tiap komponen pilih *Property Inspector* dari menu *View* atau klik dua kali pada komponen tersebut.

- *Static Text*

  Klik pada komponen *Static Text* kemudian isikan *String property* sesuai dengan nama yang kita inginkan dan pada *Tag property*nya di abaikan saja karena komponen ini tidak menghasilkan *callback* pada *M-file.*

- *Edit Text*

  Pada *Plant* ada dua *edit text, Numerator string*nya di isi '1', dan yang kedua *Denominator string*nya di isi '[1 7 10]' dan *Tag property*nya di beri nama numw untuk *Numerator*, denw untuk *Denominator*. Pada *Input* ada satu diberi nama stepw. Pada *Controller* ada tiga *edit text*. Nilai Kp *edit text string*nya di isi '10', Nilai Ki *edit text string*nya di isi '10' Nilai Kd *edit text string*nya di isi kosong ' ' dan Tag Kp *edit text*nya dikasih nama 'nilaiKpw', Ki diberi nama 'nilaiKiw' dan Kd diberi nama 'nilaiKdw'.

- *Slider*

  Slider yang di butuhkan ada tiga buah untuk mengatur nilai Kp, Ki dan Kd dan pada *Property Inspector* tag nya diberi nama *sliderKpw*, *sliderKiw*, dan *sliderKdw*, *string*nya di kasih nilai kosong ' ', *listbox/top* '0', *min* '0' dan *max* '1000' dan *value* untuk sliderKpw dan sliderKiw di beri nilai '10' sliderKdw '0'.

- *Push Button*

  Push button pertama *string*nya diberi nama *Simulasi Simulink* dan *tag*nya simulasiw yang kedua *string*nya diberi nama *Close Windows* dan tagnya diberi nama tutupw.

- *Panel*

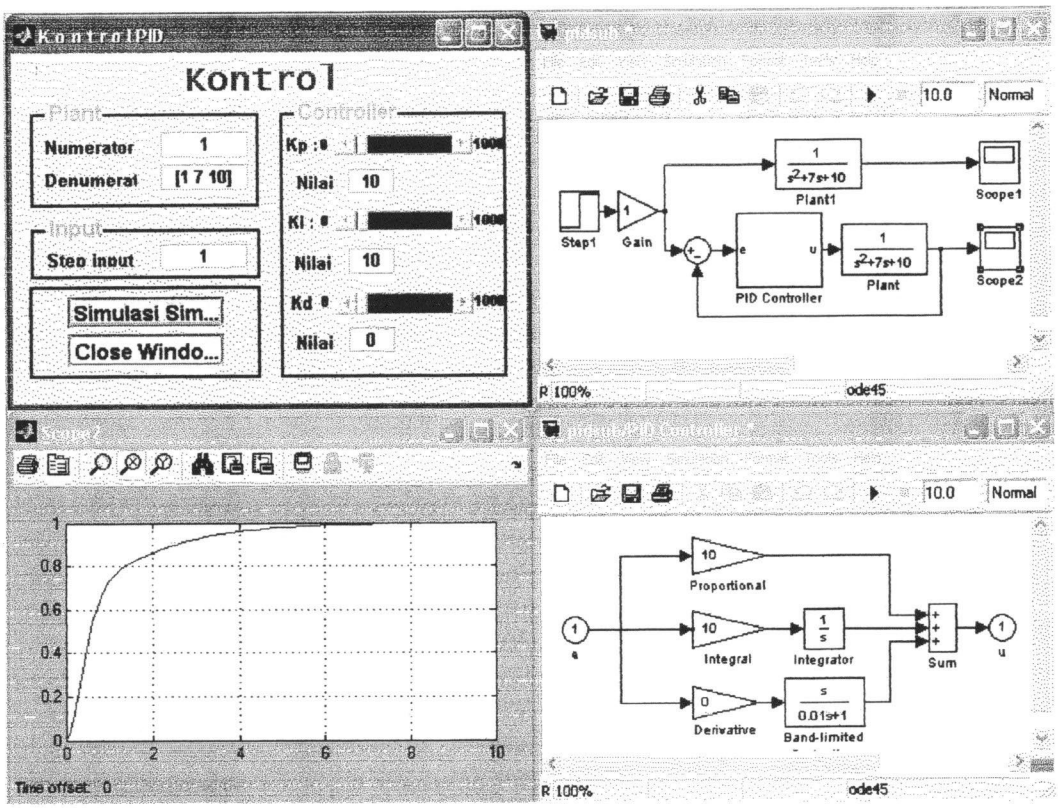
  Uipanel 1 *string*nya diberi nama *Plant*, *Uipanel* 2 *string*nya diberi nama *Controller*.

- *Button Group*

  Uipanel 3 stringnya idak usah di kasih.

Tag property berguna untuk mendiskripsikan nama fungsi yang akan di panggil. Setelah komponen selesai diatur nilainya kemudian disimpan dan diaktifkan dengan memilih *run*, dengan cara klik *icon run* atau pilih *tool, run* hal ini akan menghasilkan dua file, yaitu *file * fig (file figure)* dan *file * m (untuk pemrograman GUI)* biasa disebut dengan *M-File*. Komponen-komponen yang harus di program adalah *Edit Text, Slider* dan *Push Button*. Program lengkap dari komponen-komponen GUI Matlab terlampir pada lampiran.

Setelah melakukan pemrograman kita lakukan pengujian GUI dengan cara mengaktifkan GUI. Pengaktifan GUI melalui *M-file* dapat dilakukan dengan cara klik *icon run* atau tekan F5 atau pilih *debug run*. Hasil pengaktifan GUI yang telah dibuat akan menampilkan gambar sebagai berikut.



Gambar 3.6 Hasil dari pemrograman GUI

BAB IV

ANALISIS DAN PEMBAHASAN

Pada bab ini akan dilakukan pengujian sistem, materi pengujian meliputi pengujian *plant* yang menggunakan Orde Satu, Dua dan Orde Tiga serta pengendalian PID dengan mencoba-coba mengubah parameter-parameter Kp, Ki dan Kd yang ada pada GUI Matlab sampai menghasilkan keadaan yang diinginkan dengan memperhatikan karakteristik dari kontroler PID yang dapat dilihat pada Tabel 2.1.

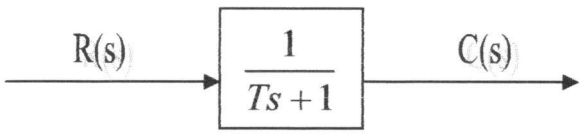
Step respon yang diamati adalah **Td** (waktu tunda), **Tr** (waktu naik), **Tp** (waktu puncak), dan **Mp** (*overshoot* maksimum) serta **Ts** (waktu tunak) dan **Ns** (nilai stabil) dan pengertiannya adalah sebagai berikut

- Waktu tunda, Td : waktu tunda adalah waktu yang di perlukan oleh tanggapan untuk mencapai setengah nilai akhir untuk waktu yang pertama satuannya *second* / detik (t).
- Waktu naik, Tr : waktu naik adalah waktu yang diperlukan oleh tanggapan untuk naik dari 10% menjadi 90% dari nilai akhir yang biasa digunakan dan satuannya second/detik (t).
- Waktu puncak, Tp : waktu puncak adalah waktu yang diperlukan tanggapan untuk mencapai puncak pertama *overshoot* satuannya second / detik (t).
- Maksimum *overshoot*, Mp : *Overshoot* maksimum adalah nilai puncak kurva tanggapan dan satuannya biasanya dalam persen (%).

- Waktu tunak, Ts : waktu tunak adalah waktu yang diperlukan untuk menanggapi kurva agar dapat mencapai dan tetap berada dalam gugus nilai akhir ukuran yang disederhanakan dengan persentase mutlak harga akhirnya (biasanya 2% atau 5%).
- Nilai stabil, Ns : nilai stabil adalah besarnya nilai amplitudo disaat keluaran sistem dalam keadaan tunak satuannya tergantung dari masukannya bisa volt (v) dll.

4.1 Pengujian Sistem Orde Satu

Sistem Orde Satu secara fisik dapat menyajikan rangkaian RC, sistem termal, atau sistem lainnya. Diagram blok yang disederhanakan dapat dilihat dari gambar 4.1.



Gambar 4.1 Diagram blok Orde Satu

Hubungan masukan dan keluarannya :

$$\frac{C(S)}{R(s)} = \frac{1}{Ts+1} \tag{4.1}$$

Plant Orde Satu berupa mesin DC yang mempunyai persamaan 4.2.

$$\frac{C(S)}{R(s)} = \frac{1}{s+10} \tag{4.2}$$

C(s) = Keluaran

R(s) = Masukan

s = Frekuensi

Langkah pertama untuk menganalisis adalah melihat *step respon loop* terbukanya terlebih dahulu, dengan cara mengisikan nilai yang dimiliki oleh sistem pada GUI, setelah hasil *step respon* didapatkan kemudian dilakukan pengendalian menggunakan kontrol PID untuk mendapatkan hasil *step respon* yang di inginkan yaitu *step respon* yang stabil menggunakan cara *trial and error* dengan memperhatikan karakteristik dari kontrol PIDnya

Fungsi *transfer loop* tertutup Orde Satu dengan kontroler PID adalah sebagai berikut :

$$\frac{C(S)}{R(s)} = \frac{K_D s^2 + K_P s + Ki}{(1+K_D)s^2 + (10+K_p)s + K_i} \tag{4.3}$$

4.1.1 Percobaan 1 Orde Satu

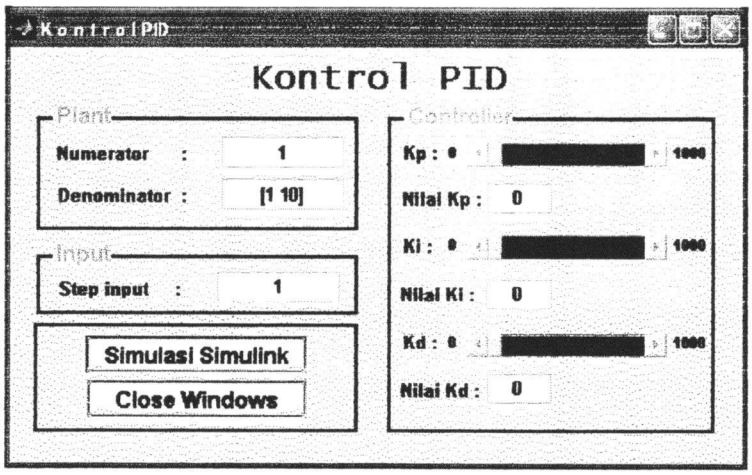
Pertama-tama melihat *step respon loop* terbuka dari sistem dengan cara memasukan nilai fungsi alih dari sistem Orde Satu ke dalam GUI Matlab.

Numerator : [1]

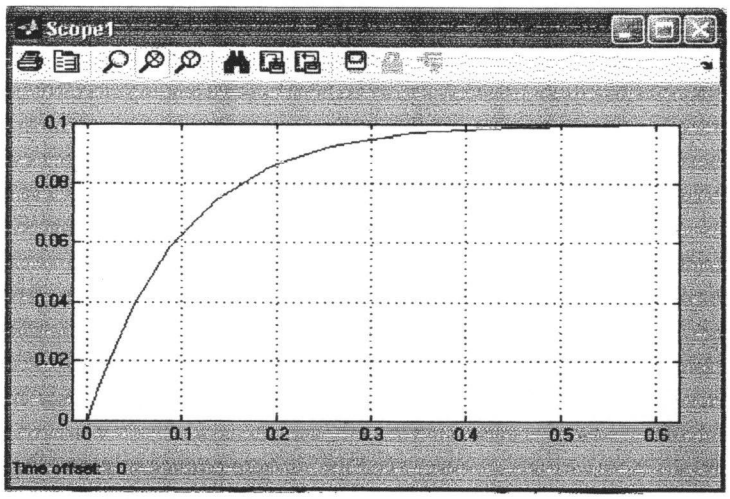
Denominator : [1 10]

Step input : [1]

Tekan tombol "Simulasi Simulink" untuk menjalankan simulasi dan lihat hasil *step respon*nya pada *scope* seperti yang terlihat pada grafik 4.1.



Gambar 4.2 Tampilan GUI Orde Satu



Grafik 4.1 Natural respon dari Orde Satu

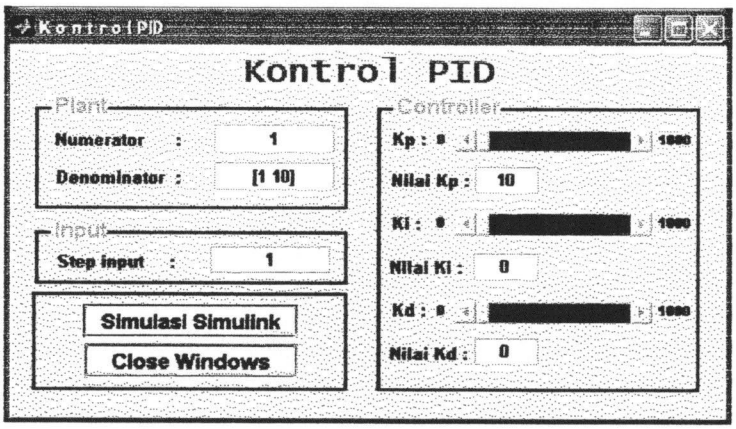
Tabel 4.1 Hasil percobaan 1 Orde Satu

Kp	Ki	Kd	Td	Tr	Tp	Mp	Ts	Ns
0	0	0	0.07	0.4	-	-	0.5	0.1

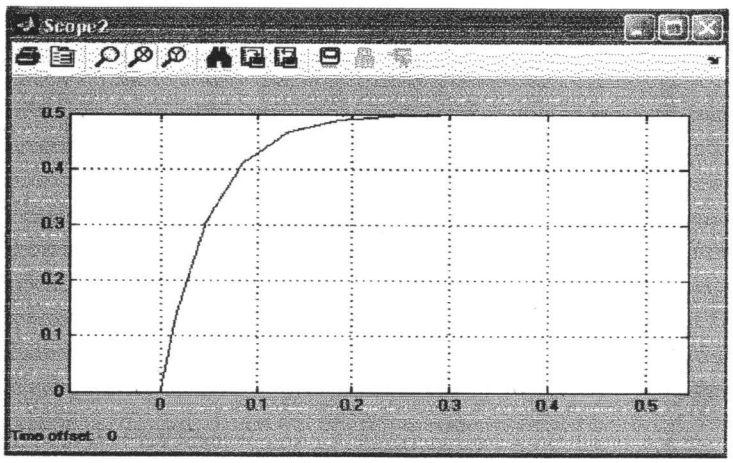
Pada Grafik 4.1 terlihat bahwa waktu tunda : 0.07s, waktu naik : 0.4s dan waktu tunaknya cukup cepat sekitar : 0.6s. Nilai akhir dari *step respon* dari sistem adalah 0.1 hal ini membuat kesalahan keadaan tunaknya sangat besar 0.9 sehingga perlu di lakukan pengendalian dengan kontrol PID agar memiliki respon yang stabil.

4.1.2 Percobaan 2 Orde Satu

Pada pecobaan ke 2 di sini dilakukan penguatan kontrol *proportional* / Kp : 10 dan hasil simulasinya seperti yang terlihat seperti pada grafik 4.2.



Gambar 4.3 Tampilan GUI Orde Satu dengan Kp : 10



Grafik 4.2 *Step respon* Orde Satu dengan kontroler P

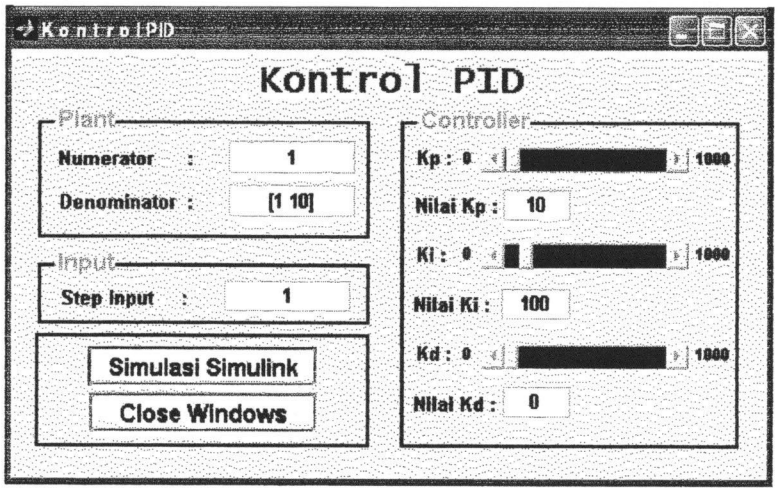
Tabel 4.2 Hasil percobaan 2 Orde Satu

Kp	Ki	Kd	Td	Tr	Tp	Mp	Ts	Ns
10	0	0	0.04	0.2	-	-	0.3	0.5

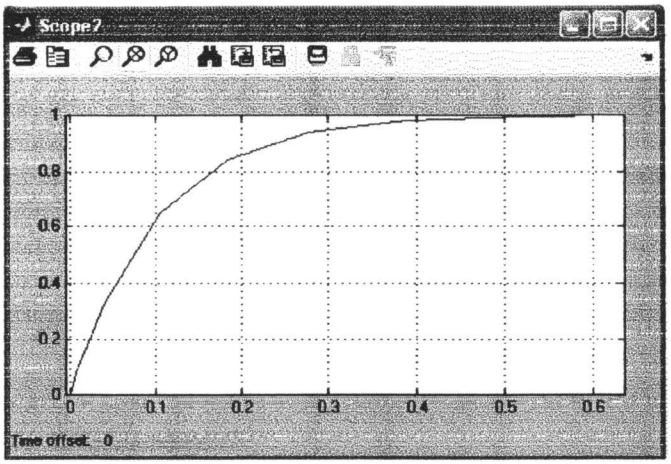
Setelah di lakukan simulasi ternyata sistem menghasilkan waktu naik 0.2s dan waktu tunak 0.3s dengan hasil nilai stabil 0.5 hal ini membuat kesalahan keadaan tunaknya juga 0.5. Pengaruh dari kontrol *proportional* mengurangi kesalahan keadaan tunaknya dan juga waktu tunaknya.

4.1.3 Percobaan 3 Orde Satu

Percobaan 3 menggunakan kontrol *proportional* : 10 dan *integral* : 100. Hasil simulasi sistem ternyata memberikan respon yang diinginkan seperti yang terlihat pada grafik 4.3.



Gambar 4.4 Tampilan GUI Orde Satu dengan Kp : 10 dan Ki : 100



Grafik 4.3 *Step respon* Orde Satu dengan pengutan P dan I

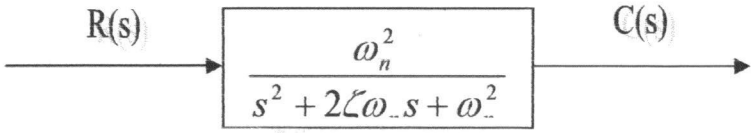
Tabel 4.3 Hasil percobaan 3 Orde Satu

Kp	Ki	Kd	Td	Tr	Tp	Mp	Ts	Ns
10	100	0	0.07	0.4	-	-	0.5	1

Sekarang di dapatkan *step respon* dengan waktu naik 0.4 s dan waktu tunak 0.5 s sudah mencapai keadaan yang cukup stabil, tanpa adanya *overshoot* dan kesalahan keadaan tunak. Akhirnya penerapan ketiga kontrol PID pada sistem ini tidak perlu dilakukan karena penggunaan kontrol PI saja sudah memberikan respon yang cukup baik.

4.2 Pengujian Sistem Orde Dua

Orde suatu sistem dapat ditentukan dengan rangkaian atau persamaan geraknya, berupa persamaan differensial ataupun fungsi alih. Orde Dua suatu sistem merupakan parameter yang menentukan karakteristik sistem tersebut dan diagram bloknya seperti pada gambar 4.5.



Gambar 4.5 Diagram Blok Orde Dua

Hubungan masukan dan keluarannya adalah sebagai berikut :

$$\frac{C(S)}{R(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{4.4}$$

C(s) = Keluaran

R(s) = Masukan

ϖ = Frekuensi osilasi pada saat ζ sama dengan nol

ζ = Damping ratio

s = Frekuensi

Misal *transfer function* sistem pegas yang akan dicoba adalah sebagai berikut :

$$\frac{C(S)}{R(s)} = \frac{1}{s^2 + 10s + 20} \tag{4.5}$$

Fungsi *transfer loop* tertutup dari sistem Orde Dua dengan PID kontroler adalah sebagai berikut:

$$\frac{C(S)}{R(s)} = \frac{K_D s^2 + K_P s + Ki}{s^3 + (10 + K_D)s^2 + (20 + K_p)s + K_i} \tag{4.6}$$

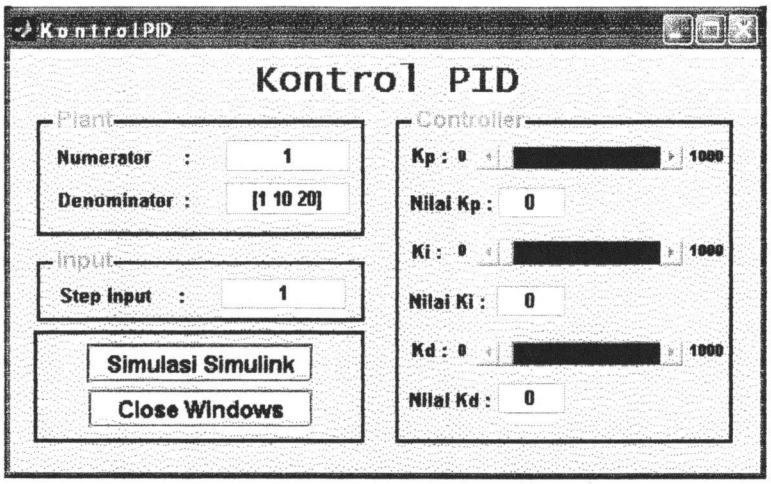
4.2.1 Percobaan 1 Orde Dua

Pertama-tama melihat *step respon loop* terbuka dari sistem dengan cara memasukan nilai fungsi alih dari sistem Orde Dua ke dalam GUI Matlab seperti yang terlihat pada gambar 4.6.

Numerator : [1]

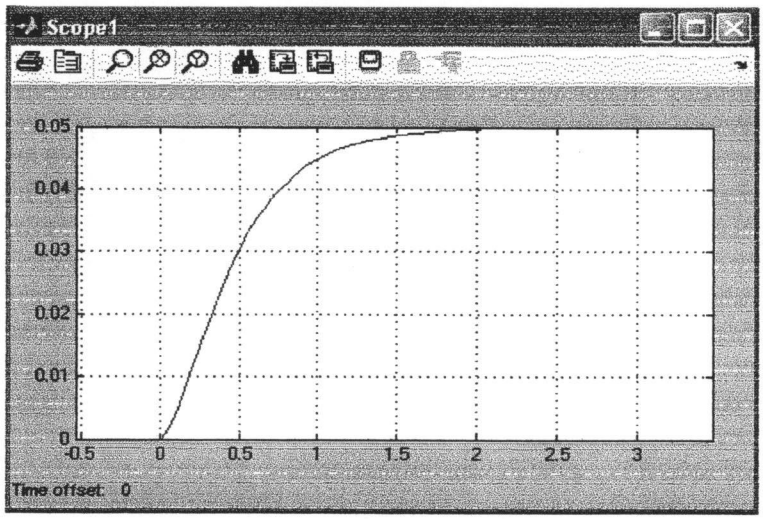
Denominaator : [1 10 20]

Step input : [1]



Gambar 4.6 Tampilan GUI Orde Dua

Tekan tombol 'Simulasi Simulink' untuk menjalankan simulasi dan lihat hasil *step respon*nya pada scope1 seperti yang terlihat pada grafik 4.4.



Grafik 4.4 *Natural respon* dari Orde Dua

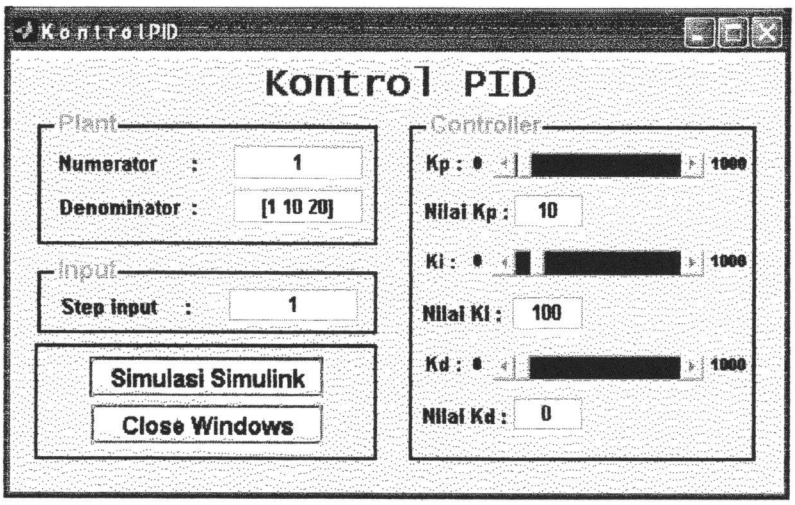
Tabel 4.4 Hasil percobaan 1 Orde Dua

Kp	Ki	Kd	Td	Tr	Tp	Mp	Ts	Ns
0	0	0	0.4	1.5	-	-	2	0.05

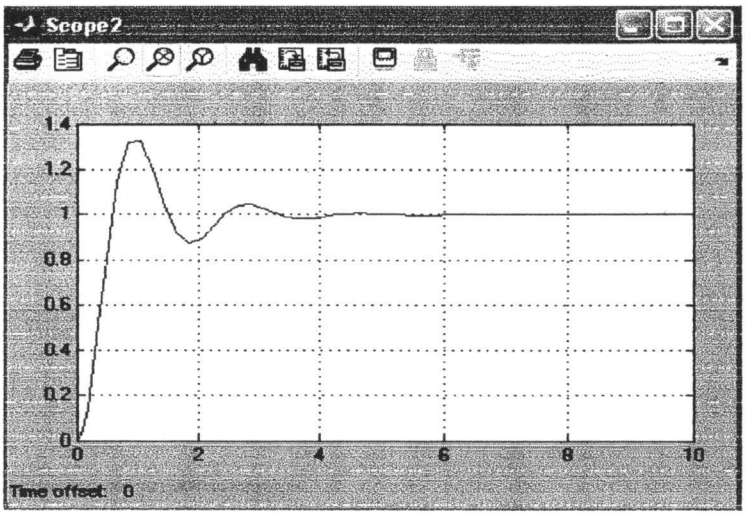
Hasil dari *step respon* terlihat waktu naik sekitar 1.5s dan waktu tunak 2s dan nilai stabil dari keluaran adalah 0.05 hal ini membuat kesalahan keadaan tunaknya 0.95 sangat besar, *natural respon* dari sistem tidak menimbulkan *overshoot*.

4.2.2 Percobaan 2 Orde Dua

Pada percobaan 2 dilakukan penggunaan kontrol *proportional* / Kp : 10 dan *integral* / Ki : 100 pada GUI Matlab seperti yang terlihat pada Gambar 4.7 dan setelah simulasi dijalankan dengan menekan tombol "Simulasi Simulink" di dapatkan respon seperti yang terlihat pada grafik 4.5.



Gambar 4.7 Tampilan GUI percobaan 2 Orde Dua dengan Kp:10 dan Ki:100



Grafik 4.5 *Step respon* Orde Dua dengan kontroler P dan I

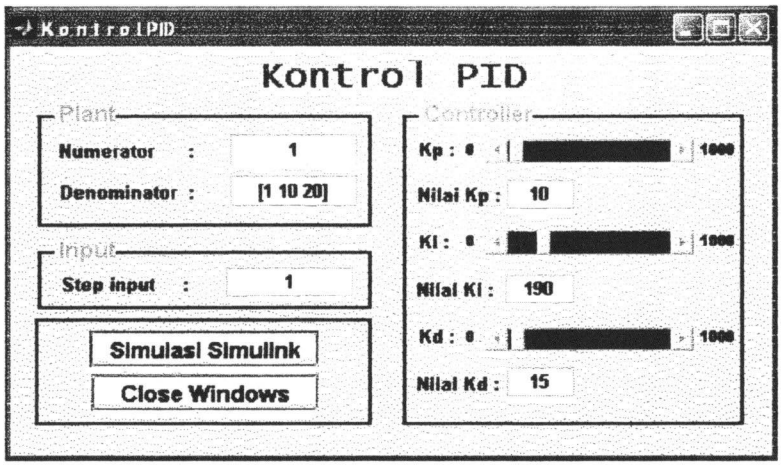
Tabel 4.5 Hasil percobaan 2 Orde Dua

Kp	Ki	Kd	Td	Tr	Tp	Mp	Ts	Ns
10	100	0	0.07	0.6	1	30	5	1

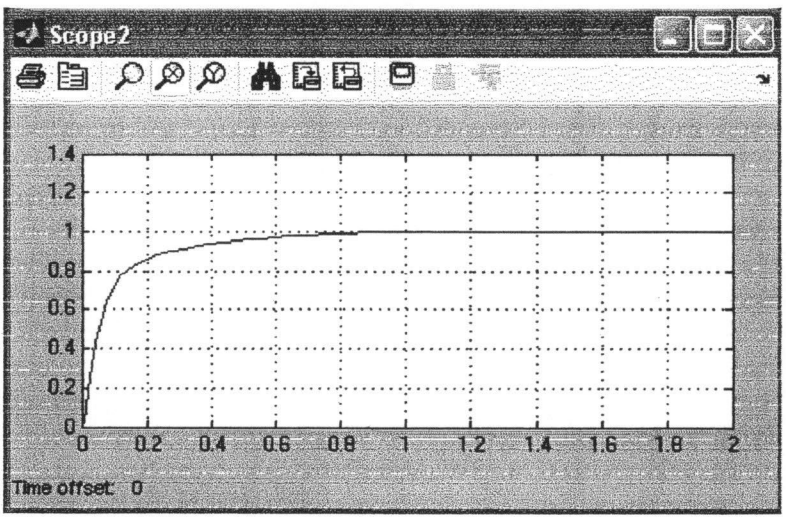
Pada grafik 4.5 waktu naiknya 0.6s dan terjadi *overshoot* 30% dengan waktu puncak 1s dan nilai puncak 1.3 sehingga membuat waktu tunaknya cukup lama yaitu 5s. *plot* di atas memperlihatkan bahwa kontrol *proportional* dan *integral* mereduksi waktu naik dan kesalahan keadaan tunak akan tetapi meningkatkan *overshoot* dan menambah waktu tunak dalam skala kecil.

4.2.3 Percobaan 3 Orde Dua

Kontroler yang digunakan adalah *proportional* / Kp: 110, *integral* / Ki: 190 dan *derivative* / Kd : 15 tampilan GUInya seperti pada gambar 4.8 dan hasil simulasinya pada grafik 4.6.



Gambar 4.8 Tampilan GUI percobaan 3 Orde Dua dengan Kp : 110, Ki : 190 dan Kd : 15



Grafik 4.6 *Step respon* Orde Dua dengan kontroler PID

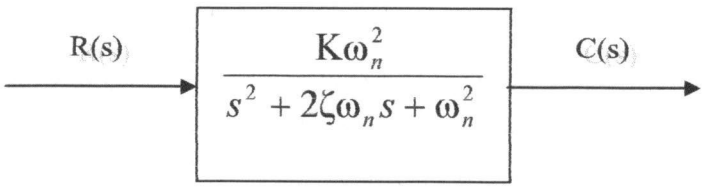
Tabel 4.6 Hasil percobaan 3 Orde Dua

Kp	Ki	Kd	Td	Tr	Tp	Mp	Ts	Ns
100	190	15	0.05	0.5	-	-	0.8	1

Percobaan 3 menghasilkan *step respon* yang cukup stabil dengan waktu naik yang cukup cepat : 0.5 dan waktu tunak : 0.8 tanpa adanya *overshoot* dan kesalahan keadaan tunak.

4.3 Pengujian sistem Orde Tiga

Blok diagram sistem Orde Tiga:



Gambar 4.9 Diagram blok Orde Tiga

Fungsi alih *loop* tertutupnya adalah

$$\frac{C(S)}{R(s)} = \frac{K\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{4.7}$$

C(s) = Keluaran

R(s) = Masukan

ϖ = Frekuensi osilasi pada saat ζ sama dengan nol

ζ = Damping ratio

s = Frekuensi

K = Bati DC

Misal *transfer function* sistem batang kendali pada pesawat adalah :

$$\frac{C(S)}{R(s)} = \frac{1.15s + 0.17}{s^3 + 0.73s^2 + 0.92s} \tag{4.8}$$

Fungsi *transfer loop* tertutup dari sistem Orde Tiga dengan PID kontroler adalah sebagai berikut :

$$\frac{C(S)}{R(s)} = \tag{4.9}$$

$$\frac{1.15K_D s^3 + (1.15K_P + 0.17K_D)s^2 + (1.51Ki + 0.17K_P)s + 0.17K_I}{s^4 + (0.73 + 1.51K_D)s^3 + (0.92 + 1.51K_p + 0.17K_D)s^2 + (1.51K_{Ii} + O.17K_P)s + 0.17K_I}$$

4.3.1 Percobaan 1 Orde Tiga

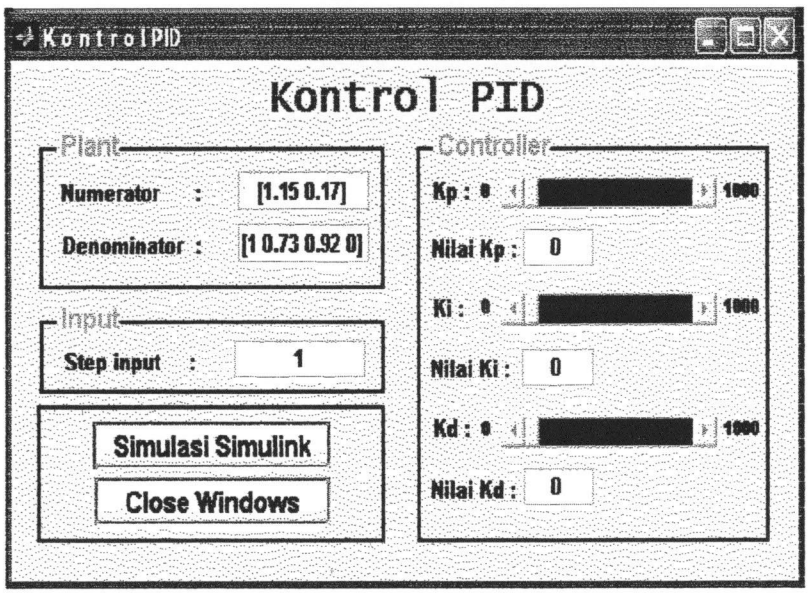
Sistem Orde Tiga dari sistem di masukan ke dalam GUI Matlab untuk melihat *loop* terbuka atau *natural respon* dari sistem.

Numerator : [1.15 0.17]

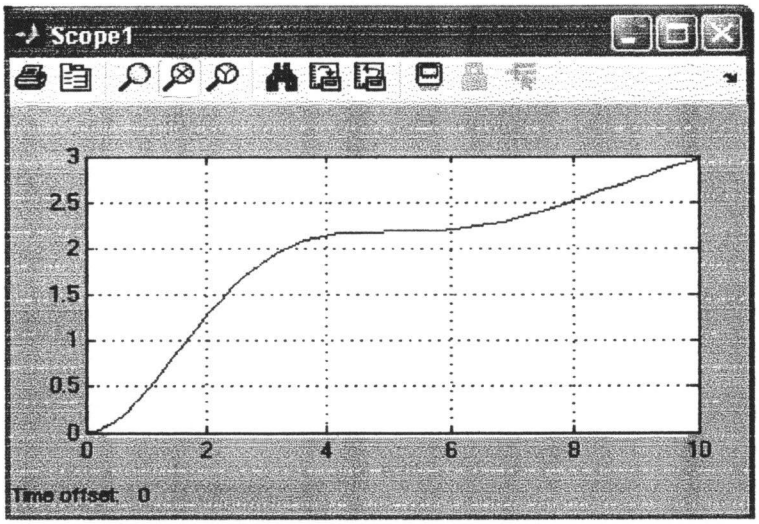
Denominator : [1 0.73 0.92 0]

Step input : [1]

Tampilan dari GUI Matlabnya dapat di lihat pada gambar 4.10 dan hasil simulasinya dapat di lihat pada grafik 4.7.



Gambar 4.10 Tampilan GUI Orde Tiga



Grafik 4.7 *Natural respon* dari Orde Tiga

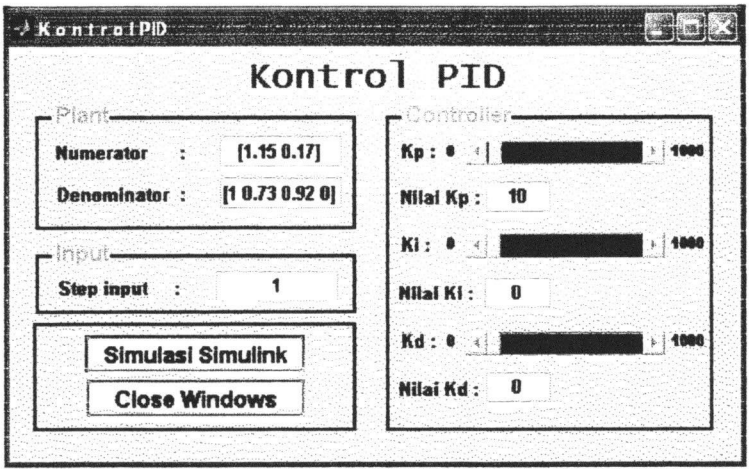
Tabel 4.7 Hasil percobaan 1 Orde Tiga

Kp	Ki	Kd	Td	Tr	Tp	Mp	Ts	Ns
0	0	0	~	~	~	~	~	~

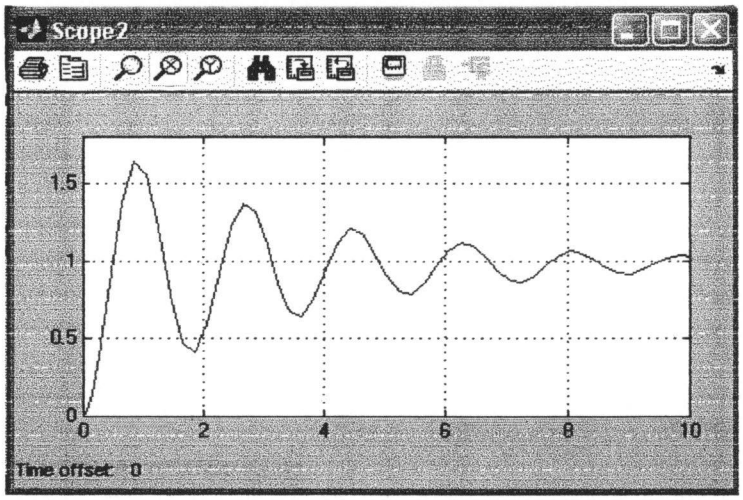
Step respon dari Orde Tiga ini ternyata tidak terkendali dengan waktu naik dan waktu tunaknya tidak dapat di tentukan.

4.3.2 Percobaan 2 Orde Tiga

Pada percobaan ini menggunakan kontrol *proportional* / Kp : 10. Tampilan GUI Matlabnya dapat di lihat pada gambar 4.11 dan keluarannya pada grafik 4.8 di bawah ini.



Gambar 4.11 Tampilan GUI percobaan 2 Orde Tiga dengan Kp:10



Grafik 4.8 *Step respon* Orde Tiga dengan kontroler P

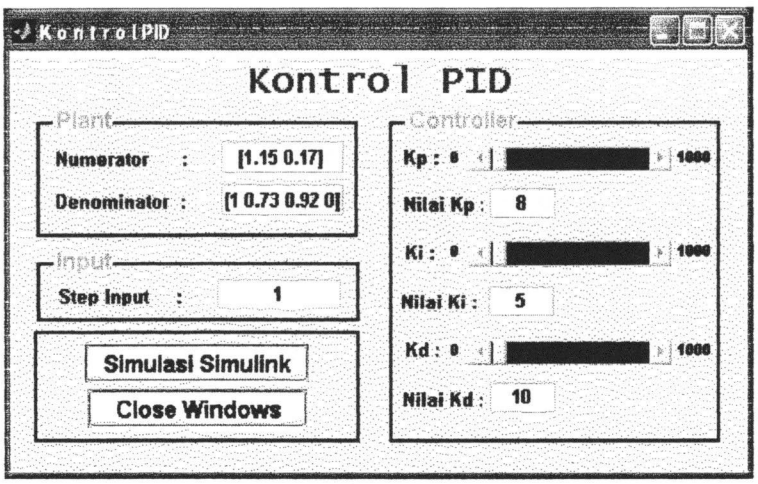
Tabel 4.8 Hasil percobaan 2 Orde Tiga

Kp	Ki	Kd	Td	Tr	Tp	Mp	Ts	Ns
10	0	0	0.3	0.4	0.85	60	16	1

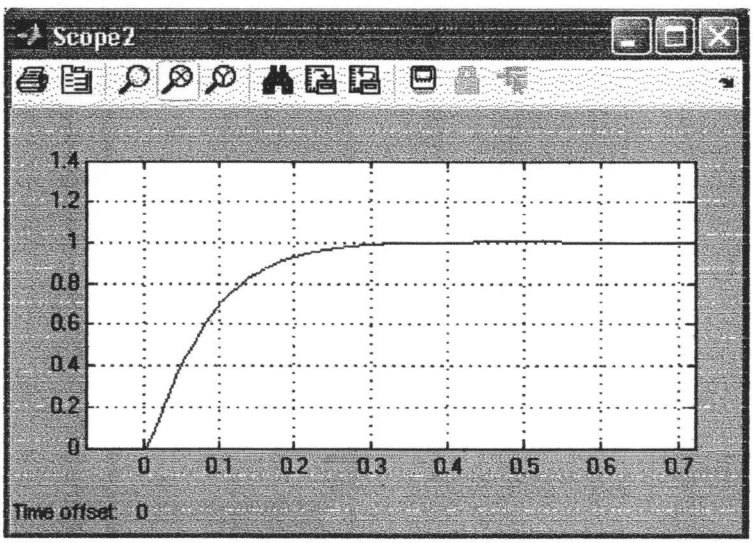
Pengendalian *proportional* menghasilkan respon yang buruk dengan waktu tunak yang sangat lama dan *maximum overshoot* 60%.

4.3.3 Percobaan 3 Orde Tiga

Kontroler yang digunakan adalah *proportional* / Kp : 8, *integral* / Ki : 5 dan *derivative* / Kd : 10 tampilan GUInya seperti pada gambar 4.12 dan keluarannya dapat di lihat pada grafik 4.9.



Gambar 4.12 Tampilan GUI percobaan 3 Orde Tiga dengan Kp : 8, Ki : 5 dan Kd : 10



Grafik 4.9 *Step respon* Orde Tiga dengan kontroler PID

Tabel 4.9 Hasil percobaan 3 Orde Tiga

Kp	Ki	Kd	Td	Tr	Tp	Mp	Ts	Ns
8	5	10	0.06	0.2	-	-	0.31	1

Tampak pada grafik 4.9 kesalahan keadaan tunaknya dapat dieliminasi dengan pengendalian kontroler PID dengan waktu naik 0.2s dan waktu tunak 0.31 tanpa adanya *overshoot.*

4.4 Pengujian sistem Orde Tinggi

Transfer function sistem Orde Tinggi misal adalah sebagai berikut :

$$\frac{C(S)}{R(s)} = \frac{1s^2 + 2s + 10}{1s^4 + 2s^3 + 3s^2 + 10s + 20} \tag{4.10}$$

C(s) = Keluaran

R(s) = Masukan

s = Frekuensi

Fungsi *transfer loop* tertutup dari sistem Orde Tinggi dengan PID kontroler adalah sebagai berikut :

$$\frac{C(S)}{R(s)} = \tag{4.11}$$

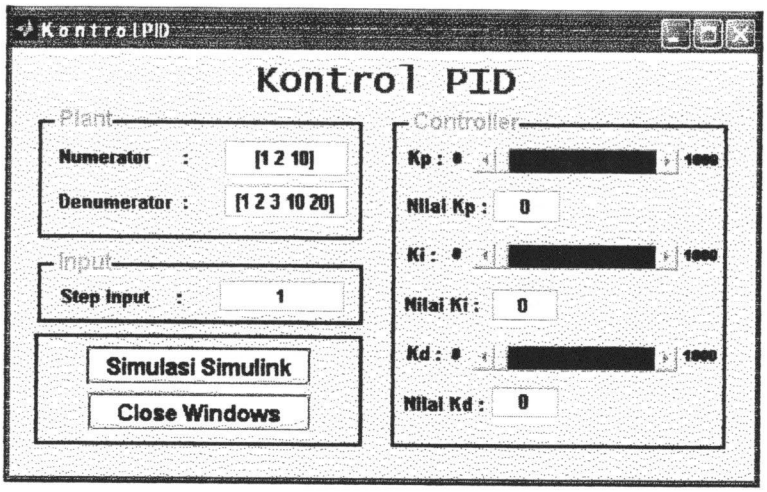
$$\frac{K_D s^4 + (2K_D + K_P)s^3 + (10K_D + 2K_P + Ki+)s^2 + (K_P + 2Ki)s + 10K_I}{s^5 + (2 + K_D)s^4 + (3 + 2K_D + K_P)s^3 + (10 + 10K_{Ii} + 2K_P + K_I)s^2 + (20 + K_P + 2K_I)s + 10K_I}$$

Numerator : [1 2 10]

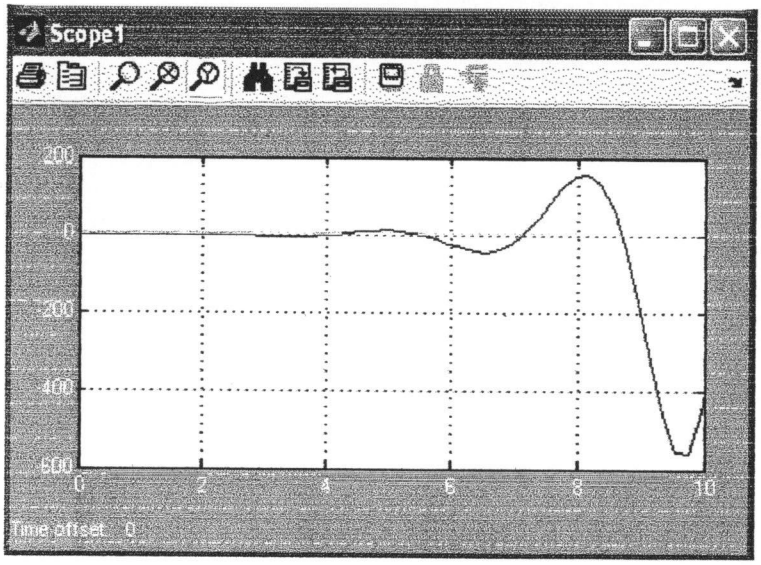
Denominator : [1 2 3 10 20]

Step input : [1]

Tampilan dari GUI Matlabnya dapat di lihat pada gambar 4.13 dan hasil simulasinya dapat di lihat pada grafik 4.10.



Gambar 4.13 Tampilan GUI Orde Tinggi



Grafik 4.10 *Natural respon* Orde Tinggi

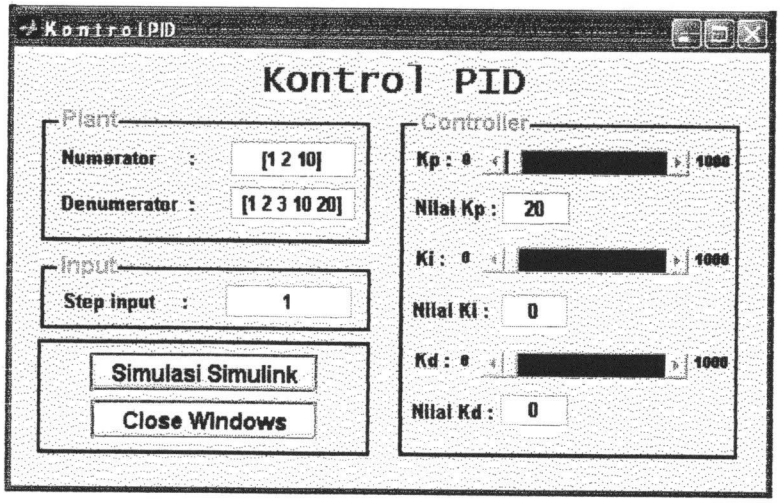
Tabel 4.10 Hasil percobaan 1 Orde Tinggi

Kp	Ki	Kd	Td	Tr	Tp	Mp	Ts	Ns
0	0	0	~	~	~	~	~	~

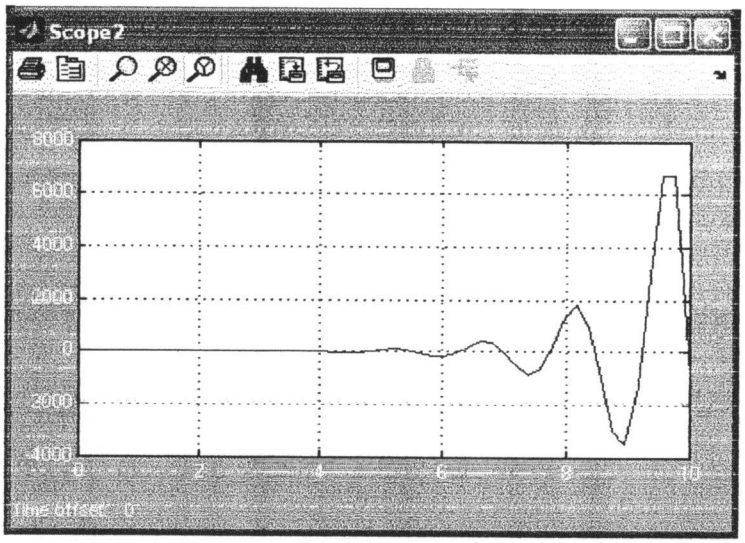
Tampak pada grafik 4.10 *Step respon* dari Orde Tinggi ini ternyata tidak terkendali dengan waktu naik dan waktu tunaknya tak terhingga dan overshoot terjadi secara terus menerus.

4.4.2 Percobaan 2 Orde Tinggi

Pada percobaan ini menggunakan kontrol *proportional* / Kp : 20. Tampilan GUI Matlabnya dapat di lihat pada gambar 4.14 dan keluarannya pada grafik 4.11 di bawah ini.



Gambar 4.14 Tampilan GUI Orde Tinggi dengan Kp : 20



Grafik 4.11 *Step respon* Orde Tinggi dengan kontroler P

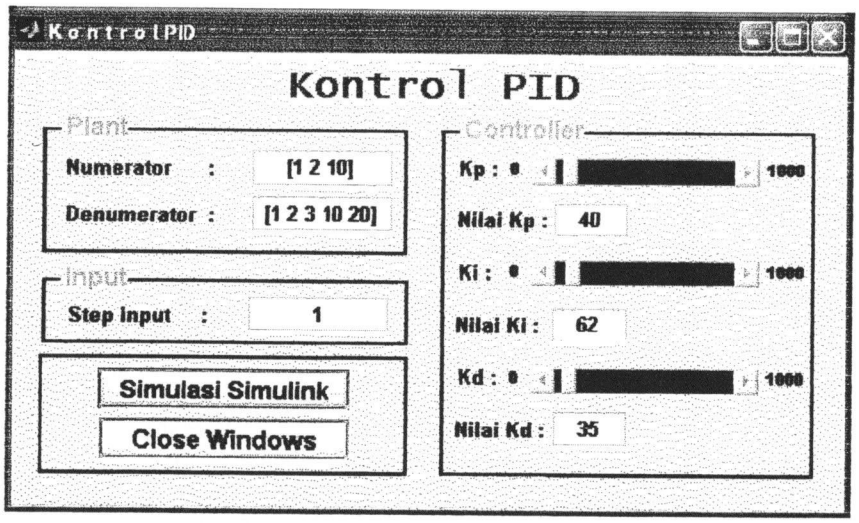
Tabel 4.10 Hasil percobaan 2 Orde Tinggi

Kp	Ki	Kd	Td	Tr	Tp	Mp	Ts	Ns
20	0	0	~	~	~	~	~	~

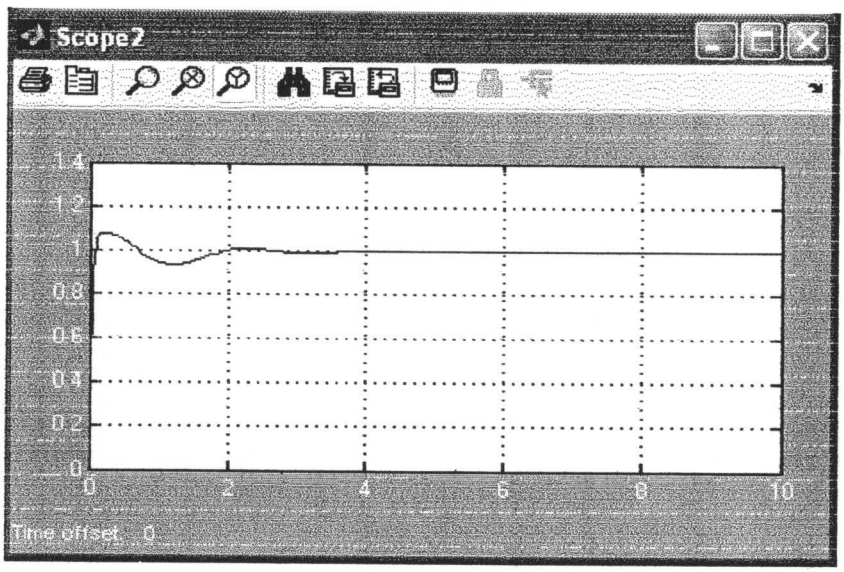
Pengendalian *proportional* menghasilkan respon yang buruk dengan waktu tunak tak terhingga dan *maximum overshoot* yang sangat besar.

4.4.3 Percobaan 3 Orde Tinggi

Kontroler yang digunakan adalah *proportional* / Kp : 40, *integral* / Ki : 62 dan *derivative* / Kd : 35 tampilan GUInya seperti pada gambar 4.15 dan keluarannya dapat di lihat pada grafik 4.12.



Gambar 4.15 Tampilan GUI Orde Tinggi dengan Kp : 40, Ki : 62 dan Kd : 35



Grafik 4.12 *Step respon* Orde Tinggi dengan kontroler PID

Tabel 4.12 Hasil percobaan 3 Orde Tinggi

Kp	Ki	Kd	Td	Tr	Tp	Mp	Ts	Ns
40	62	35	0.025	0.05	0.3	5	2.4	1

Tampak pada grafik 4.12 kesalahan keadaan tunaknya dapat dikurangi dengan pengendalian kontroler PID dan menghasilkan waktu naik 0.05s dan waktu tunak 2.4s akan tetapi overshot 5% dengan Tp : 0.3 tidak dapat dihilangkan.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan pengujian pengendali PID dengan GUI Matlab yang telah dilakukan maka dapat diambil beberapa kesimpulan sebagai berikut :

1. Kontrol PID dengan GUI Matlab dapat mengendalikan Orde Satu, Dua dan Orde Tiga.
2. Orde Satu dengan nilai *numerator* = [1] dan *denominator* = [1 10] dapat distabilkan pengendali PID dengan nilai Kp : 10, Ki : 100.
3. Orde Dua dengan nilai numerator = [1] dan denominator = [1 10 20] dapat distabilkan pengendali PID dengan nilai Kp : 100, Ki : 190 dan Kd : 15.
4. Orde Tiga dengan nilai *numerator* = [1.15 0.17] dan *denominator* = [1 0.73 0.92 0] dapat distabilkan pengendali PID dengan nilai Kp : 8, Ki : 5 dan Kd : 10.
5. Pengendali PID dengan GUI Matlab susah untuk menyetabilkan sistem yang mempunyai Orde lebih besar dari Tiga.
6. Penggunaan kendali *Proportional* untuk menyetabilkan sistem tidak bisa dilakukan jika tidak dikombinasikan dengan kendali PID lainnya yaitu *Integral* dan *Derivative.*
7. *Proportional* kontrol berfungsi untuk menambah waktu naik, d*erivative* kontrol untuk mengurangi *overshoot* dan *integral* kontrol untuk

menghilangkan kesalahan keadaan tunak, namun pada kenyataannya mengubah salah satu variabel dapat mengubah karakteristik lainnya.

5.2 Saran

Untuk pembuatan kontrol PID selanjutnya penulis menyarankan :

1. Penambahan *feature* pengendali PID dengan GUI matlab yang lebih lengkap.
2. Pembuatan simulasi baiknya minimal menggunakan intel pentium IV 1,8 Ghz atau yang setara, memori 256 MB dan VGA 64 agar software Matlab 7.0.4 dapat berjalan dengan baik.
3. Kontrol PID dengan GUI Matlab dapat di aplikasikan pada alat yang *real*.

DAFTAR PUSTAKA

- Arhami, Muhammad S.Si., M.Kom., & Desiani, Anita, S.Si., M. Kom., 2003. *Pemrograman MATLAB.* Yogyakarta, ANDI.
- Charles L., Phillips & Royce D, Harbor, 1998. *Sistem Kontrol.* Jakarta, Prenhalindo.
- MATLAB, *The Language of Technical Computing Using MATLAB*, 2005 Version 7.1.0, The MathWorks,Inc.
- Ogata, Katsuhiko, 1997. *Teknik Kontrol Automatik*, jilid 1 . Jakarta, Erlangga.
- Wahyu Dwi Hartono, Thomas & Agung Prasetyo, Y. Wahyu, 2003. *Analisis dan Desain Sistem Kontrol dengan MATLAB.* Yogyakarta, ANDI.

LAMPIRAN

❖ **Listing Program Pengendali PID dengan GUI (Graphical User Interface)**

MATLAB

```
function varargout = Hasil2(varargin)
% HASIL2 M-file for Hasil2.fig
%      HASIL2, by itself, creates a new HASIL2 or raises the existing
%      singleton*.
%
%      H = HASIL2 returns the handle to a new HASIL2 or the handle to
%      the existing singleton*.
%
%      HASIL2('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in HASIL2.M with the given input arguments.
%
%      HASIL2('Property','Value',...) creates a new HASIL2 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Hasil2_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Hasil2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Hasil2

% Last Modified by GUIDE v2.5 04-Jan-2006 06:05:31

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Hasil2_OpeningFcn, ...
                   'gui_OutputFcn',  @Hasil2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before Hasil2 is made visible.
function Hasil2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Hasil2 (see VARARGIN)

movegui(hObject,'onscreen')          % To display application onscreen

fig = handles.HasilTA

model_open(handles)

movegui(hObject,'center')            % To display application in the center of
screen


% Choose default command line output for Hasil2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Hasil2 wait for user response (see UIRESUME)
% uiwait(handles.HasilTA);


% --- Outputs from this function are returned to the command line.
function varargout = Hasil2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function model_open(handles)
% make sure the diagram is still open
if isempty(find_system('name','pidsub')),
   open_system('pidsub');open_system('pidsub/PID Controller');
   figure(handles.HasilTA)
   % Put  values of Kf and Ki from the GUI into the Block dialogs
   set_param('pidsub/Gain','Gain',...
      get (handles.Stepw,'String'))
   set_param('pidsub/Plant','Numerator',...
      get (handles.numw,'String'))
   set_param('pidsub/Plant','Denominator',...
      get (handles.denw,'String'))
```

```
    set_param('pidsub/Plant1','Numerator',...
      get (handles.numw,'String'))
    set_param('pidsub/Plant1','Denominator',...
      get (handles.denw,'String'))
    set_param('pidsub/PID Controller/Proportional','Gain',...
      get (handles.nilaikpw,'String'))
    set_param('pidsub/PID Controller/Integral','Gain',...
      get (handles.nilaikiw,'String'))
    set_param('pidsub/PID Controller/Derivative','Gain',...
      get (handles.nilaikdw,'String'))
end



function numw_Callback(hObject, eventdata, handles)
% hObject    handle to numw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of numw as text
%        str2double(get(hObject,'String')) returns contents of numw as a double

%ensure model is open
model_open(handles)

numw = get(hObject, 'String');

% Set the Transfer Fcn parameter of the Plant Numerator to the new value
set_param('pidsub/Plant','Numerator',numw)

% Set the Transfer Fcn parameter of the Plant Denominator to the new value
set_param('pidsub/Plant1','Numerator',numw)


% --- Executes during object creation, after setting all properties.
function numw_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

```
function denw_Callback(hObject, eventdata, handles)
% hObject    handle to denw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of denw as text
%        str2double(get(hObject,'String')) returns contents of denw as a double

%ensure model is open
model_open(handles)
denw = get(hObject, 'String');

% Set the Transfer Fcn parameter of the Plant Denominator to the new value
set_param('pidsub/Plant','Denominator',denw)

% Set the Transfer Fcn parameter of the Plant Denominator Block to the new
value
set_param('pidsub/Plant1','Denominator',denw)

% --- Executes during object creation, after setting all properties.
function denw_CreateFcn(hObject, eventdata, handles)
% hObject    handle to denw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on slider movement.
function sliderkpw_Callback(hObject, eventdata, handles)
% hObject    handle to sliderkpw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider

% Ensure model is open
model_open(handles)

% Get the new value for the Kpw Gain from the slider
NewVal = get(hObject,'Value');

% Set the value of the KpwCurrentValue to the new value set by slider
set(handles.nilaikpw,'String',NewVal)
```

```
% Set the Gain parameter of the Kp Gain Block to the new value
set_param('pidsub/PID Controller/Proportional','Gain',num2str(NewVal))

% --- Executes during object creation, after setting all properties.
function sliderkpw_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sliderkpw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end
function nilaikpw_Callback(hObject, eventdata, handles)
% hObject    handle to nilaikpw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nilaikpw as text
%        str2double(get(hObject,'String')) returns contents of nilaikpw as a double

% Ensure model is open
model_open(handles)

% Get the new value for the Kf Gain
NewStrVal = get(hObject,'String');
NewVal = str2double(NewStrVal);

% Check that the entered value falls within the allowable range
if  isempty(NewVal) | (NewVal< 0) | (NewVal>1000),
   % Revert to last value, as indicated by KfValueSlider
   OldVal = get(handles.sliderkpw,'Value');
   set(hObject,'String',OldVal)

else
   % Set the value of the KpValueSlider to the new value
   set(handles.sliderkpw,'Value',NewVal)

   % Set the Gain parameter of the Kp Gain Block to the new value
   set_param('pidsub/PID Controller/Proportional','Gain',NewStrVal)
end

% --- Executes during object creation, after setting all properties.
function nilaikpw_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nilaikpw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on slider movement.
function sliderkiw_Callback(hObject, eventdata, handles)
% hObject    handle to sliderkiw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% Ensure model is open
model_open(handles)

% Get the new value for the Ki Gain from the slider
NewVal = get(hObject,'Value');

% Set the value of the KiCurrentValue to the new value set by slider
set(handles.nilaikiw,'String',NewVal)

% Set the Gain parameter of the Ki Gain Block to the new value
set_param('pidsub/PID Controller/Integral','Gain',num2str(NewVal))

% --- Executes during object creation, after setting all properties.
function sliderkiw_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sliderkiw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function nilaikiw_Callback(hObject, eventdata, handles)
% hObject    handle to nilaikiw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nilaikiw as text
%        str2double(get(hObject,'String')) returns contents of nilaikiw as a double
```

```
% Ensure model is open
model_open(handles)

% Get the new value for the Ki Gain
NewStrVal = get(hObject,'String');
NewVal = str2double(NewStrVal);

% Check that the entered value falls within the allowable range
if  isempty(NewVal) | (NewVal< 0) | (NewVal>1000),
   % Revert to last value, as indicated by KiValueSlider
   OldVal = get(handles.sliderkiw,'Value');
   set(hObject,'String',OldVal)

else
   % Set the value of the KiValueSlider to the new value
   set(handles.sliderkiw,'Value',NewVal)

   % Set the Gain parameter of the Ki Gain Block to the new value
   set_param('pidsub/PID Controller/Integral','Gain',NewStrVal)
end
% --- Executes during object creation, after setting all properties.
function nilaikiw_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nilaikiw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end


% --- Executes on slider movement.
function sliderkdw_Callback(hObject, eventdata, handles)
% hObject    handle to sliderkdw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider

% Ensure model is open
model_open(handles)

% Get the new value for the KdGain from the slider
NewVal = get(hObject,'Value');
```

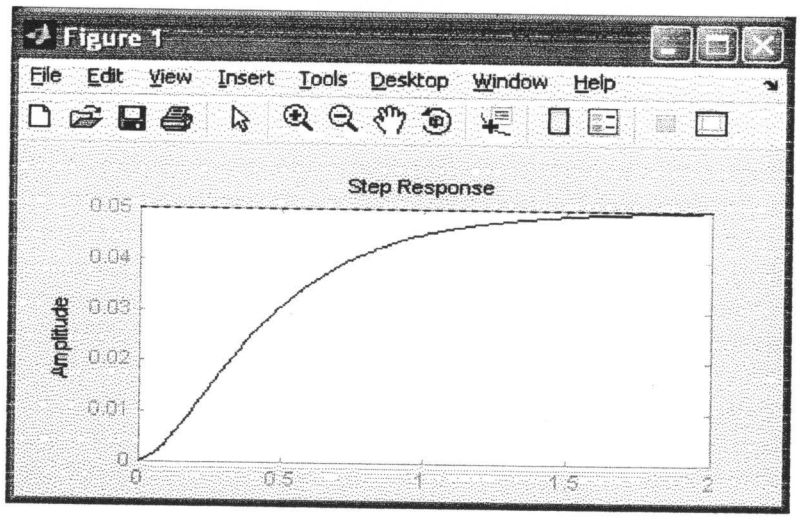
```
% Set the value of the KdCurrentValue to the new value set by slider
set(handles.nilaikdw,'String',NewVal)

% Set the Gain parameter of the Kd Gain Block to the new value
set_param('pidsub/PID Controller/Derivative','Gain',num2str(NewVal))

% --- Executes during object creation, after setting all properties.
function sliderkdw_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sliderkdw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


function nilaikdw_Callback(hObject, eventdata, handles)
% hObject    handle to nilaikdw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nilaikdw as text
%        str2double(get(hObject,'String')) returns contents of nilaikdw as a double

% Ensure model is open
model_open(handles)

% Get the new value for the Kd Gain
NewStrVal = get(hObject,'String');
NewVal = str2double(NewStrVal);

% Check that the entered value falls within the allowable range
if  isempty(NewVal) | (NewVal< 0) | (NewVal>1000),
   % Revert to last value, as indicated by KfValueSlider
   OldVal = get(handles.sliderkdw,'Value');
   set(hObject,'String',OldVal)

else
   % Set the value of the KdValueSlider to the new value
   set(handles.sliderkdw,'Value',NewVal)

   % Set the Gain parameter of the Kd Gain Block to the new value
   set_param('pidsub/PID Controller/Derivative','Gain',NewStrVal)
end
```

```
% --- Executes during object creation, after setting all properties.
function nilaikdw_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nilaikdw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in simulasiw.
function simulasiw_Callback(hObject, eventdata, handles)
% hObject    handle to simulasiw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% to run the sumulation
[timeVector,stateVector,outputVector] = sim('pidsub');

% --- Executes on button press in tutupw.
function tutupw_Callback(hObject, eventdata, handles)
% hObject    handle to tutupw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

close all;
 function Stepw_Callback(hObject, eventdata, handles)
% hObject    handle to Stepw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Stepw as text
%        str2double(get(hObject,'String')) returns contents of Stepw as a double


%ensure model is open
model_open(handles)

Stepw = get(hObject, 'String'); %NewStrVal cuma nama saja

% Set the Gain parameter of the Step Block to the new value
set_param('pidsub/Gain','Gain',Stepw)


% --- Executes during object creation, after setting all properties.
function Stepw_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Stepw (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

❖ **Listing program Matlab untuk menampilkan *natural respon* suatu sistem.**

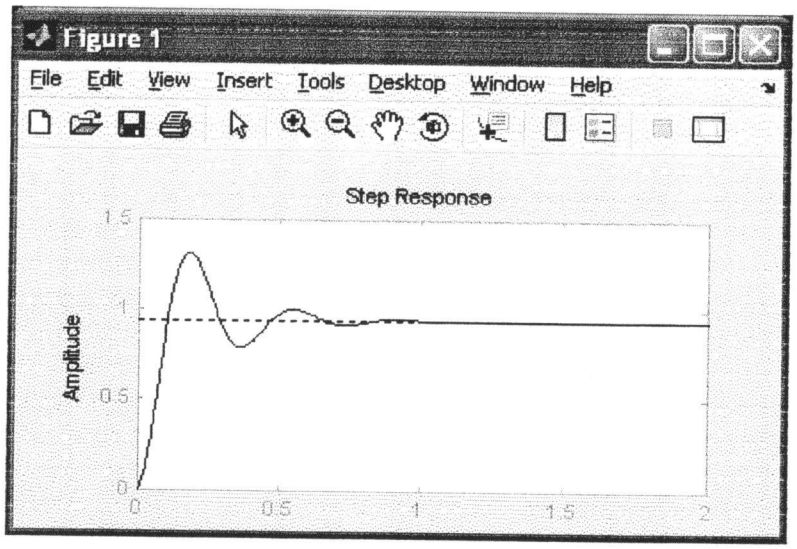
```
num=[1];
den=[1 10 20];
t=0:0.01:2;
step(num,den,t)
```



Gambar *Natural respon*

❖ **Listing program Matlab untuk menampilkan *step respon* suatu sistem dengan kontroler P.**

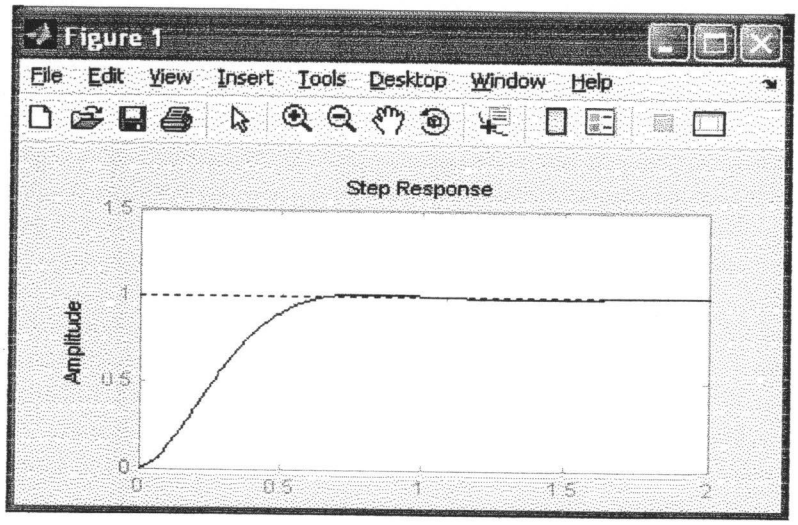
```
kp=300;
num=[kp];
den=[1 10 20+kp];
t=0:0.01:2;
step(num,den,t)
```



Gambar *Step respon* dengan kontroler P

❖ **Listing program Matlab untuk menampilkan *step respon* suatu sistem dengan kontroler PI.**

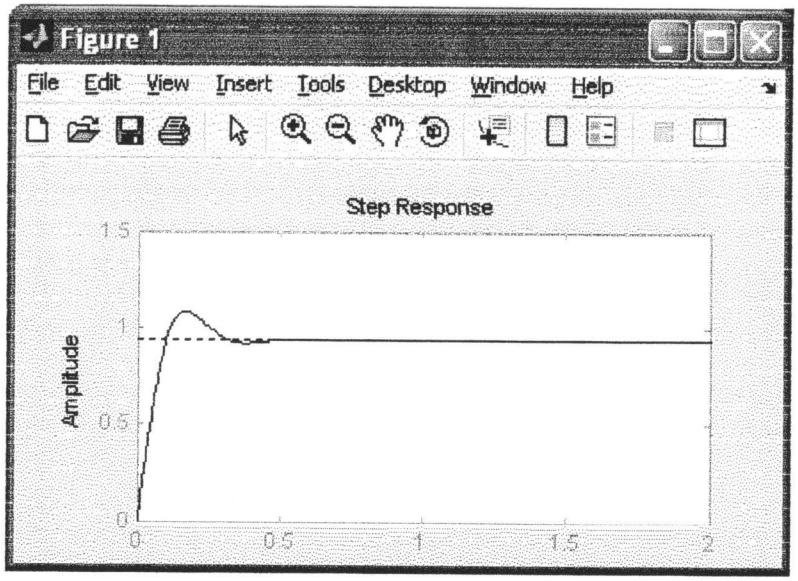
```
kp=30;
ki=70;
num=[kp ki];
den=[1 10 20+kp ki];
t=0:0.01:2;
step(num,den,t)
```



Gambar *Step respon* dengan kontroler PI

❖ **Listing program Matlab untuk menampilkan *step respon* suatu sistem dengan kontroler PD.**

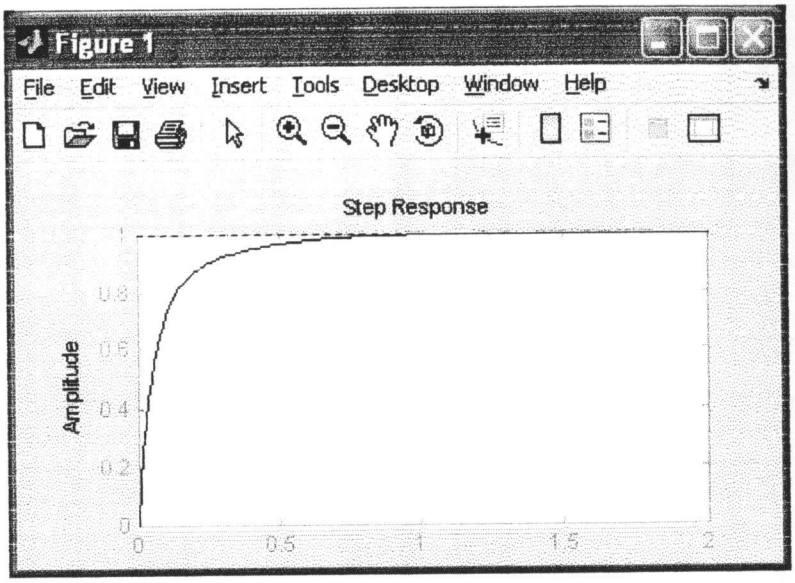
```
kp=300;
kd=10;
num=[kd kp];
den=[1 10+kd 20+kp];
t=0:0.01:2;
step(num,den,t)
```



Gambar *Step respon* dengan kontroler PD

❖ **Listing program Matlab untuk menampilkan *step respon* suatu sistem dengan kontroler PID.**

```
kp=100;
ki=190;
kd=15;
num=[kd kp ki];
den=[1 10+kd 20+kp ki];
t=0:0.01:2;
step(num,den,t)
```



Gambar *Step respon* dengan kontroler PID

❖ **Listing program Matlab untuk menampilkan *step respon* suatu sistem dan mencari nilai *loop* tertutup dengan kontroler PID.**

```
kp=100;
ki=190;
kd=15;
num=[1];
den=[1 10 20];
num1=[kd kp ki];
den1=[1 0];
num2=conv(num,num1);
den2=conv(den,den1);
[numc,denc]=cloop(num2,den2);
t=0:0.01:2;
step(numc,denc,t)
```