TUGAS AKHIR – TE141599

PENGATURAN OPTIMAL KONSUMSI GAS ALAM UNTUK PEMBANGKITAN LISTRIK PADA PLTG DI PT. INDONESIA POWER UP GRATI PASURUAN

Hanif Ryanas Putra
NRP 2215105077

Dosen Pembimbing
Yusuf Bilfaqih ST. , MT.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2017

FINAL PROJECT – **TE141599**

OPTIMAL CONTROL OF NATURAL GAS CONSUMPTION FOR POWER GENERATION FROM GAS TURBINE AT PT. INDONESIA POWER UP GRATI PASURUAN

Hanif Ryanas Putra
NRP 2215105077

Advisor
Yusuf Bilfaqih ST. , MT.

DEPARTEMENT OF ELECTRICAL ENGINEERING
Faculty of Electrical Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2017

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul " **PENGATURAN OPTIMAL KONSUMSI GAS ALAM UNTUK PEMBANGKITAN LISTRIK PADA PLTG DI PT. INDONESIA POWER UP GRATI PASURUAN** " adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan, dan bukan hasil merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan berlaku.

Surabaya, Juli 2017

Hanif Ryanas Putra
NRP 2215 105 077

PENGATURAN OPTIMAL KONSUMSI GAS ALAM UNTUK PEMBANGKITAN LISTRIK PADA PLTG DI PT. INDONESIA POWER UP GRATI PASURUAN

TUGAS AKHIR

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

Menyetujui :

Dosen Pembimbing,

Yusuf Bilfaqih, ST., MT.
NIP. 19720325 199903 1 001

SURABAYA
JULI, 2017

PENGATURAN OPTIMAL KONSUMSI GAS ALAM UNTUK PEMBANGKITAN LISTRIK PADA PLTG DI PT. INDONESIA POWER UP GRATI PASURUAN

Nama	:	Hanif Ryanas Putra
Dosen Pembimbing	:	Yusuf Bilfaqih ST., MT.

ABSTRAK

Pada PLTG PT. Indonesia Power Unit Pembangkitan (UP) Grati, daya terbangkit yang tercatat sebesar 3x100MW. Pada Pembangkit Listrik Tenaga Uap (PLTG) umumnya menggunakan bahan baku utama yaitu gas alam sebagai pendorong turbin gas dengan tekanan tinggi. Dengan begitu maka dimanfaatkan untuk dapat menggerakkan turbin hingga listrik pun terbangkit. Untuk meningkatkan efisiensi penggunaan gas alam pada pembangkit PT. Indonesia Power UP Grati ini diperlukan adanya metode untuk meningkatkan efiensinya dengan menimalkan gas alam yang digunakan dengan daya yang terbangkit mendekati daya yang tercatat yaitu maksimal sebesar 100MW untuk masing-masing unitnya. Penelitian dilakukan tentang penerapan algoritma PSO (*Particle Swarm Optimization*) pada identifikasi pemodelan sistem dan penerapan kontroler LQG (*Linear Quadratic Gaussian*) pada proses optimalisasi sistem tersebut. Hasil dari pengujian dapat dilihat bahwa penerapan algoritma PSO dapat menghasilkan nilai *fitness* model yang lebih baik yaitu sebesar 0,6901 atau 69,01%. Dan juga hasil LQG dapat meningkatkan efisiensi pada sistem ini hingga mencapai 45%.

Kata kunci: Simulasi *plant*, PSO, LQG, Optimasi Bahan Bakar

Halaman ini sengaja dikosongkan

OPTIMAL CONTROL OF NATURAL GAS COMSUMPTION FOR POWER GENERATION FROM GAS TURBINE AT PT. INDONESIA POWER UP GRATI PASURUAN

Name	:	Hanif Ryanas Putra
Advisor	:	Yusuf Bilfaqih ST., MT.

ABSTRACT

In PLTG PT. Indonesia Power Generation Unit (UP) Grati, power generated recorded at 3x100MW. In Gas Power Plant (PLTG) generally use the main raw materials of natural gas as a driver of high pressure gas turbine. With that then it is used to be able to move the turbine until electricity is infected. To increase the efficiency of natural gas utilization at PT. Indonesia Power UP Grati is needed a method to improve its efficiency by minimizing the natural gas used with power that is nearing the recorded power of 100MW maximum for each unit. The research conducted on the application of PSO algorithm (Particle Swarm Optimization) on the identification of system modeling and the application of LQG (Linear Quadratic Gaussian) controller in the process of optimizing the system. The results of the test can be seen that the application of PSO algorithm can produce a better fitness model value of 0.6901 or 69.01%. And also the LQG results can improve the efficiency of this system up to 45%.

Keywords: Plant Simulation, PSO, LQG,Fuel Optimization

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Dengan mengucapkan puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, hidayah serta karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir dengan judul :

"Pengaturan Optimal Konsumsi Gas Alam untuk Pembangkitan Listrik pada PLTG di PT. Indonesia Power UP Grati Pasuruan"

untuk menyelesaikan studi dan memperoleh gelar Sarjana Teknik di Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember.

Dengan kesempatan ini penulis menyampaikan terima kasih sebesar-besarnya kepada pihak-pihak yang banyak berjasa terutama dalam pengerjaan Tugas Akhir ini, yaitu segenap keluarga besar terutama kedua orang tua atas limpahan doa, kasih sayang dan teladan hidup bagi penulis. Bapak Yusuf Bilfaqih ST., MT., selaku dosen pembimbing yang selalu memberikan pengarahan kepada saya selama Tugas Akhir ini. Seluruh staf pengajar dan administrasi Departemen Teknik Elektro FTE-ITS. Semua rekan-rekan yang telah banyak membantu untuk menyelesaikan Tugas Akhir ini yang tidak dapat penulis sebutkan satu persatu.

Laporan Tugas Akhir ini masih jauh dari sempurna. Oleh karena itu, penulis mengharapkan kritik dan saran dari pembaca. Semoga buku laporan Tugas Akhir ini dapat memberikan manfaat bagi pembaca sebagai acuan penelitian selanjutnya

Surabaya, Juli 2017

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

HALAMAN JUDUL.....i
PERNYATAAN KEASLIAN TUGAS AKHIR.....v
LEMBAR PENGESAHAN.....vii
ABSTRAK.....ix
KATA PENGANTAR.....xiii
DAFTAR ISI.....xv
DAFTAR GAMBAR.....xvii
DAFTAR TABEL.....xix
BAB 1 PENDAHULUAN.....1
1.1 Latar Belakang.....1
1.2 Perumusan Masalah.....1
1.3 Batasan Masalah.....2
1.4 Tujuan.....2
1.5 Metodologi.....2
1.6 Sistematika Penulisan.....4
1.7 Relevansi.....4
BAB 2 TINJAUAN PUSTAKA.....5
2.1 Pembangkit Listrik Tenaga Gas.....5
2.1.1 Pengertian Umum.....5
2.1.2 Komponen Utama pada PLTG.....6
2.1.3 Kelemahan dan Keunggulan PLTG.....7
2.2 Identifikasi Sistem.....8
2.2.1 Identifikasi Statis.....9
2.2.2 Identifikasi Dinamis.....10
2.3 Pemodelan dalam *State-Space*.....12
2.4 Teori Kontrol Optimal.....13
2.4.1 *Linear Quadratic Regulator*.....14
2.4.2 *Discrete-Time Linear Quadratic Regulator*.....17
2.4.3 *Linear Quadratic Regulator* dengan *Servo* Sistem Tipe 1.....19
2.4.4 *Linear Quadratic Gaussian* (LQG).....23
2.4.5 *Discrete-Time Linear Quadratic Gaussian*.....25
2.5 *Swarm Intelligence*.....27
2.5.1 *Particle Swarm Optimization* (PSO).....28
2.5.2 PSO Modifikasi.....34
BAB 3 PERANCANGAN DAN IMPLEMENTASI SISTEM...35
3.1 Pengambilan Data.....35

3.2 Pembuatan Program Identifikasi ARX dan *State-Space* 36
3.3 Penerapan Algoritma PSO pada Identifikasi Sistem.....40
3.4 Perancangan Kontroler menggunakan *Linear Quadratic Gaussian* (LQG) 42
BAB 4 HASIL SIMULASI DAN ANALISA DATA.....47
4.1 Hasil Simulasi Identifikasi dan Pemodelan Sistem 47
4.2 Hasil Penerapan Algoritma PSO pada Identifikasi Model ARX 51
4.3 Analisa Data Hasil Identifikasi Sistem 60
4.4 Hasil Simulasi Kontrol Optimal menggunakan LQG 61
4.4.1 Hasil Simulasi LQG pada PLTG dengan Daya 30MW 63
4.4.2 Hasil Simulasi LQG pada PLTG dengan Daya 50MW 64
4.4.3 Hasil Simulasi LQG pada PLTG dengan Daya 75MW 66
4.4.4 Hasil Simulasi LQG pada PLTG dengan Daya 100MW 68
4.5 Analisa Data Hasil Simulasi Kontrol Optimal 69
BAB 5 PENUTUP.....71
5.1 Kesimpulan.....71
5.2 Saran.....71
DAFTAR PUSTAKA 73
LAMPIRAN.....75
RIWAYAT PENULIS 85

DAFTAR GAMBAR

Gambar 1.1 Metodologi Penelitian 2
Gambar 2.1 Contoh Turbin Gas pada PLTG 6
Gambar 2.2 Ilustrasi *Generator* pada PLTG 7
Gambar 2.3 Respon Sistem Orde 2 9
Gambar 2.4 Diagram Blok Kontrol Optimal 14
Gambar 2.5 Diagram Blok *Servo* Sistem Tanpa *Integrator* 20
Gambar 2.6 Diagram Blok *Servo* Sistem dengan *Integrator* 21
Gambar 2.7 *Regulator* Dinamik menggunakan *Observer* dan *Full-State Feedback* 24
Gambar 2.8 Diagram *Regulator* Dinamik LQG 25
Gambar 2.9 Topologi PSO : *Ring Topology* (Kiri) dan *Star Topology* (Kanan) 30
Gambar 2.10 Ilustrasi dari Suatu Partikel pada PSO 31
Gambar 2.11 Ilustrasi Perubahan Kecepatan pada PSO 32
Gambar 2.12 Ilustrasi Perubahan Posisi pada PSO 33
Gambar 3.1 Hasil *Plot* Data Masukan dan Keluaran PLTG 35
Gambar 3.2 Hasil *Plot* Data Masukan dan Keluaran PLTG saat 100MW 36
Gambar 3.3 Cara Mengetahui Nilai Varian dari *Noise* Sistem 38
Gambar 3.4 *Pseudocode* Algoritma PSO 40
Gambar 3.5 Hasil *Plot* Sinyal Kontrol LQR dengan *Trial-Error* q dan r 43
Gambar 3.6 Simulasi Kontroler LQG pada *Simulink* 44
Gambar 3.7 *Subsistem*: (a) *Plant*, (b) *Observer* dan (c) *Filter* Kalman 45
Gambar 4.1 Rancangan *Simulink* Model ARX 47
Gambar 4.2 Hasil Simulasi Model ARX 48
Gambar 4.3 Hasil *Error* Model ARX 49
Gambar 4.4 Hasil Simulasi Model ARX dengan Data Beban Penuh 50
Gambar 4.5 Hasil *Error* Simulasi Model ARX dengan Data Beban Penuh 51
Gambar 4.6 Hasil Running Program PSO pada Matlab 52
Gambar 4.7 Hasil Nilai *Fitness* menggunakan Algoritma PSO 53
Gambar 4.8 Hasil Model *State-Space* dari Algoritma PSO 54
Gambar 4.9 Hasil Simulasi Model dengan Algortima PSO 54
Gambar 4.10 Hasil *Error* Model dengan Algoritma PSO 55

Gambar 4.11 Hasil Perbandingan dari Model ARX dan Model PSO.....55
Gambar 4.12 Hasil Perbandingan Nilai *Error* Model ARX dan Model PSO.....56
Gambar 4.13 Hasil Nilai *Fitness* Model pada Data Beban Penuh menggunakan Algoritma PSO 57
Gambar 4.14 Hasil Model *State-Space* dari Algoritma PSO 58
Gambar 4.15 Hasil Perbandingan Respon dengan Algoritma PSO.....59
Gambar 4.16 Hasil Perbandingan *Error* Model ARX dan ARX-PSO.....59
Gambar 4.17 Hasil *Plot* Sinyal Kontrol dengan Kontroler LQR ..61
Gambar 4.18 Hasil *Plot State* Optimal dengan Kontroler LQR 62
Gambar 4.20 Hasil Simulasi dengan LQG pada PLTG 30MW 63
Gambar 4.21 Hasil Perbandingan Simulasi LQG dengan Kondisi Nyata PLTG 30MW 64
Gambar 4.22 Hasil *Error* pada nilai *Flow* dan sinyal kontrol LQG pada PLTG 30MW 64
Gambar 4.23 Hasil Simulasi dengan LQG pada PLTG 50MW 65
Gambar 4.24 Hasil Perbandingan Simulasi LQG dengan Kondisi Nyata PLTG 50MW 65
Gambar 4.25 Hasil *Error* pada nilai *Flow* dan sinyal kontrol LQG pada PLTG 50MW 66
Gambar 4.26 Hasil Simulasi dengan LQG pada PLTG 75MW 66
Gambar 4.27 Hasil Perbandingan Simulasi LQG dengan Kondisi Nyata PLTG 75MW 67
Gambar 4.28 Hasil *Error* pada nilai *Flow* dan sinyal kontrol LQG pada PLTG 75MW 67
Gambar 4.29 Hasil Simulasi dengan LQG pada PLTG 100MW ..68
Gambar 4.30 Hasil Perbandingan Simulasi LQG dengan Kondisi Nyata PLTG 100MW 69
Gambar 4.31 Hasil *Error* pada nilai *Flow* dan sinyal kontrol LQG pada PLTG 100MW 69

DAFTAR TABEL

Tabel 2.1 Sifat pada Matriks 15
Tabel 3.1 Matriks Pembobot q dan r dengan Indeks Performansi..43
Tabel 4.1 Hasil Perbandingan Nilai *Fitness* pada Model 61
Tabel 4.2 Hasil Perbandingan Nilai *Set Point* dan
Efisiensi PLTG 70

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

Pada bab ini, dibahas mengenai latar belakang, cakupan permasalahan, dan tujuan serta manfaat dilakukan penelitian Tugas Akhir ini.

1.1 Latar Belakang

Turbin gas (GT) diketahui memiliki biaya modal yang rendah untuk rasio daya, fleksibilitas tinggi, keandalan yang tinggi. Oleh karena itu, turbin gas sangat kompatibel untuk digunakan sebagai pembangkit tenaga listrik. Pada PLTG PT. Indonesia Power UP Grati mempunyai 3 unit turbin gas dengan daya terbangkit yang tercatat sebesar 100MW untuk tiap unitnya. Sehingga biasa disebut dengan PLTG berkapasitas 3x100 MW. Dalam rangka peningkatan efisiensi dari PLTG dengan meminimalkan penggunaan gas alam serta nilai daya yang terbangkit dapat mendekati dari daya yang dibutuhkan.

Dikarenakan ketersediaan dari (resource) sumber daya gas alam yang tidak dapat diperbaharui dan jumlahnya terbatas. Maka diperlukan adanya upaya meminimalkan penggunaan gas alam sebagai bahan bakar utama pembangkitan.

Dengan menggunakan metode identifikasi sistem didapatkan prediksi nilai parameter dari *flowrate* atau aliran gas terhadap keluaran dari daya yang terbangkit. [1] Setelah itu, didapatkan persamaan pendekatan sistemnya dengan prediksi tersebut. Untuk mengoptimalkan pendekatan sistemnya ditambahkan algoritma PSO (*Particle Swarm Optimization*) sehingga pendekatan pemodelan lebih akurat. Setelah didapatkan pendekatan modelnya digunakan metode kontrol optimal LQG (*Linear Quadratic Gaussian)* untuk meminimalkan *cost function* pada sistem ini. Sehingga efisiensi pada pembangkit ini dapat meningkat.

1.2 Perumusan Masalah

Permasalahan pada Tugas Akhir ini adalah tentang bagaimana cara untuk meminimalkan pemakaian gas alam namun tetap menjaga nilai daya yang terbangkit mendekati atau sesuai dengan nilai daya yang dibutuhkan.

1.3 Batasan Masalah

Pada Tugas Akhir ini dilakukan pembatasan masalah sebagai berikut .

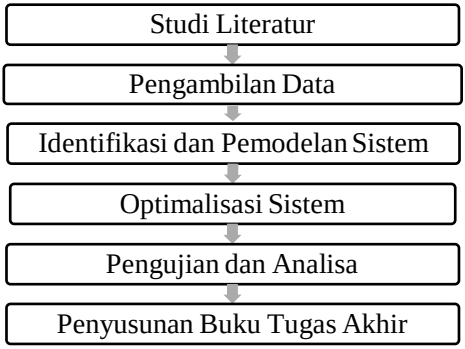
a. Data pengukuran diperoleh dari PLTG di PT. Indonesia Power UP Grati, Pasuruan.
b. Pemodelan dilakukan menggunakan pendekatan *black-box* bedasarkan data hasil pengukuran.
c. Variabel keadaan/*state* sistem tidak merepresentasikan besaran fisik yang ada pada sistem.
d. Pemodelan memiliki keterbatasan data untuk merepresentasikan besaran frekuensi putaran pada turbin gas yang menggerakkan generator.

1.4 Tujuan

Tujuan yang ingin dicapai dari Tugas Akhir ini adalah mendapatkan sistem pengaturan yang mampu meminimalkan penggunaan gas alam dengan tetap mendapatkan nilai daya yang terbangkit mendekati atau sesuai dengan nilai *set point* daya sehingga dapat meningkatkan efisiensi dari pengolahan gas alam pada pembangkitan listrik di PT. Indonesia Power UP Grati, Pasuruan.

1.5 Metodologi

Penelitian ini akan dilakukan melalui beberapa tahap, yaitu akan dijelaskan melalui diagram alir pada Gambar 1.1 berikut.



Gambar 1.1 Metodologi Penelitian

1.5.1 Studi Literatur

Pada tahap ini akan dilakukan kegiatan pengumpulan dan pengkajian hal-hal terkait teori, informasi maupun hasil eksperimen serupa yang dapat dijadikan referensi dalam proses penyusunan Tugas Akhir ini. Sumber yang dikumpulkan dan dikaji dapat diperoleh melalui berbagai sumber ilmiah seperti buku, hasil penelitan, maupun jurnal ilmiah yang telah dipublikasikan.

1.5.2 Pengambilan Data

Pada tahap ini dilakukan pengambilan data pada pembangkit listrik tenaga gas (PLTG) milik PT. Indonesia Power yang berlokasi di Grati Kabupaten Pasuruan. Data yang diambil merupakan data aliran (*flowrate)* gas alam dan juga data dari daya yang terbangkit setiap harinya.

1.5.3 Identifikasi dan Pemodelan Sistem

Pada tahap ini sistem pembangkitan listrik pada PLTG diidentifikasi dengan menggunakan data-data yang telah diperoleh tanpa menghiraukan isi dalam sistem tersebut. Teknik identifikasi ini biasa disebut dengan *Box-Jenkins.* Setelah itu ditambahkan algoritma PSO untuk mendapatkan hasil pemodelan yang lebih baik. Sistem dimodelkan menjadi dalam persamaan beda dan *state-space*.

1.5.4 Optimalisasi Sistem

Setelah didapatkan pemodelan sistem digunakan kontrol optimal untuk mengoptimalkan sinyal kontrol sehingga efisiensi pembangkit akan meningkat.

1.5.5 Pengujian dan Analisa

Dengan menghitung nilai kesalahan atau *error* menggunakan metode *Root Mean Square Error* (MSE) untuk mengetahui adanya adanya *error* pendekatan dengan bentuk kuadrat terkecil.

1.5.6 Penyusunan Buku Tugas Akhir

Pada tahap ini akan dilakukan penyusunan laporan terkait hasil proses Tugas Akhir yang telah dilakukan. Penyusunan buku Tugas Akhir dilakukan sebagai bentuk laporan tertulis dari proses dan hasil kerja terkait topik yang diusulkan.

1.6 Sistematika Penulisan

Sistematika penulisan dalam Tugas Akhir ini terdiri atas lima bab dengan uraian sebagai berikut.

Bab I : Pendahuluan
Bab ini membahas tentang penjelasan mengenai latar belakang, permasalahan, batasan masalah, tujuan, metodologi, sistematika pembahasan dan relevansi.

Bab II : Tinjauan Pustaka
Bab ini membahas tentang teori-teori mengenai pembangkitan listrik, identifikasi sistem, serta kontrol optimal.

Bab III : Perancangan Sistem
Bab ini membahas tentang perancangan dari sistem yang dibuat untuk mengatasi permasalahan optimalisasi

Bab IV : Simulasi dan Analisa Data
Bab ini membahas tentang hasil simulasi analisis pemodelan sistem serta hasil dari kontrol optimal yang dilakukan

Bab V : Penutup
Bab ini berisi kesimpulan dan saran dari hasil pembahasan yang telah dilakukan.

1.7 Relevansi

Pada Tugas Akhir ini didapatkan manfaat sebagai berikut :

a. Dengan simulasi yang telah dilakukan, didapatkan nilai optimal dari aliran gas sebagai sumber utama pada PLTG sehingga efisiensi dapat ditingkatkan.
b. Dapat digunakan untuk referensi atau rujukan pada penelitian selanjutnya.

BAB 2
TINJAUAN PUSTAKA

Pada bab ini, dibahas mengenai beberapa teori dan konsep yang digunakan dalam pengerjaan Tugas Akhir ini.

2.1 Pembangkit Listrik Tenaga Gas

2.1.1 Pengertian Umum

Pembangkitan adalah proses produksi tenaga listrik yang dilakukan dalam pusat-pusat tenaga listrik atau sentral-sentral dengan menggunakan generator. PLTG adalah salah satu jenis pembangkit listrik yang menggunakan turbin gas sebagai *prime mover*-nya dengan gas sebagai fluida kerjanya. Dibandingkan dengan pembangkit lainnya turbin gas merupakan pembangkit sederhana yang terdiri dari atas empat komponen utama yaitu, kompresor, ruang bakar (*combustor*), turbin gas, dan generator. Bahan bakar pada PLTG berupa minyak bumi atau gas alam yang dibakar di dalam ruang bakar. [2]

Secara garis besar urutan kerja dari proses operasi PLTG adalah sebagai berikut:

a. Proses Starting

   Pada proses start awal untuk memutar turbin menggunakan mesin diesel sampai putaran poros *turbine*/*compressor* mencapai putaran 3400 rpm maka secara otomatis diesel dilepas dan akan berhenti

b. Proses kompresi

   Udara dari luar kemudian dihisap melalui *air inlet* oleh kompresor dan masuk ke ruang bakar dengan cara dikabutkan bersama bahan bakar lewat *nozzle* secara terus menerus dengan kecepatan tinggi.

c. Transformasi energi thermis ke mekanik

   Kemudian udara dan bahan bakar dikabutkan ke dalam ruang bakar diberi pengapian (*ignition*) oleh busi (*spark plug*) saat pemulaan pembakaran. Dan seterusnya hasil pembakarannya berupa gas dengan temperatur dan tekanan tinggi dialirkan menjadi tenaga mekanis pada perputaran poros.

d. Transformasi energi mekanis ke energi listrik

   Poros turbin berputar hingga 5100 rpm yang sekaligus memutar poros generator sehingga menghasilkan tenaga listrik.

Putaran turbin 5100 rpm diturunkan oleh *load gear* menjadi 3000 rpm dan kecepatan putaran turbin ini digunakan untuk memutar generator.

.

2.1.2 Komponen Utama pada PLTG

Bagian-bagian utama yang menyusun PLTG antara lain:

a. Saluran Udara Masuk

Saluran udara masuk ini dilengkapi dengan saringan penangkap bintik-bintik air dan debu. Tujuannya untuk menghindari korosi pada sudu-sudu kompresor yang diakibatkan oleh bintik-bintik air dan debu yang terkandung pada udara masuk ke saluran.

b. Kompresor

Merupakan alat yang digunakan untuk mengkompresikan udara dengan jumlah yang besar untuk keperluan pembakran, pendinginan, dan lain-lain.

c. Ruang Bakar (*Combustor*)

Bagian-bagian yang menunjang proses pembakaran pada ruang bakar antara lain sistem penyalaan, *flame detector*, dan *cross fire tube*. Dari hasil pembakaran bahan bakar, gas panas yang dihasilkan digunakan untuk menggerakkan turbin.

d. Turbin Gas

Merupakan bagian terpenting dari perangkat PLTG yang mengkonversikan energi panas dari hasil pembakaran di ruang bakar yang bertemperatur dan bertekanan tinggi ke suatu energi yang baru yaitu energi mekanik. Seperti pada Gambar 2.1 berikut.

Gambar 2.1 Contoh Turbin Gas pada PLTG

e. Saluran Gas Buang

Suatu bagian dari sistem turbin, di mana gas yang telah dipergunakan untuk memutar poros turbin dan kemudian dibunag pada atmosfer udara.

f. Bantalan

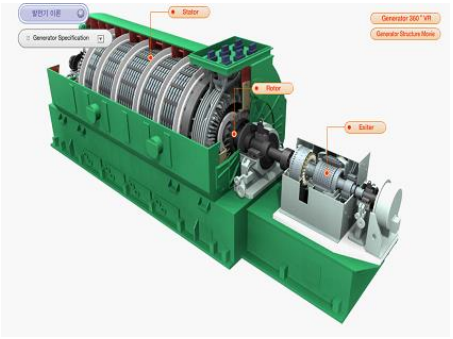
Unit turbin gas menggunakan *Journal bearing* dan *Thrust bearing* sebagai bantalan. Fungsi bagian ini untuk menunjang rotor turbin sebagai penghubung rotor dan stator turbin.

g. Load Gear

Disebut juga *reduction gear* ata *load coupling* yang digunakan untuk mengurangi kecepatan yang dibutuhkan oleh generator.

h. Generator

Merupakan komponen penting yang berfungsi mengubah energi mekanis dari turbin menjadi energi listrik yang disalurkan ke beberapa area kerjanya. Untuk gambaran tentang generator seperti pada Gambar 2.2 berikut.



Gambar 2.2 Ilustrasi *Generator* pada PLTG

2.1.3 Kelemahan dan Keunggulan PLTG

Dari segi operasi, unit PLTG tergolong unit yang masa startnya singkat yaitu sekitar 15 – 30 menit dan umumnya dapat distart tanpa pasukan daya listrik dari luar karena menggunakan mesin diesel sebagai penggerak awalnya (*diesel engine motor start*). Dari segi pemeliharaan, unit PLTG mempunyai selang waktu pemeliharaan (*time between overhaul*) yang pendek yaitu sekitar 4000 – 5000 jam operasi. Selain ukuran jam operasi juga dapat

dipakai jumlah *start-stop* sebagai acuan dalam penentuan waktu *overhaul*. Jadi walaupun belum mencapai 5000 jam operasi tetapi telah mencapai 300 kali *start-stop* maka unit PLTG tersebut sudah harus di-inspeksi untuk pemeliharaan. Dalam proses inspeksi, hal-hal yang perlu diperhatikan adalah bagian-bagian yang terkena aliran gas hasil pembakaran yang suhunya bisa mencapai 1300°C seperti ruang bakar, saluran gas panas dan juga sudu-sudu turbin. Bagian-bagian ini umumnya rawan mengalami kerusakan atau keretakan sehingga perlu dilas atau diganti bila perlu. [2]

Proses *start-stop* akan mempercepat proses kerusakan/ keretakan karena proses tersebut menyebabkan proses pemuaian dan pengerutan yang tidak kecil pada bagian-bagian tersebut. Hal ini disebabkan sewaktu unit PLTG dingin suhunya sama dengan suhu ruangan yaitu sekitar 30°C namun pada saat beroperasi suhunya dapat mencapai hingga 1300°C demikian pula sebaliknya. Pada saat unit PLTG *shut-down,* porosnya harus tetap diputar secara perlahan untuk menghindari terjadinya pembengkokan pada poros hingga suhunya dianggap cukup aman untuk itu.

Unit PLTG umumnya merupakan unit pembangkit dengan efisiensi yang paling rendah, yaitu sekitar 15-25% saja. PLTG di Indonesia menerapkan sistem terbuka (*open cycle*), berbeda dengan di negara yang memiliki temperatur yang dingin, gas buang dari PLTG dapat dimanfaatkan sebagai penghangat ruangan. Di Indonesia PLTG dapat diterpakan menggunakan sistem tertutup (*combine cycle*), di mana gas buang PLTG dimanfaatkan untuk memanaskan air pada unit pembangkit PLTU pembangkit jenis ini disebut PLTGU.

2.2 Identifikasi Sistem

Identifikasi sistem merupakan proses pengamatan suatu sistem dan menentukan parameter-parameter yang dimiliki sistem tersebut. Identifikasi dilakukan dengan cara mengamati respon sistem terhadap suatu masukan tertentu.

Terdapat dua jenis identifikasi sistem yaitu identifikasi statis dan identifikasi dinamis. Perbedaan dari kedua metode identifikasi ini terletak pada masukan sistem yang diberikan. Pada identifikasi statis, sinyal masukan yang diberikan pada sistem adalah sinyal kontinyu beraturan seperti sinyal *step*, *ramp,* dan *sinusoidal*. Sedangkan pada identifikasi dinamis, sinyal masukan yang diberikan berupa sinyal yang tidak beraturan.

2.2.1 Identifikasi Statis

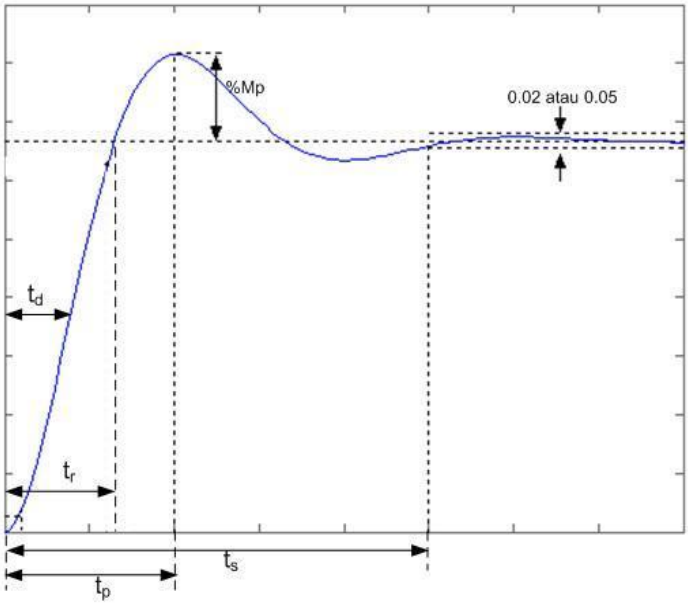
Identifikasi statis merupakan metode identifikasi dengan mengamati sinyal keluaran ketika sistem diberikan masukan sinyal kontinyu dan teratur seperti sinyal *step*, *ramp*, dan *sinusoidal*. Identifikasi statis ini dilakukan untuk mencari parameter dari sistem melalui pendekatan secara grafis. Karakteristik respon transien sistem terdiri dari:

1. Spesifikasi Teoritis

   Pada sistem orde satu terdapat konstanta waktu (τ) yang merupakan waktu yang dibutuhkan respon mulai t=0 sampai dengan respon mencapai 63,2% dari respon *Steady State*. Pada sistem orde dua terdapat frekuensi teredam (ω_n) dan konstanta redaman (ξ).

2. Spesifikasi Praktis

   Merupakan spesifikasi dengan melihat dari waktu-waktu tertentu dalam respon sistem seperti pada Gambar 2.3 , antara lain:



Gambar 2.3 Respon Sistem Orde 2

a. Waktu Tunda / *Time Delay* (t_d)

Adalah waktu yang diperlukan oleh respon untuk mencapai setengah dari nilai *Steady State* tunak waktu pertama.

b. Waktu Naik / *Rise Time* (t_r)

Adalah waktu yang dibutuhkan oleh respon untuk naik dari 5% ke 95% atau 10% ke 90% dari nilai *Steady State*.

c. Waktu Puncak / *Peak Time* (t_p)\

Adalah waktu yang diperlukan respon untuk mencapai puncak pertama *overshoot*.

d. Waktu Tunak / *Settling Time* (t_s)

Adalah waktu yang dibutuhkan oleh respon untuk mencapai keadaan tunak atau stabil atau dianggap stabil.

2.2.2 Identifikasi Dinamis

Identifikasi dinamis merupakan metode untuk mencari parameter-parameter dengan memberikan masukan pada suatu sistem yang berupa sinyal acak atau tidak beraturan. Identifikasi dinamis dilakukan berdasarkan model pendekatan struktur diskrit. Secara umum pendekatan sistem dilakukan menggunakan model matematis seperti pada persamaan 2.1. [3]

$$y(k)=\frac{B(q^{-1})}{A(q^{-1})}u(k)+\frac{q^{-d}C(q^{-1})}{A(q^{-1})}\eta(k) \qquad \textbf{(2.1)}$$

Di mana :

d (faktor *delay*)

$B(q^{-1})=b_1 q^{-1}+...+ b_{nb}q^{-nb}$ (parameter masukan)

$A(q^{-1})=1+a_1 q^{-1}+...+ a_{na}q^{-na}$ (parameter keluaran)

$C(q^{-1})=c_0+c_1 q^{-1}+...+ c_{nc}q^{-nc}$ (parameter noise)

$q^{-1}y(k)=y(k-i)$; $q^{-1}u(k)=u(k-i)$

Adapun terdapat beberapa bentuk struktur pendekatan parameter antara lain.

a. Struktur *Auto Regressive* (AR)

Pada struktur ini $c_0 \neq 0$; $c_i = 0$ untuk $i = 1,2,3,.$; $a_j \neq 0$ untuk $j = 1,2,3,..$ Dengan demikian bentuk umum dari struktur ini adalah seperti pada persamaan 2.2 – 2.4 berikut.

$$y(k)=\frac{q^{-d}C(q^{-1})}{A(q^{-1})}\eta(k) \quad \textbf{(2.2)}$$

$$y(k)+a_1y(k-1)+...+a_{na}y(k-na)=c_0\eta(k-d) \quad \textbf{(2.3)}$$

$$y(k)=-a_1y(k-1)-...-a_{na}y(k-na)+c_0\eta(k-d) \quad \textbf{(2.4)}$$

b. Struktur *Moving Average* (MA)

Pada struktur ini $c_0 \neq 0$; untuk $i = 1,2,3,..$; $a_j = 0$ untuk $j = 1,2,3,..$ Dengan demikian bentuk umum dari struktur ini adalah seperti pada persamaan 2.5 – 2.6 berikut

$$y(k)=q^{-d}C(q^{-1})\,\eta(k) \quad \textbf{(2.5)}$$

$$y(k)=c_0\eta(k-d)+c_1\eta(k-d-1)+...+c_{nc}\eta(k-d-nc) \quad \textbf{(2.6)}$$

c. Struktur *Auto Regressive Moving Average* (ARMA)

Pada struktur ini $c_0 \neq 0$; untuk $i = 1,2,3,..$; $a_j \neq 0$ untuk $j = 1,2,3,..$ Dengan demikian bentuk umum dari struktur ini adalah seperti pada persamaan 2.7 – 2.9 berikut.

$$y(k)=\frac{q^{-d}C(q^{-1})}{A(q^{-1})}\eta(k) \quad \textbf{(2.7)}$$

$$y(k)+a_1y(k-1)+\ldots+a_{na}y(k-na)=c_0\eta(k-d) + c_1\eta(k-d-1)+....+ c_{nc}\eta(k-d-nc) \quad \textbf{(2.8)}$$

$$y(k)=-a_1y(k-1)-\ldots-a_{na}y(k-na)+c_0\eta(k-d) + c_1\eta(k-d-1)+....+ c_{nc}\eta(k-d-nc) \quad \textbf{(2.9)}$$

d. Struktur *Auto Regressive Exogenous Input* (ARX)

Pada struktur ini $c_0 \neq 0$; $c_i = 0$ untuk $i = 1,2,3,..$; $a_j \neq 0$ untuk $j = 1,2,3,..$; $b_k \neq 0$ untuk $k = 1,2,3,..$ Dengan demikian bentuk umum dari struktur ini adalah seperti pada persamaan 2.10 – 2.12 berikut.

$$y(k)=\frac{b_0}{A(q^{-1})}u(k)+\frac{q^{-d}C(q^{-1})}{A(q^{-1})}\eta(k) \quad \textbf{(2.10)}$$

$$y(k)+a_1y(k-1)+\ldots+a_{na}y(k-na)=b_1u(k-1)+\ldots+b_{nb}u(k-nb)+c_0\eta(k) \quad \textbf{(2.11)}$$

$$y(k)=-a_1y(k-1)-\ldots-a_{na}y(k-na)+b_1u(k-1)+\ldots+b_{nb}u(k-nb)+c_0\eta(k) \quad \textbf{(2.12)}$$

e. Struktur *Moving Average Exogenous Input* (MAX)

Pada struktur ini $c_0 \neq 0$; untuk $i = 1,2,3,.$; $a_j = 0$ untuk $j = 1,2,3,.$; $b_k \neq 0$ untuk $k = 1,2,3,..$ Dengan demikian bentuk umum dari struktur ini adalah seperti pada persamaan 2.13– 2.14 berikut.

$$y(k)=B(q^{-1})\,u(k)+q^{-d}C(q^{-1})\,\eta(k) \quad \textbf{(2.13)}$$

$$y(k)=b_1u(k-1)+\ldots+b_{nb}u(k-nb)+c_0\eta(k-d)+c_1\eta(k-d-1)+....+c_{nc}\eta(k-d-nc) \quad \textbf{(2.14)}$$

f. Struktur *Auto Regressive Moving Average Exogenus Input* (ARMAX)

Pada struktur ini $c_0 \neq 0$; untuk $i = 0,1,2,3,.$; $a_j \neq 0$ untuk $j = 1,2,3,..$ $b_k \neq 0$ untuk $k = 1,2,3,..$ Dengan demikian bentuk umum dari struktur ini adalah seperti pada persamaan 2.15– 2.17 berikut

$$y(k)=\frac{B(q^{-1})}{A(q^{-1})}u(k)+\frac{q^{-d}C(q^{-1})}{A(q^{-1})}\eta(k) \quad \textbf{(2.15)}$$

$$y(k)+a_1y(k-1)+\ldots+a_{na}y(k-na)=b_1u(k-1)+\ldots+b_{nb}u(k-nb)+ c_0\eta(k-d) + c_1\eta(k-d-1)+\mathbf{....}+ c_{nc}\eta(k-d-nc) \quad \textbf{(2.16)}$$

$$y(k)=-a_1y(k-1)-\ldots-a_{na}y(k-na)+b_1u(k-1)+\ldots+b_{nb}u(k-nb)+ c_0\eta(k-d) + c_1\eta(k-d-1)+....+ c_{nc}\eta(k-d-nc) \quad \textbf{(2.17)}$$

2.3 Pemodelan dalam *State-Space*

Vektor *State* merupakan sebuah vektor yang komponen-komponennya merupakan variable *State* suatu sistem. Persamaan *State-Space* merupakan sebuah bentuk matematis dalam matriks dan vektor yang merepresentasikan suatu sistem. Persamaan *State-Space*

secara umum dituliskan pada persamaan 2.18 untuk persamaan *state* dan persamaan 2.19 untuk persamaan *output*. [4]

$$\dot{x}(t)=Ax(t)+Bu(t) \quad \textbf{(2.18)}$$
$$y(t)=Cx(t)+Du(t) \quad \textbf{(2.19)}$$

Dengan matris A merupakan matriks *State*, matriks B merupakan matriks masukan, matriks C merupakan matriks keluaran, dan matriks D merupakan matriks transmisi langsung.

Persamaan *State-Space* di atas merupakan persamaan dalam bentuk waktu kontinyu. Untuk persamaan *State-Space* dalam waktu diskrit dapat dituliskan dalam persamaan 2.20 dan 2.21 berikut.

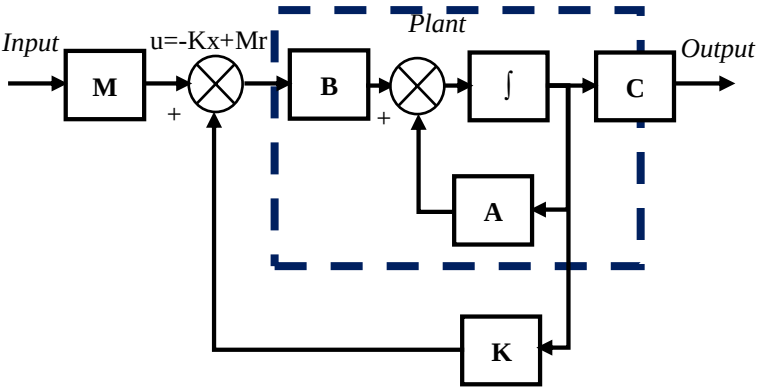
$$x(k+1)=Ax(k)+Bu(k) \quad \textbf{(2.20)}$$
$$y(k)=Cx(k)+Du(k) \quad \textbf{(2.21)}$$

Dengan matris A merupakan matriks *State*, matriks B merupakan matriks masukan, matriks C merupakan matriks keluaran, dan matriks D merupakan matriks transmisi langsung. Vektor $x(k)$ merupakan vektor *State* pada saat $T = k$. Vektor $x(k+1)$ merupakan vektor *State* pada saat $T = k + 1$. Vektor $u(k)$ merupakan vektor masukan.

2.4 Teori Kontrol Optimal

Istilah optimal merupakan suatu hasil yang paling baik yang dapat dicapai dengan memperhatikan kondisi dan kendala dari suatu sistem. Dalam sistem kontrol, istilah optimal sering kali merujuk pada nilai minimal, misalnya meminimalkan bahan bakar, waktu, dan juga kesalahan. [5]

Adapun blok diagram sistem kontrol optimal secara umum terdapat pada Gambar 2.4.



Gambar 2.4 Diagram Blok Kontrol Optimal

Sistem kontrol dapat dikatakan baik adalah sistem kontrol yang mempunyai daya tanggap yang cepat dan stabil. Namun, tidak memerlukan energi yang berlebihan. Sistem kontrol tersebut dapat dicapai dengan melalui pengaturan indeks performansi yang tepat. Ssitem kontrol yang berdasarkan optimasi indeks performansi adalah seperti pada persamaan 2.22 berikut.

$$J = \int_{t_0}^{T} L(x,u,t)\, dt \qquad \textbf{(2.22)}$$

Di mana :
J = Indeks Performansi
L(x,u,t) = Fungsi dari x,u, dan t
T = waktu

Salah satu metode yang biasa digunakan untuk meminimalkan indeks performansi yaitu persamaan Aljabar Riccati yang digunakan untuk mengoptimalkan sistem proses yang berbentuk linear. Suatu sistem kontrol akan optimal pada indeks kerja yang berbeda.

2.4.1 *Linear Quadratic Regulator*

Metode ini digunakan pada sistem LTI (*Linear Time Iinvariant*) dengan persamaan *plant* seperti pada persamaan 2.23 berikut.

$$\dot{x}(t)=Ax(t)+Bu(t) \quad \textbf{(2.23)}$$

Dengan rumus *cost function* adalah seperti pada persmaan 2.24 berikut.

$$J=\frac{1}{2}\int_0^{\infty}[x^T(t)Qx(t)+u^T(t)Ru(t)]\,dt \quad \textbf{(2.24)}$$

Di mana, x(t) adalah vektor *state* orde ke-n, u(t) adalah vektor kontrol orde ke-n, A adalah matriks *state* nxn, B adalah matriks kontrol rxr. Nilai matriks pembobot Q harus bernilai semi-definit postif dan nilai R harus definit positif. Hal ini berarti nilai skalar x^TQx selalu bernilai positif atau nol setiap waktu t untuk semua fungsi x(t) dan nilai skalar u^TRu harus selalu positif setiap waktu dari u(t). Dengan terpenuhinya persyaratan ini maka akan dapat dipastikan nilai J akan bernilai bagus. Pada bagian eigenvalue, nilai Q tidak boleh bernilai non-negatif maka nilai R harus bernilai positif. Jika kedua matriks dipilih berbentuk diagonal berarti semua masukan Q harus bernilai positif dan nilai R juga positif. Berikut merupakan pada Tabel 2.1 terdapat sifat-sifat matriks.

Tabel 2.1 Sifat pada Matriks

A disebut definit positif	$\det(A_i) > 0$
A disebut definit negative	$(-1)^i \det(A_i) < 0$
A disebut semi definit positif	$\det(A_i) \geq 0$
A disebut semi definit negatif	$(-1)^i \det(A_i) \leq 0$

Tujuan utama dari kontrol optimal adalah untuk mencari nilai gain K dengan cara meminimalkan indeks performansi J. Indeks performansi dapat diintrepertasikan sebagai fungsi energi. Jadi, semakin kecil indeks performansi maka akan semakin kecil pula total energi dari sistem loop tertutup. Dengan catatan bahwa *state* x(t) dan *input* u(t) adalah pemberat bagi J jadi apabila diharapkan nilai J keecil maka nilai x(t) dan u(t) tidak boleh terlalu besar. Apabila nilai J diperkecil maka sistem dapat dipastian terbatas namun karena sistem adalah integral tak terbatas dari x(t). Oleh karena itu, dapat diketahui bahwa x(t) akan bernilai nol apabila t bernilai tak terbatas. Hal ini dapat dipastikan bahwa sistem *close loop* akan menjadi stabil.

Nilai dua matriks Q dan R dipilih sesuai dengan desain yang diinginkan. Tergantung dari parameter yang dipilih maka sistem *loop* tertutup akan mempunyai respon yang berbeda-beda. Pada umumnya pemilihan matriks Q dengan nilai yang besar dapat membuat nilai J menjadi lebih kecil oleh karena itu nilai x(t) haruslah bernilai kecil. Sebaliknya dengan memmilih nilai R yang besar maka nilai input u(t) harus tetap kecil untuk menjaga agar nilai J tetap kecil.

Dari uraian di atas dapat disimpulkan bahwa semakin besar nilai Q maka hasil dari *pole* dari sistem *loop* tertutupnya berada pada bagian kiri bidang s. Sehingga nilai *state* meluruh secara lambat. Sebaliknya semakin besar nilai R maka semakin kecil usaha yang digunakan dan endapatkan nilai *state* x(t) yang besar.

Langkah pertama untuk mendesain kontroler LQR adalah memilih matriks bobot Q dan R. Masukan R lebih berat daripada *state* sementara ketika nilai bobt Q *state* lebih dari *input.* Respon sistem *loop* tertutup dapat disimulasikan. LQR kontroler diberikan dengan persamaan 2.25 berikut ini.

$$u = -Kx \qquad \textbf{(2.25)}$$

Di mana nilai K adalah konstanta umpan balik yang diperoleh dari penyelesaian persamaan aljabar Riccati. Apabila elemen-elemen matriks K yang tidak diketahui ditentukan sedemikian rupa maka nilai $u = -Kx$ adalah optimal untuk syarat awal x(0).

Dari indeks persamaan *state-space* sistem dan indeks performansi didapat nilai matriks K yang bernilai optimal dengan perhitungan seperti pada persamaan 2.26 berikut.

$$K=R^{-1}B^{T}P \qquad \textbf{(2.26)}$$

Di mana nilai P adalah unik, yaitu solusi semi definit positif untuk persamaan Riccati harus memenuhi persamaan 2.27 berikut ini.

$$A^{T}P+PA - PBR^{-1}B^{T}P+Q=0 \qquad \textbf{(2.27)}$$

Persamaan di atas merupakan persamaan dari aljabar Riccati. Dalam perancangan teknik kontrol optimal LQR, setelah matriks P

diketahui maka nilai P tersebut disubstitusikan untuk mencari nilai matriks optimal K.

Untuk sistem *non-zero setpoint* apabila nilai keluaran plant diinginkan sama dengan referensi maka perlu dirancang dengan tambahan seperti persamaan 2.28 dan 2.29 berikut.

$$u = -Kx + Mr \tag{2.28}$$

Di mana,

$$M = [C(BK - A)^{-1}B^{-1}] \tag{2.29}$$

2.4.2 *Discrete-Time Linear Quadratic Regulator*

LQR dengan satuan waktu diskrit tidak terlalu berbeda dengan waktu kontinyu. Terdapat banyak cara yang sama digunakan dalam pencarian nilai optimalnya. Untuk sistem diskrit digunakan sistem LTI seperti *plant* dengan persamaan 2.30 berikut.

$$x(k+1) = A(k)x(k) + B(k)u(k) \tag{2.30}$$

Di mana $k = k_0, k_f, \ldots, k_f - 1$, x(k) merupakan vektor *state* orde ke n, u(k) merupakan vektor kontrol orde ke n, dan matriks A dan B sebagai representasi dari *state-space* untuk waktu diskrit. Untuk rumus *terminal cost function* terdapat pada persamaan 2.31 dan 2.32 berikut. [6]

$$J = J(x(k_0), u(k_0), k_0) \tag{2.31}$$

$$J = \frac{1}{2}x'(k_f)F(k_f)x(k_f) + \frac{1}{2}\sum_{k=k_0}^{k_f-1}[x'(k)Q(k)x(k) + u'(k)R(k)u(k)] \tag{2.32}$$

Di mana, $F(k_f)$ dan $Q(k)$ merupakan matriks persegi yang masing-masing mempunyai orde nxn. Dan juga $R(k)$ merupakan positif definit matriks.

Untuk persamaan ajabar Riccati biasa digunakan notasi P dan untuk perhitungannya adalah sebagai berikut dengan diketahui $E(k) = B(k)R^{-1}(k)B'(k)$.

$$P(k)x^*(k)=Q(k)x^*(k)+A'(k)P(k+1)x^*(k+1) \quad \textbf{(2.33)}$$

$$x^*(k+1)=A(k)x^*(k)-E(k)P(k+1)x^*(k+1) \quad \textbf{(2.34)}$$

Setelah dilakukan penyederhanaan rumus maka didapatkan persamaan 2.35 dan 2.36 berikut ini.

$$x^*(k+1)=[I+E(k)P(k+1)]^{-1}A(k)x^*(k) \quad \textbf{(2.35)}$$

$$P(k)x^*(k)=Q(k)x^*(k) + A'(k)P(k+1)[I+E(k)P(k+1)]^{-1}A(k)x^*(k) \quad \textbf{(2.36)}$$

Setelah variabel $x^*(k)$ dieliminasi pada persamaan 2.35 dan 2.36 maka didapatkan persamaan 2.37 dan 2.38 berikut.

$$P(k)=A'(k)P(k+1)[I+E(k)P(k+1)]^{-1}A(k)+Q(k) \quad \textbf{(2.37)}$$

$$P(k)=A'(k)[P(k+1)+E(k)]^{-1}A(k)+Q(k) \quad \textbf{(2.38)}$$

Setelah didapatkan matriks Riccati maka diperoleh *close-loop* kontrol optimal dengan persamaan 2.39 dan 2.40 berikut.

$$u^*(k)=-R^{-1}(k)B'(k)A^{-T}(k)[P(k)-Q(k)]x^*(k) \quad \textbf{(2.39)}$$

$$u^*(k)=-K(k)x^*(k) \quad \textbf{(2.40)}$$

Di mana, matriks umpan balik K disebut dengan penguat *feedback*. Maka, didapatkan *state* optimal dengan mensubstitusikan nilai kontrol optimal $u^*(k)$ seperti persamaan 2.41 berikut.

$$x^*(k+1)=\big(A(k)-B(k)K(k)\big)x^*(k) \quad \textbf{(2.41)}$$

Untuk nilai k_f yang tak terhingga (∞) maka perhitungan PI (*performance index)* dan aljabar Riccati berubah menjadi seperti pada persamaan 2.42 dan 2.43 berikut.

$$J=\frac{1}{2}\sum_{k=k_0}^{\infty}[x^{*'}(k)Qx(k)+u^{*'}(k)Ru(k)] \quad \textbf{(2.42)}$$

$$P(k)=P(k+1)=\bar{P} \quad \textbf{(2.43)}$$

Untuk indeks performansi dapat juga dicari dengan rumus seperti persamaan 2.44 berikut. [6]

$$J^*(k0)=\frac{1}{2}x^{*'}(k0)\bar{P}x^*(k0) \quad \textbf{(2.44)}$$

2.4.3 *Linear Quadratic Regulator* dengan *Servo* Sistem Tipe 1

Seperti yang telah diketahui bahwa kontroler LQR bertujuan untuk meregulasi atau membuat nilai *output* menjadi nol atau mendekati nol dengan *input* seminimal mungkin atau dengan kata lain meminimalkan energi *cost function*.

Untuk mendapatkan nilai *output* yang tetap sesuai dengan *set point* atau referensi yang diberikan dengan input yang seminimal mungkin maka kontroler LQR diberikan tambahan dengan *servo* sistem tipe 1. Pada *servo* sistem tipe 1 *plant* ditambahkan dengan sebuah *gain forward* dan *integrator* untuk *plant* tipe 0 (tanpa *integrator)*. Dengan asumsi $y = x_1$ dan nilai referensi r adalah fungsi *step*. Berikut merupakan persamaan untuk plant dengan *integrator* adalah seperti pada persamaan 2.45 dan 2.46 berikut.

$$u=-Kx+ k_1 r \quad \textbf{(2.45)}$$

$$\dot{x}=Ax+Bu \quad \textbf{(2.46)}$$

Setelah dilakukan subtitusi nilai sinyal kontrol (u) pada persamaan 2.45 ke persamaan 2.46 maka didapatkan persamaan *state* baru seperti pada persamaan 2.47 berikut ini.

$$\dot{x}=(A-BK)x+ Bk_1 r \quad \textbf{(2.47)}$$

Di mana, saat nilai $y(\infty)$ akan mencapai nilai r dan nilai $u(\infty)$ akan mencapai nilai nol maka dapat ditulis dengan persamaan 2.48 seperti berikut.

$$\dot{x}(\infty)=(A-BK)x(\infty)+ Bk_1 r(\infty) \quad \textbf{(2.48)}$$

Setelah dilakukan pengurangan pada kedua persamaan 2.47 dan 2.48 dengan asumsi nilai $r(t) = r(\infty)$ karena referensi merupakan fungsi *step* maka didapatkan persamaan 2.49 seperti berikut.

$$\dot{x}(t)-\dot{x}(\infty)=(A-BK)[x(t)-x(\infty)] \quad \textbf{(2.49)}$$

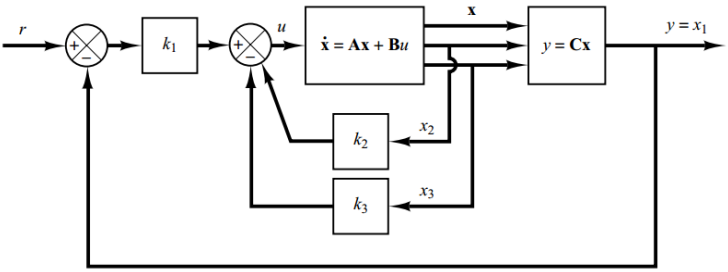
Di mana, nilai *error* dapat didefinisikan dengan persamaan 2.50 berikut ini.

$$x(t)-x(\infty)=e(t) \quad \textbf{(2.50)}$$

Dari persamaan 2.50 maka nilai *error* didapatkan persamaan 2.51 sebagai berikut.

$$\dot{e}=(A-BK)e \quad \textbf{(2.51)}$$

Diagram blok sistemnya menjadi seperti pada Gambar 2.5 berikut ini. [4]

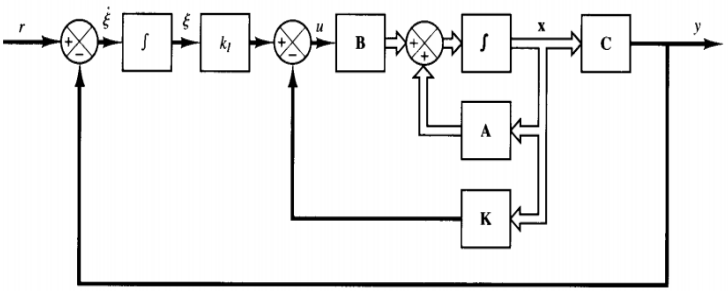


Gambar 2.5 Diagram Blok *Servo* Sistem Tanpa *Integrator*

Untuk desain *servo* sistem dengan *plant* tanpa *integrator* maka persamaannya didapatkan seperti pada persamaan 2.52 – 2.55 berikut ini.

$$\dot{x}=Ax+Bu \quad \textbf{(2.52)}$$
$$y=Cx \quad \textbf{(2.53)}$$
$$u=-Kx+k_1\xi \quad \textbf{(2.54)}$$
$$\dot{\xi}=r-y=r-Cx \quad \textbf{(2.55)}$$

Di mana, ξ merupakan keluaran *integrator* yang bernilai skalar dan juga merupakan nilai selisih antara masukan dan keluaran sistem dan r merupakan referensi sistem. Maka diagram blok *servo* sistem dengan menggunakan integrator terdapat pada Gambar 2.6 berikut ini.



Gambar 2.6 Diagram Blok *Servo* Sistem dengan *Integrator*

Di mana, persamaan sistemnya adalah seperti pada persamaan 2.56 berikut.

$$\begin{bmatrix}\dot{x}(t)\\ \dot{\xi}(t)\end{bmatrix}=\begin{bmatrix}A & 0\\ -C & 0\end{bmatrix}\begin{bmatrix}x(t)\\ \xi(t)\end{bmatrix}+\begin{bmatrix}B\\ 0\end{bmatrix}u(t)+\begin{bmatrix}0\\ 1\end{bmatrix}r(t) \qquad \textbf{(2.56)}$$

Di mana, saat *steady-state* persamaannya menjadi seperti persamaan 2.57 berikut.

$$\begin{bmatrix}\dot{x}(\infty)\\ \dot{\xi}(\infty)\end{bmatrix}=\begin{bmatrix}A & 0\\ -C & 0\end{bmatrix}\begin{bmatrix}x(\infty)\\ \xi(\infty)\end{bmatrix}+\begin{bmatrix}B\\ 0\end{bmatrix}u(\infty)+\begin{bmatrix}0\\ 1\end{bmatrix}r(\infty) \qquad \textbf{(2.57)}$$

Nilai **r** bernilai konstan dikarenakan merupakan fungsi *step* maka $\mathbf{r(t) = r(\infty)}$ maka setelah persamaan 2.56 dikurangkan dengan persamaan 2.57 didapatkan persamaan 2.58 seperti berikut ini. **[4]**

$$\begin{bmatrix}x(t)-\dot{x}(\infty)\\ \xi(t)-\dot{\xi}(\infty)\end{bmatrix}=\begin{bmatrix}A & 0\\ -C & 0\end{bmatrix}\begin{bmatrix}x(t)-x(\infty)\\ \xi(t)-\xi(\infty)\end{bmatrix}+\begin{bmatrix}B\\ 0\end{bmatrix}[u(t)-u(\infty)] \qquad \textbf{(2.58)}$$

Di mana,

$$x_e(t)=x(t)-x(\infty)$$
$$\xi_e(t)=\xi(t)-\xi(\infty)$$
$$u_e(t)=u(t)-u(\infty)$$

Dari persamaan 2.58 maka didapatkan persamaan yang dituliskan seperti pada persamaan 2.59 berikut.

$$\begin{bmatrix}\dot{x_e}(t)\\ \dot{\xi_e}(t)\end{bmatrix}=\begin{bmatrix}A & 0\\ -C & 0\end{bmatrix}\begin{bmatrix}x_e(t)\\ \xi_e(t)\end{bmatrix}+\begin{bmatrix}B\\ 0\end{bmatrix}u_e(t) \quad \textbf{(2.59)}$$

Di mana, nilai sinyal kontrol dari persamaan 2.59 adalah seperti pada persaman 2.60 berikut.

$$u_e(t)=-Kx_e(t)+k_1\xi_e(t) \quad \textbf{(2.60)}$$

Dari persamaan 2.59 dan 2.50 Maka nilai *error* nya didapatkan seperti pada persamaan 2.61 dan untuk persamaan sinyal kontrol dapat dituliskan seperti pada persamaan 2.62 berikut..

$$\dot{e}=\hat{A}e+\hat{B}u_e \quad \textbf{(2.61)}$$

Di mana,

$$\hat{A}=\begin{bmatrix}A & 0\\ -C & 0\end{bmatrix};\ \hat{B}=\begin{bmatrix}B\\ 0\end{bmatrix}$$

$$u_e=-\hat{K}e \quad \textbf{(2.62)}$$

Di mana, didapatkan persamaan 2.63 sebagai *gain feedback* dan *gain integrator*

$$\hat{K}=[K \quad -k_1] \quad \textbf{(2.63)}$$

Sehingga nilai *gain feedback* nya didapatkan seperti cara di atas dan untuk sistem waktu diskrit menggunakan cara yang sama hanya mengubah besaran sampel waktu yang biasanya dinotasikan dengan n atau k.

2.4.4 *Linear Quadratic Gaussian* (LQG)

Persamaan *state plant* dan *output* pengukuran diberikan dalam persamaan 2.64 dan 2.65 berikut. [7]

$$\dot{x}=Ax+Bu+Gw \quad \textbf{(2.64)}$$

$$y=Cx+v \quad \textbf{(2.65)}$$

Dengan $x(t) \in R$, $u(t)$ sebagai *input* kontrol, $w(t)$ sebagai *disturbance* atau *noise* yang terdapat pada sistem, dan $v(t)$ sebagai *noise* pengukuran. Dengan anggapan bahwa persamaan 2.66 adalah persamaan *control state feedback.*

$$u=-Kx+r \quad \textbf{(2.66)}$$

Dengan $r(t)$ sebagai *input* referensi dan *gain state feedback* K dapat diperoleh dengan menggunakan teknik LQR atau yang lain. Maka dapat diperoleh persamaan 2.67 berikut.

$$\dot{x}=(A-BK)x+Br+Gw \quad \textbf{(2.67)}$$

Desain *full-state feedback* menjamin sistem *closed-loop* stabil. Hal ini disebabkan karena seluruh pole dari $(A-BK)$ dapat ditempatkan pada lokasi yang diinginkan. Kerugian menggunakan *full-state feedback* adalah dalam implementasinya tidak praktis bahkan kadangkala tidak mungkin karena biasanya tidak semua *state* sistem dapat diukur.

Jadi, implementasi hukum kontrol dengan menggunakan *state feedback* diperlukan sebuah *observer* yang berguna untuk melakukan estimasi *state* yang tidak terukur atau *state* terukur tetapi hasil pengunkurannya terkontaminasi *noise* atau biasa disebut *noisy measurement.*

Persamaan *state observer* atau Kalman filter adalah seperti pada persamaan 2.68 berikut.

$$\dot{x}=(A-LC)x+Bu+Ly \quad \textbf{(2.68)}$$

Gain filter L dapat diperoleh dengan menggunakan teknik *prediction observer*. Jika desain filter Kalman digunakan, gain filter

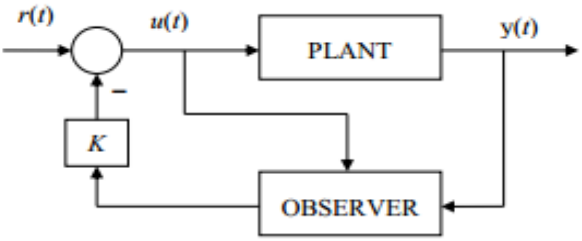
secara mudah dapat diperoleh dengan menggunakan ARE (*Algebraic Riccati Equation*).

Dikarenakan tidak seluruh *state* dapat terukur, maka persamaan sinyal kontrol tidak dapat diimplementasikan. Dalam praktek, state feedback menggunakan state estimasi sehingga didapatkan persamaan 2.69 berikut.

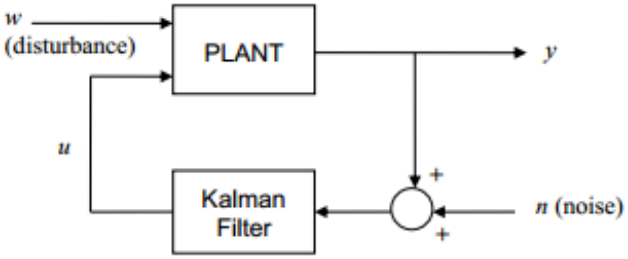
$$u=-K\hat{x}+r \qquad \textbf{(2.69)}$$

Struktur *closed-loop* dengan menggunakan kontroler ini terdapat pada Gambar 2.7. *Gain feedback* K dan *gain observer* L dapat menggunakan teknik-teknik yang menjamin sifat-sifat dari $(A - BK)$ dan $(A - LC)$.

Linear Quadratic Gaussian (LQG) merupakan teknik *state-space* untuk mendesain regulator dinamik optimal. Diagram LQG terdapat pada Gambar 2.8. Objektif desain LQG sama dengan LQR yaitu untuk regulasi output y mendekati nol. Plant diberi gangguan *w* dan pada pengukuran output *y* terdapat *noise v*.



Gambar 2.7 *Regulator* Dinamik menggunakan *Observer* dan *Full-State Feedback*



Gambar 2.8 Diagram *Regulator* Dinamik LQG

Pada Gambar 2.7 dan Gambar 2.8 dapat terlihat perbedaan antara *observer* dan Kalman filter yaitu jika pada *observer* tidak terdapat *disturbance* dan *noise* pengukuran jadi sistem dapat diasumsikan ideal. Namun, pada Kalman filter berlaku sebalikya jadi terdapat *disturbance* dan *noise* pengukuran pada modelnya.

2.4.5 *Discrete-Time Linear Quadratic Gaussian*

Dalam desain LQG untuk sistem waktu diskrit, state dinyatakan dalam persamaan 2.70 dan 2.71 berikut. [7]

$$x_{k+1}=Ax_k+Bu_k+Gw_k \quad \textbf{(2.70)}$$

$$y_k=Cx_k+v_k \quad \textbf{(2.71)}$$

Filter Kalman *steady state* ditulis dalam bentuk reskursi priori dengan persamaan 2.72 dan 2.73 berikut ini.

$$\hat{x}_{k+1}^{-}=Ax_k+Bu_k+AK_f(y_k-C\hat{x}_k) \quad \textbf{(2.72)}$$

atau rekursi priori

$$\hat{x}_{k+1}=Ax_k+Bu_k+AK_f[y_k-C(Ax_k+Bu_k)] \quad \textbf{(2.73)}$$

Di mana, K_f adalah *gain* Kalman *steady-state.* Estimasi *state* priori x_k^- tidak menggunakan pengukuran y_k sedangkan rekursi

posteriori $\hat{x}_k$ menggunakan pengukuran y_k. Maka, kontrol *feedback* dapat dinyatakan pada persamaan 2.74 dan 2.75 berikut.

$$u_k=-K\hat{x}_k^-+r_k \quad \textbf{(2.74)}$$

atau

$$u_k=-K\hat{x}_k+r_k \quad \textbf{(2.75)}$$

Dengan catatan bahwa *gain filter diskrit* adalah $L = KA_f$. *Gain feedback* K dan *gain observer* dapat dipilih agar memberikan perilaku yang cocok dari $(A - BK)$ dan $(A - LC)$. Dalam desain LQG digital, *gain feedback* dipilih menggunakan ARE LQR diskrit dan *gain observer* dipilih dengan menggunakan ARE filter Kalman diskrit.

Untuk persamaan desain *state feedback* LQR adalah seperti pada persamaan 2.76 dan 2.77 berikut.

$$0=A^TPA-P+Q-A^TPB\left(R+B^TPB\right)^{-1}B^TPA \quad \textbf{(2.76)}$$

$$K=\left(R+B^TPB\right)^{-1}B^TPA \quad \textbf{(2.77)}$$

Untuk persamaan desain *gain filter* Kalman adalah seperti pada persamaan 2.78 dan 2.79 berikut.

$$0=APA^T-P+GQ_fG^T-APC^T\left(CPC^T+R_f\right)^{-1}CPA^T \quad \textbf{(2.78)}$$

$$L=APC^T\left(CPC^T+R_f\right)^{-1} \quad \textbf{(2.79)}$$

Di mana, Q_f dan R_f adalah matriks kovarian *noise*. Untuk desain regulator dinamik LQG adalah seperti pada persamaan 2.80 dan 2.81 berikut.

$$\hat{x}_{k+1}^-=Ax_k^-+Bu_k+L(y_k-C\hat{x}_k^-) \quad \textbf{(2.80)}$$

$$u_k=-K\hat{x}_k+r_k \quad \textbf{(2.81)}$$

2.5 *Swarm Intelligence*

Swarm bisa diartikan kawanan, kelompok, kerumunan, gerombolan, atau koloni. Sehingga *Swarm Intelligence* berarti kecerdasan yang dihasilkan dari adanya tingkah laku kawanan atau kelompok. Sebagai contoh, seekor semut yang sendirian tidak memiliki kecerdasan yang luar biasa. Tetapi, sekawanan semut yang terdiri dari ratusan atau bahkan ribuan ekor semut, dengan komunikasi dan kerjasama yang baik melalui zat yang disebut *pheromone*, dapat secara cepat menemukan jalur terpendek antara sumber makanan dan sarang mereka. [8]

Istilah *Swarm Intelligence* (SI) digunakan untuk suatu disiplin ilmu yang berhubungan dengan alam dan sistem-sistem buatan yang tersusun atas banyak individu yang berkoordinasi menggunakan kontrol terdesentralisasi dan dapat mengorganisasi diri sendiri (*self-organization*). Secara khusus, disiplin ini focus pada tingkah laku kolektif yang dihasilkan dari interaksi lokal antara satu individu dengan individu lain dan antara individu dengan lingkungannya. *Swarm Intelligence* juga mempelajari tingkah laku social seperti kawanan burung yang terbang berduyun-duyun (*bird flocking*) atau gerombolan ikan yang berenang berkelompok (*fish schooling*), dan sekelompok binatang daratan yang bergerak dalam aturan tertentu (*herds of land animals*). Beberapa sistem buatan manusia juga termasuk ke dalam domain SI, seperti sistem *multi-robot* dan program-program komputer tertentu yang dibangun untuk menyelesaikan permasalahan optimasi dan analisa data.

SI memiliki karakter multidisipln yang kuat karena sistem-sistem dengan karakteristik di atas dapat diamati dalam suatu varietas domain. Riset dalam SI dapat diklasifikasikan berdasarkan kriteria berbeda.

Klasifikasi yang pertama ini membagi riset SI ke dalam dua area berdasarkan natural dan tidaknya sistem yang dianalisa. Riset SI Natural mempelajari sistem-sistem biologis, sedangkan SI Artifisial mempelajari hal-hal buatan manusia.

Klasifikasi yang kedua dilakukan berdasarkan tujuannya yaitu ilmiah atau rekayasa (*engineering*). Pada tujuan ilmiah, riset dilakukan untuk memodelkan sistem SI dan memahami mekanisme-mekanisme yang mengizinkan suatu sistem, sebagai suatu sistem yang utuh, untuk bertingkah laku dalam suatu cara yang terkoordinasi sebagai hasil interaksi lokal antara individu dengan individu lain dan

antara individu dengan lingkungannya. Seballiknya, untuk tujuan rekayasa, riset dilakukan untuk mengeksploitasi pemahaman yang dihasilkan dari riset bertujuan untuk merancang sistem-sistem yang mampu memecahkan permasalahan praktis.

Sistem SI biasanya memiliki empat sifat seperti berikut:

a. Tersusun atas banyak individu
b. Individu-individu penyusunnya relatif homogen (yaitu: semua individunya identik atau terdiri dari beberapa jenis saja)
c. Interaksi-interaksi antar individu terjadi berdasarkan aturan tingkah laku sederhana yang hanya memanfaatkan informasi lokal di mana pertukaran individu terjadi secara langsung atau melalui lingkungannya
d. Tingkah laku sistem secara keseluruhan dihasilkan dari interaksi antara individu dengan individu dan antara individu dengan lingkungannya, yaitu mengorganisasi diri dalam tingkah laku berkelompok.

Sistem SI mampu melakukan aksi daam suatu cara yang terkoordinasi tanpa hadirnya suatu coordinator atau pengontrol eksternal. Tingkah laku setiap individu digambarkan secara probabilistik: setiap individu memiliki suatu tingkah laku yang bergantung pada persepsi lokal pada individu-individu yang menjadi tetangganya.

2.5.1 *Particle Swarm Optimization* (PSO)

PSO merupakan teknik optimasi berbasis populasi yang dikembangkan oleh James Kennedy dan Russ Eberhart pada tahun 1995. Teknik ini terinspirasi oleh tingkah laku sosial pada kawanan burung yang terbang berduyun-duyun (*bird flocking*) atau gerombolan ikan yang berenang berkelompok (*fish schooling*). Semisal, kawanan burung bangau, dalam jumlah sangat banyak, bisa terbang membentuk formasi tertentu tanpa bertabrakan satu sama lain. Dan juga gerombolan ikan yang berjumlah ribuan bisa bergerak sangat cepat tanpa tabrakan meskipun jarak antar ikan begitu dekat. Burung maupun ikan ini memeiliki kecerdasan yang luar biasa sehingga bisa menjaga jarak tetap stabil dengan mengatur kecepatan terbang atau berenangnya. Kecerdasan seperti inilah yang diadopsi oleh ke dua ilmuwan tersebut untuk membangun suatu teknik

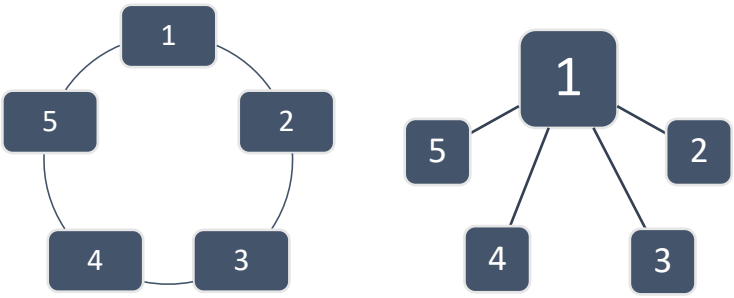
optimasi, yang mereka namakan *Particle Swarm Optimization* (optimasi kawanan partikel). Mereka menyebutnya partikel karena pada dasarnya banyak makhluk hidup yang memliki kecerdasan seperti burung dan ikan, misalnya belalang, lebah, dan sebagainya.

PSO memiliki banyak kesamaan dengan teknik-teknik *evolutionary computation* yang lain, seperti *Genetic Algorithm* (GA), *Evolutonary Strategies* (ES), dan sebagainya. PSO maupun GA dimulai dengan suatu populasi yang terdiri dari sejumlah individu (yang menyatakan solusi) yang dibangkitkan secara acak dan selanjutnya melakukan pencarian solusi optimum melalui perbaikan individu untuk sejumah generasi tertentu. Tetapi, berbeda dengan GA, PSO tidak menggunakan operator-operator evolusi seperti rekombinasi (*crossover*) dan mutasi. PSO memiliki *memory* untuk menyimpan solusi terbaik, sedangkan pada GA bisa mati dan digantikan oleh individu baru. Pada PSO, posisi dan kecepatan terbang partikel di-*update* pada setiap iterasi sehingga partikel tersebut bisa menghasilkan solusi baru yang lebih baik.

Pada PSO, solusi-solusi (partikel) yang potensial, "terbang" di dalam ruang masalah mengikuti partikel-partikel yang optimum saat ini (*current optimum particle*). Dengan konsep ini, PSO lebih mudah diimplementasikan dan parameter yang harus disetel hanya sedikit. PSO telah berhasil diaplikasikan pada banyak area, seperti optimasi fungsi, pelatihan *artificial neural network* (ANN), *fuzzy system control*, dan area-area lainnya.

Untuk konsep dasar dari PSO adalah dengan simulasi sebagai berikut. Misalkan terdapat skenario seperti: sekelompok burung secara acak sedang mencari makanan pada suatu area. Hanya terdapat satu potong makanan di area tersebut. Semua burung tidak tahu di mana lokasi makanan sebenarnya. Tetapi, mereka tahu berapa jauh makanan tersebut pada setiap iterasi. Cara paling efektif adalah mengikuti burung yang paling dekat dengan makanan. Maka PSO belajar dari skenario tersebut untuk menyelesaikan permasalahan optimasi. Pada PSO, satu "burung" menyatakan satu solusi di dalam ruang masalah. Istilah "burung" bisa diganti juga dengan yang lebih umum yaitu "partikel". Setiap partikel memliki nilai *fitness* yang dievaluasi menggunakan fungsi *fitness* untuk dioptimasi. Setiap partikel juga memiliki vektor *velocity* yang berupa kecepatan dan arah "terbang". Partikel terbang di dalam ruang masalah mengikuti partikel-partikel yang optimum saat ini.

Untuk menggambarkan hubungan suatu partikel dengan partikel lainnya, terapat dua topologi dasar PSO yang digunakan di dalam banyak literatur, yaitu: *Ring Topology* dan *Star Topology*. Pada *ring topology*, suatu partikel terhubung pada dua partikel lainnya, sehingga memiliki ukuran ketetanggaan (*neighborhood*) sama dengan 3. Sedangkan pada *star topology*, suatu partikel bisa terhubung pada semua partikel lainnya, sehingga disebut *global neighborhood*. Untuk topologi dari keduanya dapat dilihat pada Gambar 2.9. [8]



Gambar 2.9 Topologi PSO : *Ring Topology* (Kiri) dan *Star Topology* (Kanan)

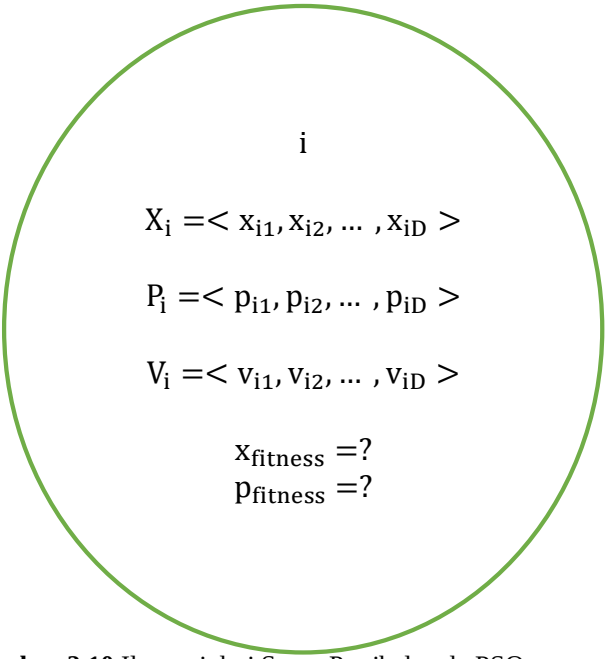
Pada PSO standar, suatu partikel (solusi) tersusun atas tiga buah vektor dan dua nilai *fitness*. Seperti pada Gambar 2.10

Untuk tiga buah vektornya antara lain:

- **X** : yaitu vektor yang menyimpan posisi partikel saat ini di dalam ruang pencarian;
- **P** : yaitu vektor untuk menyimpan posisi dari solusi terbaik sejauh ini yang ditemukan oleh partikel tersebut; dan
- **V** ; yaitu vektor berisi gradien (arah) yang menyatakan ke mana partikel akan "terbang".

Untuk dua nilai *fitness*-nya antara lain:

- ***x-fitness*** : menyimpan nilai *fitness* dari x-vektor;
- ***p-fitness*** : menyimpan nilai *fitness* dari p-vektor.



Gambar 2.10 Ilustrasi dari Suatu Partikel pada PSO

PSO dimulai dengan sekumpulan partikel (solusi) yang dibangkitkan secara acak. Setiap partikel kemudian dievaluasi kualitasnya menggunakan fungsi *fitness*. Selanjutnya, partikel-partikel akan terbang mengikuti partikel yang optimum. Pada setiap generasi, setiap partikel di-*update* mengikuti dua nilai "terbaik". Yang pertama adalah *fitness* terbaik yang dicapai oleh satu partikel saat ini. Nilai *fitness* terbaik ini dilambangkan dengan *p* dan disimpan di *memory*. Sedangkan nilai "terbaik" yang ke dua adalah *fitness* terbaik yang dicapai oleh semua partikel dalam topoolgi ketetanggaan. Indeks *g* digunakan untuk menunjuk partikel dengan *fitness* terbaik tersebut. Jika menggunakan topologi ketetanggaan yang berupa *ring topology*, maka cara ini disebut sebagai PSO versi global. Tetapi, jika topologi ketetanggaannya berupa *star topology* maka cara ini disebut PSO versi lokal.

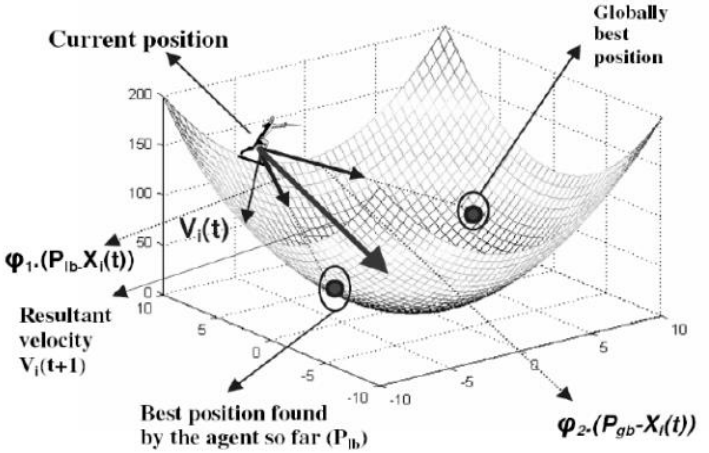
Setelah menemukan dua nilai "terbaik", suatu partikel *i* pada posisi X_i meng-*update* vektor *velocity* dan kemudian meng-*update* posisinya menggunakan persamaan 2.82 dan 2.83 berikut.

$$v_{id}=v_{id}+\varphi_1*r*(p_{id}-x_{id})+\varphi_2*r*(p_{gd}-x_{id}) \quad \textbf{(2.82)}$$

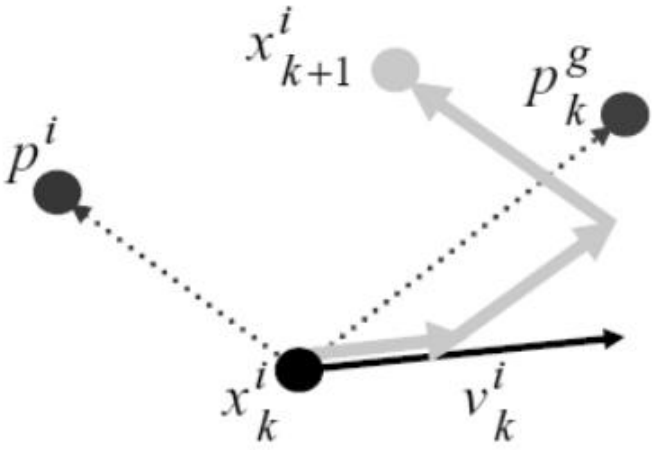
$$x_{id}=x_{id}+v_{id} \quad \textbf{(2.83)}$$

Di mana,
i = partikel ke-i;
d = dimensi ke-d;
φ_1= laju belajar (*learning rates*) untuk komponen *cognition* (kecerdasan individu);
φ_2= laju belajar (*learning rates*) untuk komponen *social* (hubungan sosial antar individu);
p = vektor nilai *fitness* terbaik yang dihasilkan sejauh ini;
g = indeks dari partikel dengan *fitness* terbaik di dalam topologi ketetanggaan; dan
r = bilangan acak dalam interval [0,1].

Berikut merupakan ilustrasi pergerakan partikel pada PSO terdapat pada Gambar 2.11 dan 2.12.



Gambar 2.11 Ilustrasi Perubahan Kecepatan pada PSO



Gambar 2.12 Ilustrasi Perubahan Posisi pada PSO

PSO memliki beberapa parameter untuk diatur-atur (disetel), antara lain:

a. Jumlah partikel

Biasanya disetel antara 20 sampai 40. Tetapi, untuk sebagian besar masalah, 10 partikel sudah cukup besar untuk mendapatkan hasil yang bagus. Jika jumlah partikel yang terlalu kecil bisa terjebak pada optimum lokal meskipun waktu prosesnya sangat cepat.

b. Dimensi partikel

Hal ini bergantung pada masalah yang akan dioptimasi.

c. Rentang nilai dari partikel

Hal ini juga bergantung pada masalah yang akan dioptimasi.

d. Vmax

Variabel ini yang menentukan perubahan maksimum yang bisa dilakukan oleh suatu partikel dalam satu iterasi. Biasanya nilai *Vmax* diset sama dengan rentang nilai partikel.

e. Learning rate atau laju pembelaran

Biasanya disetel sama menjadi $\varphi_1 = \varphi_2 = 2$. Tetapi para ahli terkadang menggunakan laju belajar lain. Pada bukunya, Carlisle menyarankan $\varphi_1 = 2{,}8$ dan $\varphi_2 = 1{,}3$, tetapi dengan sedikit modifikasi pada rumus peng-*update*-an *velocity*-nya.

f. Kondisi berhenti

PSO bisa dihentikan sampai sejumlah iterasi tertentu atau sampai tingkat kesalahan tertentu yang diinginkan oleh *user*.

2.5.2 PSO Modifikasi

Beberapa ahli telah melakukan modifikasi terhadap PSO diantaranya adalah Y. Shi dan Russ Eberhart, yang pada tahun 1998 mengusulkan adanya faktor lain yang mereka sebut bobot *inertia* (kelembaman). Shi menyatakan bahwa rumus peng-*update*-an *velocity* yang terlalu terjebak pada optimum lokal.

Shi mengusulkan penambahan bobot kelembaman yang diberi notasi ω, pada rumus peng-*update*-an *velocity* sehingga menjadi seperti pada persamaan 2.84 berikut.

$$v_{id}=\omega*v_{id}+\varphi_1*r*\left(p_{id}-x_{id}\right)+\varphi_2*r*\left(p_{gd}-x_{id}\right) \quad \textbf{(2.84)}$$

Pada tahun 1999, Maurice Clerc mengembangkan suatu koefisien batasan yang diberi notasi K, untuk membatasi dan mengontrol *velocity*, sehingga rumus peng-*update*-an *velocity* dimodifikasi menjadi seperti pada persamaan 2.85 berikut.

$$v_{id}=K[v_{id}+\varphi_1*r*\left(p_{id}-x_{id}\right)+\varphi_2*r*\left(p_{gd}-x_{id}\right)] \quad \textbf{(2.85)}$$

Di mana,

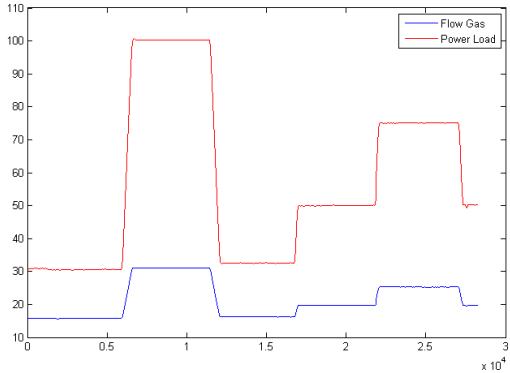
$$K=\frac{2}{|2-\varphi-\sqrt{\varphi^2-4\varphi}|}$$

BAB 3
PERANCANGAN DAN IMPLEMENTASI SISTEM

Pada bab ini, dibahas mengenai beberapa cara untuk merancang sistem kontrol optimal dan juga implementasinya.

3.1 Pengambilan Data

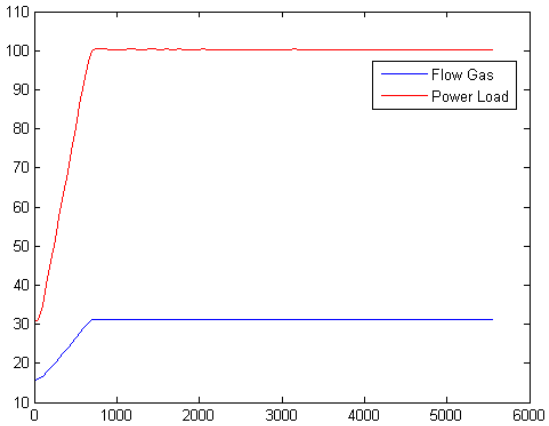
Pengambilan data dilakukan secara *realtime* menggunakan komputer klien yang ada pada Pembangkit Listrik Tenaga Gas (PLTG) di PT. Indonesia Power UP Grati Pasuruan. Data *real* yang diambil adalah data masukan berupa *Flow Gas* dan data keluaran pembangkit berupa daya yang terbangkit dengan sampel waktu sebesar 50 detik. Data *real* yang diambil dilakukan pada tanggal 17 Juni 2016 pukul 8.00-15.32 WIB. Untuk hasil *plot* data dapat dilihat pada Gambar 3.1 berikut.



Gambar 3.1 Hasil *Plot* Data Masukan dan Keluaran PLTG

Kemudian, didapatkan sebanyak 566 data berupa *file* dalam bentuk excel yang nantinya datanya dibaca pada *workspace* di Matlab. Data yang dalam bentuk excel ini selanjutnya diletakkan pada *directory* program Matlab agar nantinya dapat terbaca Matlab dan menjadi variabel pada jendela *Variable Editor*. Dengan data nilai aliran gas dalam bentuk variabel bernama "Flow" dan data nilai keluaran daya yang terbangkit dalam bentuk variabel bernama "PowerLoad". Variabel ini terdapat pada *workspace* pada *software* Matlab.

Untuk membandingkan hasil identifikasi digunakan pada data saat terjadi transien pada *set point* tertentu atau nilai *Power Load* mencapai 100MW maka didapatkan sebanyak 112 data. Untuk hasil *plot* data dapat dilihat pada Gambar 3.2 berikut.



Gambar 3.2 Hasil *Plot* Data Masukan dan Keluaran PLTG saat 100MW

3.2 Pembuatan Program Identifikasi ARX dan *State-Space*

Pada *software* Matlab terdapat fungsi ARX langsung dengan memasukkan nilai data yang ingin diolah dan menentukan banyaknya *order* yang ingin digunakan. Maka pertama, membuat data objek dari kedua data yang diperoleh tadi yaitu, data masukan dan data keluaran. Dan untuk penentuan *order* parameternya dengan memasukkan nilai 3 untuk order parameternya dan nilai 1 untuk faktor *delay* yang digunakan pada pemodelan sistem.

Setelah didapatkan nilai model diskrit ARX maka model diubah menjadi bentuk *state-space* agar dapat diberlakukan kontrol optimal pada model sistem. Sehingga model dapat optimal. Dalam hal ini *state-space* dalam satuan waktu diskrit.

Algoritma pemrograman pada Matlab untuk identifikasi model tersebut adalah sebagai berikut.

1. Inisialiasasi parameter yang diperlukan dalam pengolahan data.
2. Baca data masukan serta keluaran.
3. Membuat data objek dengan data masukan serta keluaran PLTG yang ada di dalamnya.
4. Mengolah data objek menggunakan fungsi ARX dengan menentukan nilai *order* dari parameternya.
5. Model ARX didapatkan dengan *order* parameter sesuai dengan yang ditentukan.
6. Untuk mengetahui nilai varian *noise* sistem dapat dengan melihat pada *workspace* model ARX, setelah itu dapat dilihat pada nilai *noise variance.*
7. Merubah model ARX yang telah didapat menjadi model *state-space* dengan satuan waktu diskrit.
8. Kemudian ditentukan *initial state* sebagai kondisi mula dari masing-masing *state*.

Kemudian dapat diperoleh program Matlab dengan fungsi perintah ARX. Dengan begitu model ARX yang didapatkan adalah seperti pada persamaan 3.1 – 3.3 berikut.

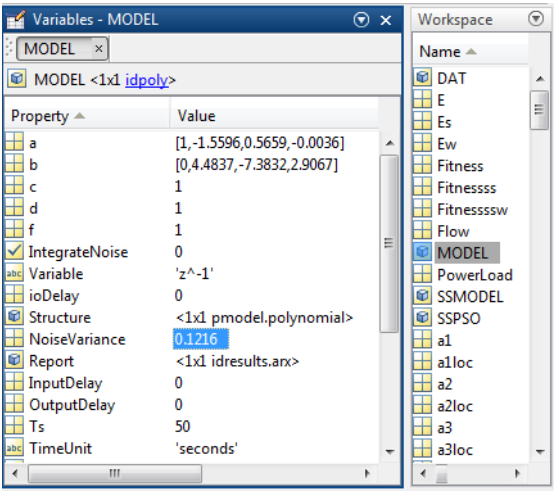
$$A(z)y(t)=B(z)u(t)+e(t) \quad \textbf{(3.1)}$$

Di mana,

$$A(z)=1-1.56\ z^{-1}-0.5659\ z^{-2}-0.00362\ z^{-3} \quad \textbf{(3.2)}$$

$$B(z)=4.484\ z^{-1}-7.383\ z^{-2}+2.907\ z^{-3} \quad \textbf{(3.3)}$$

Dan untuk nilai y(t) merupakan nilai keluaran PLTG sedangkan nilai u(t) merupakan nilai masukan PLTG serta nilai e(t) merupakan *noise* dari sistem dengan varian sebesar 0.1216. Seperti terlihat pada Gambar 3.3 berikut.



Gambar 3.3 Cara Mengetahui Nilai Varian dari *Noise* Sistem

Kemudian model ARX di atas diubah menjadi model *state-space* dalam satuan waktu diskrit dengan perintah ss(). Setelah itu, persaman modelnya didapatkan seperti pada persamaan 3.4 dan 3.5 berikut ini.

$$\dot{x}=\begin{bmatrix}1.56 & -0.5659 & 0.05792\\ 1 & 0 & 0\\ 0 & 0.0625 & 0\end{bmatrix}x+\begin{bmatrix}8\\0\\0\end{bmatrix}u \qquad \textbf{(3.4)}$$

$$y=[0.5605 \quad -0.9229 \quad 5.813]\ x \qquad \textbf{(3.5)}$$

Untuk mendapatkan persamaan *state-space* dengan tambahan *disturbance* dapat dengan fungsi perintah idss(). Kemudian persaman modelnya didapatkan seperti pada persamaan 3.6 dan 3.7 berikut ini.

$$\dot{x}=\begin{bmatrix}0 & 0 & 0.05792\\ 0.0625 & 0 & -0.5659\\ 0 & 1 & 1.56\end{bmatrix}x+\begin{bmatrix}5.813\\-0.9229\\0.5605\end{bmatrix}u+\begin{bmatrix}0.00724\\-0.07073\\0.195\end{bmatrix}w \qquad \textbf{(3.6)}$$

$$y=[0 \ \ 0 \ \ 8]\ x+[0]\ u \qquad \textbf{(3.7)}$$

Meskipun matriks A, B, dan C nilainya berubah namun hasil keluarannya tetap seperti dengan keluaran dari persamaan 3.6 dan 3.7 dengan tambahan *disturbance*.

Kemudian dengan mengubah banyaknya *sample* data menjadi 112 *sample*. Didapatkan model ARX yang berbeda seperti pada persamaan 3.8-3.10.

$$A(z)y(t)=B(z)u(t)+e(t) \quad \textbf{(3.8)}$$

Di mana,

$$A(z)=1-1.551\ z^{-1}-0.8274\ z^{-2}-0.182\ z^{-3} \quad \textbf{(3.9)}$$

$$B(z)=2.538\ z^{-1}-2.934\ z^{-2}+0.6995\ z^{-3} \quad \textbf{(3.10)}$$

Kemudian persaman *state-space* modelnya didapatkan seperti pada persamaan 3.11 dan 3.12 berikut ini.

$$\dot{x}=\begin{bmatrix}1.551 & -0.8274 & 0.364\\ 1 & 0 & 0\\ 0 & 0.5 & 0\end{bmatrix}x+\begin{bmatrix}\mathbf{2}\\ 0\\ 0\end{bmatrix}u \quad \textbf{(3.11)}$$

$$y=[0.5605 \quad -0.9229 \quad 5.813]\ x \quad \textbf{(3.12)}$$

Kemudian dengan tambahan *disturbance* persaman modelnya didapatkan seperti pada persamaan 3.13 dan 3.14 berikut ini.

$$\dot{x}=\begin{bmatrix}0 & 0 & 0.364\\ 0.5 & 0 & -0.8274\\ 0 & 1 & 1.551\end{bmatrix}x+\begin{bmatrix}0.6995\\ -1.467\\ 1.269\end{bmatrix}u+\begin{bmatrix}0.182\\ -0.4173\\ 0.7755\end{bmatrix}w \quad \textbf{(3.13)}$$

$$y=[0 \ \ 0 \ \ 2]\ x+[0]\ u \quad \textbf{(3.14)}$$

Setelah diperoleh model dari sistem tersebut dengan nilai varian dari *disturbance* diketahui sebesar 0,0621. Maka dilanjutkan pada proses selanjutnya yaitu dengan menambahkan algoritma PSO pada identifikasi model kemudian dilanjutkan proses optimalisasi dengan menggunakan kontrol optimal.

3.3 Penerapan Algoritma PSO pada Identifikasi Sistem

Setelah didapatkan pemodelan sistem menggunakan fungsi ARX. Kemudian ditambahkan algoritma PSO untuk mendapatkan nilai *fitness* yang lebih baik dengan mengubah nilai-nilai koefisien parameter *lag* dari model ARX yang sudah didapatkan.

Secara singkat, *pseudocode* algoritma PSO dapat dituliskan seperti pada Gambar 3.4 berikut.

```
for setiap partikel
      Inisialisasi partikel
end

repeat
    for setiap partikel
       Hitung nilai fitness
         if nilai fitness baru lebih baik
            daripada nilai fitness lama

             Update nilai fitness partikel

         end
    end

    Pilih partikel dengan nilai fitness
    terbaik diantara semua partikel
    tetangganya dan simpan nilai fitness
    tersebut

    for setiap partikel
       Hitung velocity partikel
       Update posisi partikel
    end

until (KriteriaBerhenti = true)
```

Gambar 3.4 *Pseudocode* Algoritma PSO

Perubahan nilai-nilai koefisien ini bersifat acak dengan syarat batas yang telah diberikan pada algoritma PSO. Setelah mendapatkan

nilai acak tiap koefisien parameter maka hasilnya akan dievaluasi dengan nilai *fitness* nya. Koefisien parameter dengan nilai *fitness* terbaik akan disimpan hingga proses perhitungan berhenti. Proses *looping* PSO akan berhenti jika nilai *fitness* dari koefisien parameter telah melebihi nilai *fitness* yang diinginkan. Dalam hal ini, lebih dari nilai *fitness* hasil identifikasi ARX sebelumnya.

Untuk inisialisasi parameter yang dibutuhkan adalah sebagai berikut:

a. Jumlah Partikel

Jumlah partikel didefinisikan sebanyak sepuluh. Hal ini mengacu pada dasar teori yang menyebutkan bahwa nilai 10 partikel sudah cukup besar untuk mendapatkan hasil yang bagus.

b. Dimensi Partikel

Dimensi partikel didefiniskan sebanyak 6 buah. Hal ini dikarenakan sesuai dengan banyaknya parameter *lag* pada model ARX.

c. Rentang nilai dari partikel

Untuk rentang nilai bernilai acak dengan bilangan ± 0.02 untuk parameter *lag* output model dan ± 0.025 untuk parameter *lag* input model.

d. Vmax

Untuk kecepatan maksimal juga bernilai acak dengan bilangan ± 0.05.

e. Learning Rate

Untuk laju pembelajaran menggunakan yang umumnya digunakan dalam algoritma PSO yaitu $\varphi_1 = \varphi_2 = 2$.

f. Iterasi

Untuk iterasi peng-*update*-an posisi dibatasi hingga mencapai iterasi ke-25.

g. Kondisi Berhenti

Untuk kondisi berhentinya PSO ditentukan saat nilai *fitness* model PSO lebih besar dari nilai *fitness* model ARX.

Dengan program yang telah dibuat sesuai *pseudocode* pada Gambar 3.4 dan inisialisasi parameter yang telah ditentukan maka akan didapatkan model PSO dengan nilai *fitness* yang lebih bagus daripada model ARX.

Kemudian agar dapat dikontol menggunakan kontrol optimal maka model PSO diubah menjadi model *state-space* ditambah dengan *disturbance* sebagai *noise* sistem.

3.4 Perancangan Kontroler menggunakan *Linear Quadratic Gaussian* (LQG)

Kontroler ini berguna untuk menjaga kesetimbangan perbandigan antara bahan bakar dan udara sesuai dengan *set point*. Dengan adanya kontroler ini maka diharapkan pada saat beban bertambah keseimbangan perbandingan nilai bahan bakar akan mengikuti dan mengembalikan *plant* dalam keadaan sesuai dengan *set point*.

Pada sistem ini terdapat *noise* sistem maka dari itu digunakan kontroler LQG. Untuk mendapatkan nilai *state feedback* diperlukan *observer* yang berguna untuk estimasi *state* yang tidak terukur atau *state* terukur namun hasil pengukurannya terkontaminasi oleh *noise.* Dengan adanya *noise* sistem pula maka *observer* perlu dirancang dengan Kalman filter.

Langkah pertama yang dilakukan adalah mencari nilai Q dan R. Di mana nilai Q dan R ini adalah standar deviasi *noise* pada *plant* untuk nilai Q dan standar deviasi *noise* pada pengukuran untuk nilai R. Nilai Q dan R ini digunakan untuk mendapatkan *gain* kalman. Maka dengan perintah (*Digital Algrebaic Riccati Equation*) "DARE" pada Matlab dapat diperoleh *gain* kalman.

Langkah berikutnya adalah mendapatkan *gain feedback* dengan memilih matriks q dan r. Di mana matriks q dan r harus bersifat semi definit positif. Cara untuk mendapatkan matriks q dan r dengan metode *trial and error* yaitu dengan mencoba memasukkan nilai matriksnya dan diamati hasil grafik yang dihasilkan. Hasil grafik yang paling bagus yang diambil dan dengan melihat nilai indeks performansinya dengan rumus seperti pada persamaan 3.15 berikut ini.

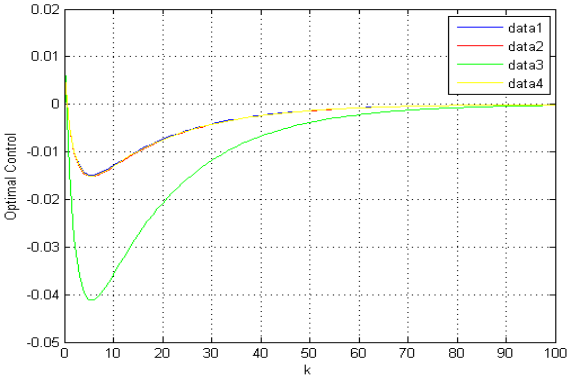
$$J^*(k0)=\frac{1}{2}x^{*'}(k0)\bar{P}x^*(k0) \qquad \textbf{(3.15)}$$

Berikut merupakan Tabel 3.1 yaitu tabel pemilihan matriks q dan r serta hasil respon dari setiap nilai q dan r pada Gambar 3.6.

Tabel 3.1 Matriks Pembobot q dan r dengan Indeks Performansi

No	Q	R	Indeks Performansi **(J)**
1	$\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}$	[1]	2.2592
2	$\begin{bmatrix}10 & 0 & 0\\0 & 10 & 0\\0 & 0 & 10\end{bmatrix}$	[1]	22.5772
3	$\begin{bmatrix}5 & 0 & 0\\0 & 10 & 0\\0 & 0 & 15\end{bmatrix}$	[0.5]	24.9892
4	$\begin{bmatrix}15 & 0 & 0\\0 & 10 & 0\\0 & 0 & 5\end{bmatrix}$	[0.5]	2.2592

Dari Tabel 3.1 hasil pada semua sinyal kontrolnya ditampilkan untuk masing-masing q dan r pada Gambar 3.6.



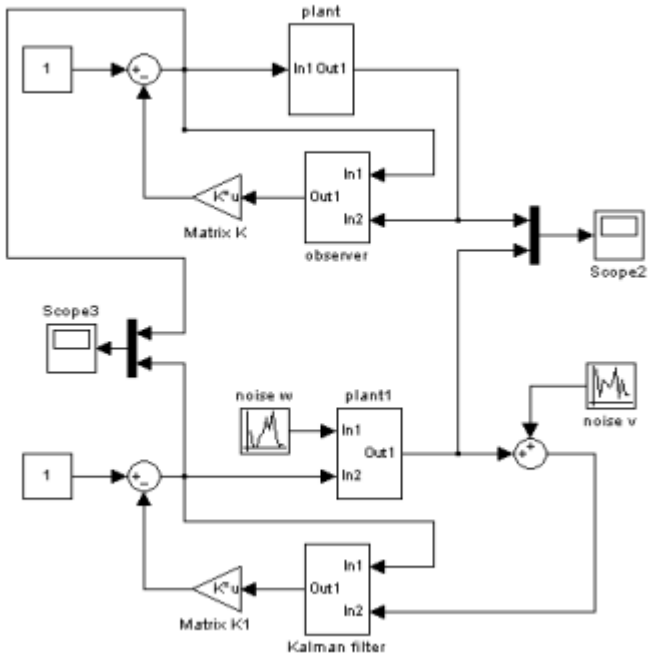
Gambar 3.5 Hasil *Plot* Sinyal Kontrol LQR dengan *Trial-Error* q dan r

Pada Gambar 3.6, terlihat hamper tidak ada perbedaan pada keempat hasil sinyal kontrol menggunakan LQR. Namun hasil pada data 1 terlihat lebih bagus karena selisih sedikit lebih cepat menuju

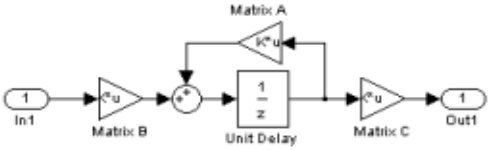
ke nilai nol. Maka dari itu nilai matriks q dan r nya dipilih dengan nilai adalah sebagai berikut.

$$q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}; r = [1]$$

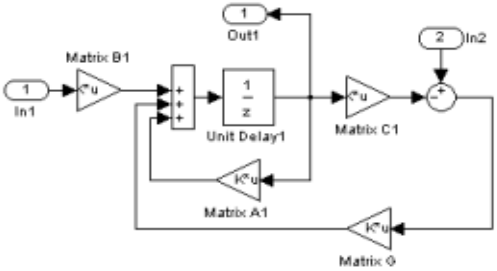
Setelah didapatkan matriks q dan r serta *gain feedback* dan *gain* Kalman, kemudian perancangan pada *Simulink* diagram blok dari LQG seperti pada Gambar 3.7 dan Gambar 3.8 berikut.



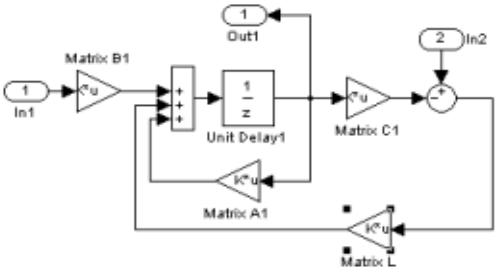
Gambar 3.6 Simulasi Kontroler LQG pada *Simulink*



(a)



(b)



(c)

Gambar 3.7 *Subsistem*: (a) *Plant*, (b) *Observer* dan (c) *Filter* Kalman

Dengan memasukkan semua nilai matriks pada *gain* yang terdapat pada Gambar 3.8 maka dapat diperoleh hasil simulasi kontroler LQG pada *Simulink* tersebut.

BAB 4
HASIL SIMULASI DAN ANALISA DATA

Pada bab ini, dibahas mengenai hasil simulasi sistem kontrol optimal yang telah dilakukan dan analisa data hasil simulasinya

4.1 Hasil Simulasi Identifikasi dan Pemodelan Sistem

Pada tahap ini hasil identifikasi pemodelan sistem disimulasikan untuk validasi *error* yang didapatkan. *Error* merupakan selisih antara hasil keluaran PLTG berupa daya MW dengan hasil simulasinya. Setelah didapatkan persamaan model diskritnya seperti pada persamaan 4.1 – 4.3 berikut.

$$A(z)y(t)=B(z)u(t)+e(t) \quad \textbf{(4.1)}$$

Di mana,

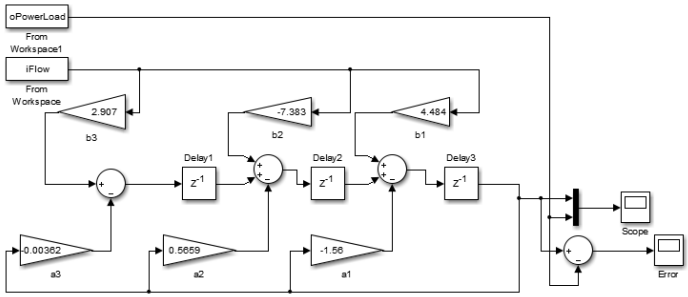
$$A(z)=1-1.56\ z^{-1}+0.5659\ z^{-2}-0.00362\ z^{-3} \quad \textbf{(4.2)}$$

$$B(z)=4.484\ z^{-1}-7.383\ z^{-2}+2.907\ z^{-3} \quad \textbf{(4.3)}$$

Dan untuk persamaan beda dari model ARX tersebut adalah seperti pada persamaan 4.4 berikut.

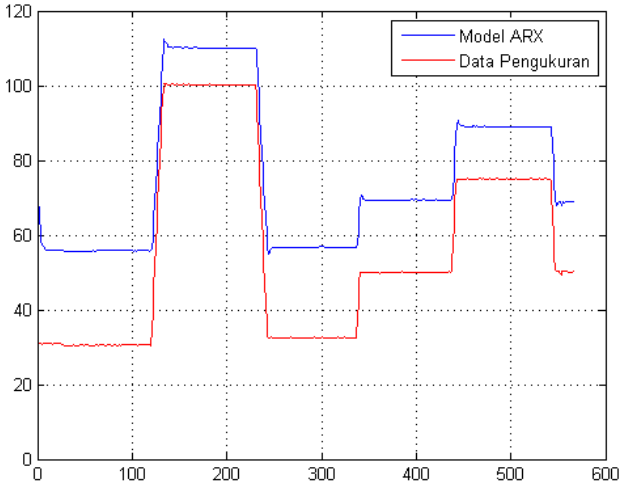
$$y(k)=1.56\ y(k-1)-0.5659\ y(k-2)+0.00362\ y(k-3) +4.484\ x(k-1)-7.383\ x(k-2)+2.907\ x(k-3) \quad \textbf{(4.4)}$$

Maka, dirancang simulasinya pada *Simulink* seperti pada Gambar 4.1 berikut.



Gambar 4.1 Rancangan *Simulink* Model ARX

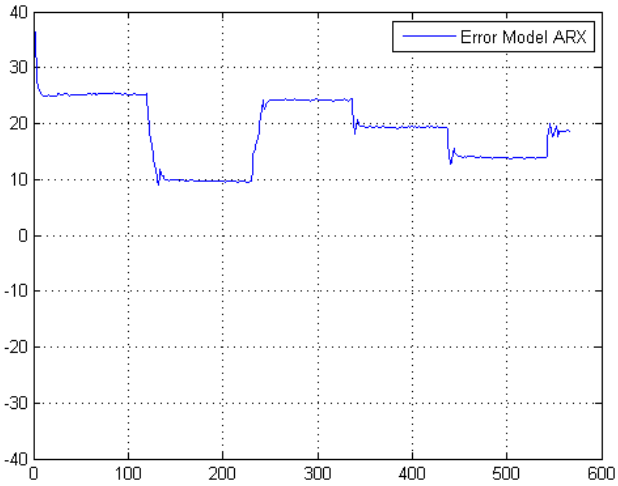
Pada Gambar 4.1, nilai dari enam parameter model diubah menjadi *gain* sebagai pengali sesuai dengan faktor *delay* dari masing-masing parameternya. Rancangan ini dirancang dalam bentuk kanonik. Untuk data masukan dan keluaran digunakan dari *workspace* dengan menambahkan waktu sampelnya pada kolom pertama dari masing-masing data. Setelah itu hasil simulasi dapat dilihat pada Gambar 4.2 berikut.



Gambar 4.2 Hasil Simulasi Model ARX

Terlihat pada Gambar 4.2 terdapat selisih atau *error* dari keluaran model ARX dan data keluaran yang terukur. Berikut pada Gambar 4.3 menampilkan grafik *error*nya.

Pada Gambar 4.3, terlihat hasil *error* yang terjadi bernilai negatif. Hal ini dikarenakan hasil keluaran model ARX lebih besar dari hasil keluaran pengukuran.



Gambar 4.3 Hasil *Error* Model ARX

Setelah itu, model ARX diubah menjadi model *state-space* diskrit menjadi seperti pada persamaan 4.5 dan 4.6 berikut.

$$\dot{x}=\begin{bmatrix}1.56 & -0.5659 & 0.05792\\ 1 & 0 & 0\\ 0 & 0.0625 & 0\end{bmatrix}x+\begin{bmatrix}8\\0\\0\end{bmatrix}u \qquad \textbf{(4.5)}$$

$$y=[0.5605 \quad -0.9229 \quad 5.813]\,x \qquad \textbf{(4.6)}$$

Untuk model ARX hasil identifikasi dengan menggunakan data pada saat transien di set point tertentu atau saat transien di beban 100MW didapatkan pada persamaan 4.7 – 4.10 berikut.

$$A(z)y(t)=B(z)u(t)+e(t) \qquad \textbf{(4.7)}$$

Di mana,

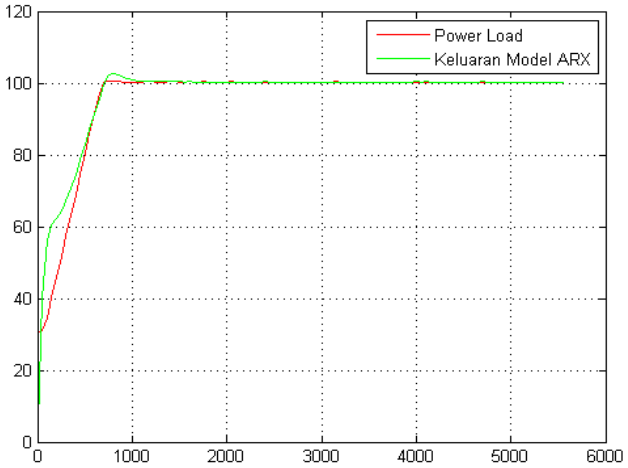
$$A(z)=1-1.551\,z^{-1}-0.8274\,z^{-2}-0.182\,z^{-3} \qquad \textbf{(4.8)}$$

$$B(z)=2.538\,z^{-1}-2.934\,z^{-2}+0.6995\,z^{-3} \qquad \textbf{(4.9)}$$

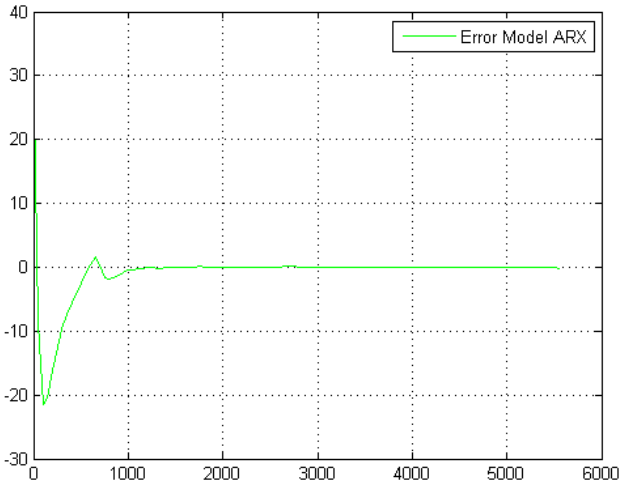
Dan untuk persamaan beda dari model ARX tersebut adalah seperti pada persamaan 4.10 berikut.

$$y(k)=1.551\,y(k-1)+0.8274\,y(k-2)+0.182\,y(k-3) +2.538\,x(k-1)-2.934\,x(k-2)+0.6995\,x(k-3) \quad \textbf{(4.10)}$$

Hasil simulasi dari keluaran model ARX dan keluaran PLTG berupa *Power Load* dapat dilihat pada Gambar 4.4 dan Gambar 4.5 untuk *error* dari keduanya.



Gambar 4.4 Hasil Simulasi Model ARX dengan Data Beban Penuh



Gambar 4.5 Hasil *Error* Simulasi Model ARX dengan Data Beban Penuh

Setelah itu, model ARX diubah menjadi model *state-space* diskrit menjadi seperti pada persamaan 4.11 dan 4.12 berikut.

$$\dot{x}=\begin{bmatrix} 0 & 0 & 0.364 \\ 0.5 & 0 & -0.8274 \\ 0 & 1 & 1.551 \end{bmatrix} x+\begin{bmatrix} 0.6995 \\ -1.467 \\ 1.269 \end{bmatrix} u+\begin{bmatrix} 0.182 \\ -0.4173 \\ 0.7755 \end{bmatrix} w \qquad \textbf{(4.11)}$$

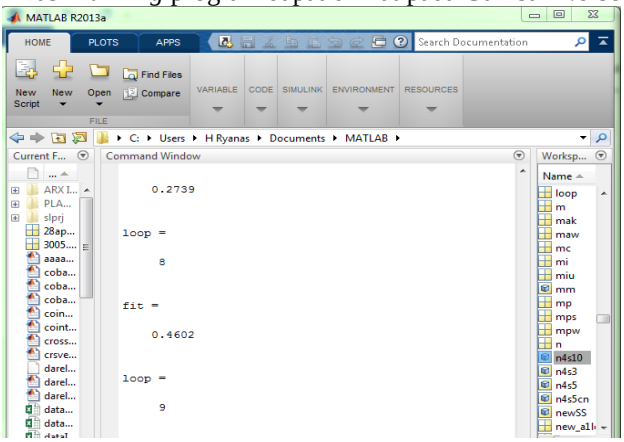
$$y=[0 \;\; 0 \;\; 2]\, x+[0]\, u \qquad \textbf{(4.12)}$$

4.2 Hasil Penerapan Algoritma PSO pada Identifikasi Model ARX

Pada tahap ini algoritma PSO ditambahkan dalam proses identifikasi. Jadi nilai parameter yang telah didapatkan model ARX akan diacak menggunakan 10 partikel pada tiap parameternya. Setiap parameter ini disebut dengan dimensi jadi pada model ini terdapat 6 dimensi dengan total 60 partikel.

Kemudian program akan dijalankan setelah melakukan inisialisasi pada program. Program berjalan hingga nilai *fitness* didapatkan melebihi nilai *fitness* dari model ARX sebelumnya.

Hasil *running* program dapat dilihat pada Gambar 4.6 berikut.



Gambar 4.6 Hasil Running Program PSO pada Matlab

Setelah itu, hasil dari nilai *fitness* yang didapatkan sebesar 0.6901 atau 69,01%. Hasil nilai *fitness* ini lebih besar daripada nilai *fitness* pada model ARX.

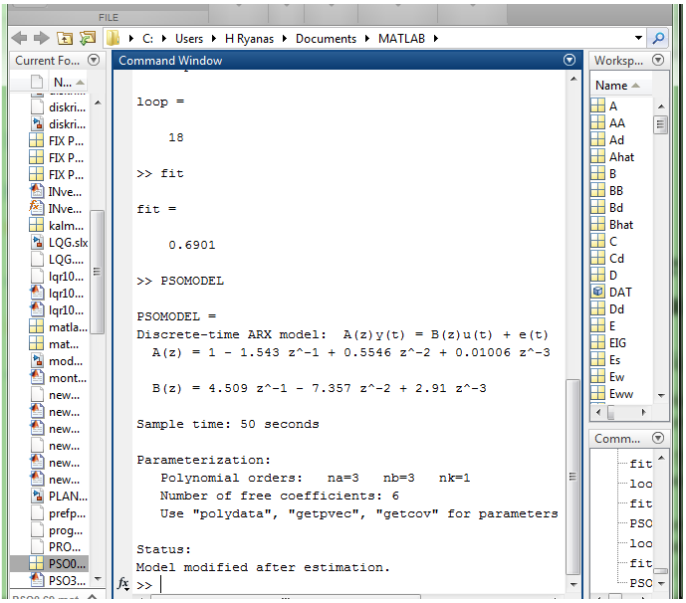
Hasil nilai *fitness* pada model PSO terdapat pada Gambar 4.7 berikut. Terlihat pada Gambar 4.7 persamaan modelnya adalah seperti pada persamaan 4.13 – 4.15 berikut.

$$A(z)y(t)=B(z)u(t)+e(t) \quad \textbf{(4.13)}$$

Di mana,

$$A(z)=1-1.543z^{-1}+0.5546z^{-2}-0.01006z^{-3} \quad \textbf{(4.14)}$$

$$B(z)=4.509z^{-1}-7.357z^{-2}-2.91\ z^{-3} \quad \textbf{(4.15)}$$

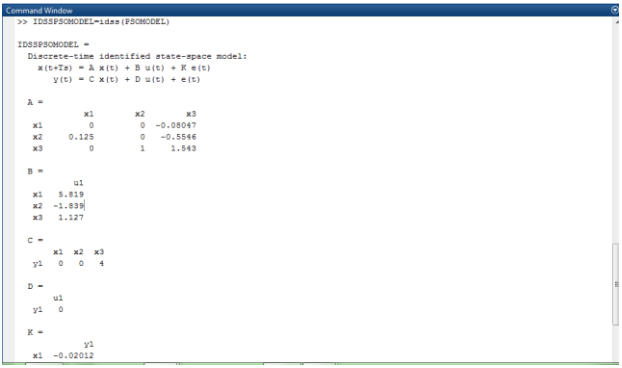


Gambar 4.7 Hasil Nilai *Fitness* menggunakan Algoritma PSO

Dan untuk persamaan beda dari model ARX tersebut adalah seperti pada persamaan 4.16 berikut.

$$y(k)=1.543\,y(k-1)-0.5546\,y(k-2)+0.01006\,y(k-3) +4.509\,x(k-1)-7.357\,x(k-2)+2.91\,x(k-3) \quad \textbf{(4.16)}$$

Kemudian model diubah menjadi persamaan *state-space* agar dapat dikontrol menggunakan kontrol optimal untuk selanjutnya. Untuk proses pengubahan model *state-space* dapat dilihat pada Gambar 4.8.

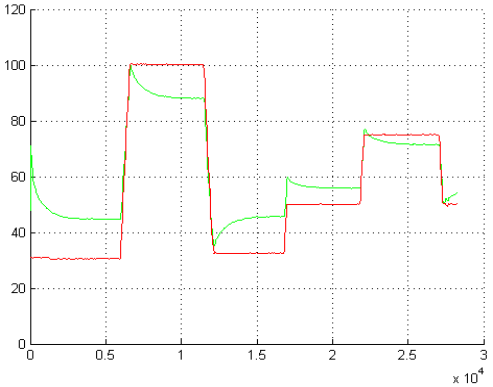


Gambar 4.8 Hasil Model *State-Space* dari Algoritma PSO

Setelah itu, model tambahan algoritma PSO diubah menjadi model *state-space* diskrit menjadi seperti pada persamaan 4.17 dan 4.18 berikut.

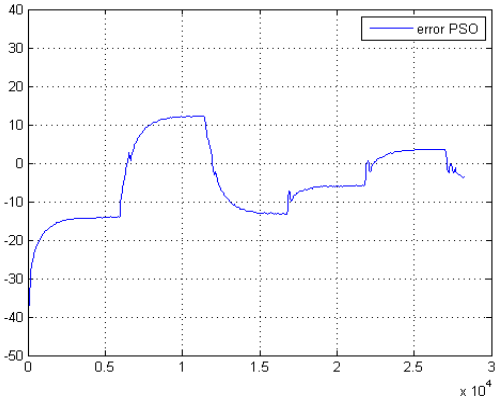
$$\dot{x}=\begin{bmatrix} 0 & 0 & -0.08047 \\ 0.125 & 0 & -0.5546 \\ 0 & 1 & 1.543 \end{bmatrix} x+\begin{bmatrix} 5.819 \\ -1.839 \\ 1.127 \end{bmatrix} u+\begin{bmatrix} -0.02012 \\ -0.1386 \\ 0.3857 \end{bmatrix} w \quad \textbf{(4.17)}$$

$$y=[0 \;\; 0 \;\; 4]\, x+[0]\, u \quad \textbf{(4.18)}$$



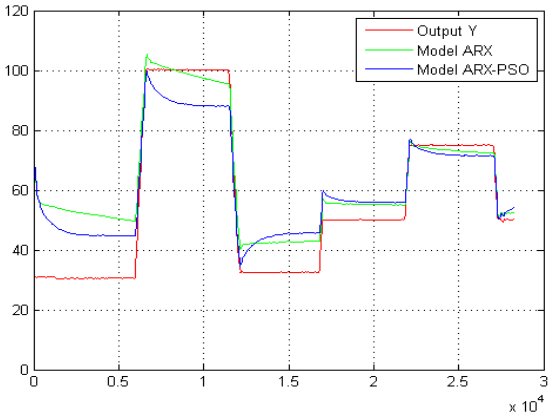
Gambar 4.9 Hasil Simulasi Model dengan Algortima PSO

Selanjutnya pada Gambar 4.10 menampilkan *error* dari model dengan tambahan algoritma PSO dengan hasil keluaran yang terukur sebenarnya.

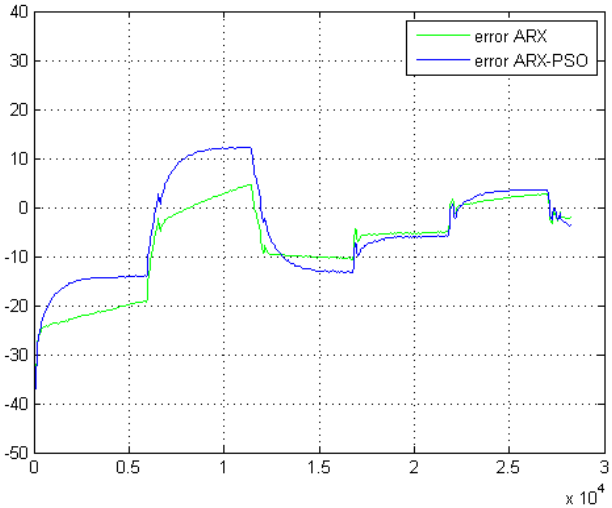


Gambar 4.10 Hasil *Error* Model dengan Algoritma PSO

Selanjutnya untuk melihat perbandingan model PSO dengan model ARX sebelumnya. Pada Gambar 4.11 ditampilkan grafik dari kedua model dan hasil keluaran sebenarnya serta pada Gambar 4.12 menampilkan perbandingan *error* pada kedua model tersebut.



Gambar 4.11 Hasil Perbandingan dari Model ARX dan Model PSO

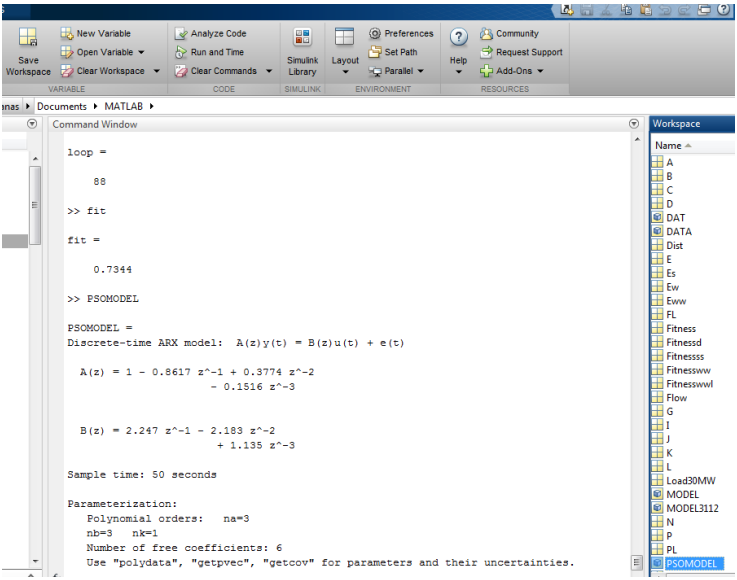


Gambar 4.12 Hasil Perbandingan Nilai *Error* Model ARX dan Model PSO

Jadi, model PSO lebih baik dikarenakan nilai *fitness*-nya sebesar 0,6901 atau 69,01% sedangkan untuk model ARX nilai *fitness*-nya sebesar 0,6712 atau 67,12%.

Kemudian dengan hanya menggunakan hasil identifikasi pada data saat beban penuh program akan dijalankan setelah melakukan inisialisasi pada program. Setelah itu, hasil dari nilai *fitness* yang didapatkan sebesar 0.7344 atau 73,44%. Hasil nilai *fitness* ini lebih besar daripada nilai *fitness* pada model ARX hasil identifikasi saat beban penuh.

Untuk hasil nilai *fitness* pada model PSO terdapat pada Gambar 4.20 berikut.



Gambar 4.13 Hasil Nilai *Fitness* Model pada Data Beban Penuh menggunakan Algoritma PSO

Terlihat pada Gambar 4.13 persamaan modelnya adalah seperti pada persamaan 4.19 – 4.21 berikut.

$$\mathbf{A(z)y(t) = B(z)u(t) + e(t)} \qquad \textbf{(4.19)}$$

Di mana,

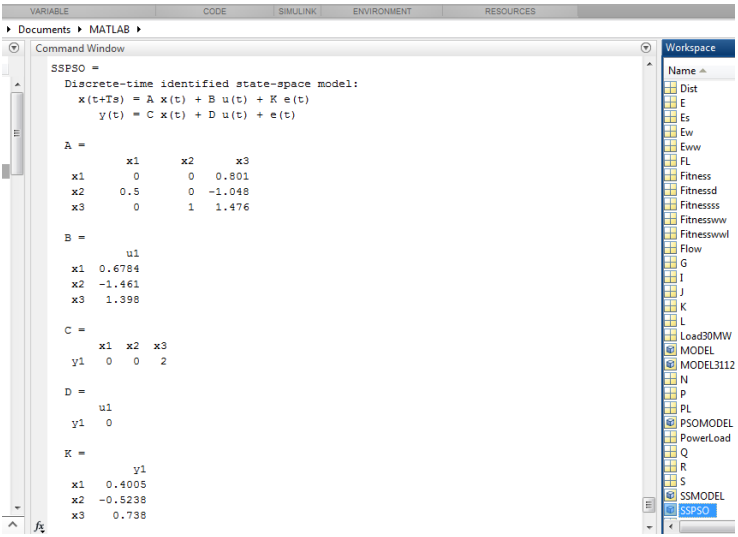
$$A(z)=1-0.8617\ z^{-1}+0.3774\ z^{-2}-0.1516\ z^{-3} \qquad \textbf{(4.20)}$$

$$B(z)=2.247\ z^{-1}-2.183\ z^{-2}+1.135\ z^{-3} \qquad \textbf{(4.21)}$$

Dan untuk persamaan beda dari model ARX tersebut adalah seperti pada persamaan 4.22 berikut.

$$y(k)=0.8617\ y(k-1)-0.3774\ y(k-2)+0.1516\ y(k-3) + 2.247\ x(k-1)-2.183\ x(k-2)+1.135\ x(k-3) \qquad \textbf{(4.22)}$$

Kemudian model diubah menjadi persamaan *state-space* agar dapat dikontrol menggunakan kontrol optimal untuk selanjutnya. Untuk proses pengubahan model *state-space* dapat dilihat pada Gambar 4.14.



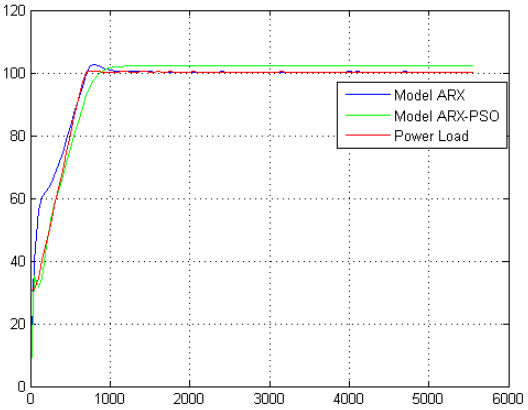
Gambar 4.14 Hasil Model *State-Space* dari Algoritma PSO

Setelah itu, model tambahan algoritma PSO diubah menjadi model *state-space* diskrit menjadi seperti pada persamaan 4.21 dan 4.22 berikut.

$$\dot{x}=\begin{bmatrix}0 & 0 & -0.801\\ 0.5 & 0 & -1.048\\ 0 & 1 & 1.476\end{bmatrix}x+\begin{bmatrix}0.6784\\ -1.461\\ 1.398\end{bmatrix}u+\begin{bmatrix}0.4005\\ -0.5238\\ 0.738\end{bmatrix}w \qquad \textbf{(4.21)}$$

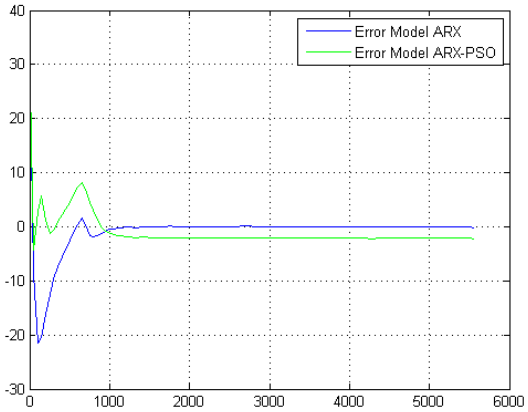
$$y=[0\ \ 0\ \ 2]\,x+[0]\,u \qquad \textbf{(4.22)}$$

Pada Gambar 4.15 terdapat perbandingan hasil respon keluaran dari Model ARX-PSO dengan Model ARX sebelumnya dan juga dengan keluaran pembangkit.



Gambar 4.15 Hasil Perbandingan Respon dengan Algoritma PSO

Selanjutnya pada Gambar 4.16 menampilkan *error* dari model ARX-PSO dengan hasil keluaran yang terukur sebenarnya.



Gambar 4.16 Hasil Perbandingan *Error* Model ARX dan ARX- PSO

4.3 Analisa Data Hasil Identifikasi Sistem

Dari hasil identifikasi sistem dihitung nilai kecocokan atau nilai *fitness* terhadap data keluaran yang sebenarnya untuk menghitungnya dengan mendapatkan nilai RMSE (*Root Mean Square Error*) terlebih dahulu maka dengan menggunakan rumus pada persamaan 4.23 dan 4.24 berikut ini.

$$\text{RMSE}=\sqrt{\frac{\sum_{i=1}^{n} e_i^2}{n}} \qquad \textbf{(4.23)}$$

Di mana,

n = banyaknya data

$\mathbf{e_i}$ = selisih data (*error*) ke i

Maka,

$$\text{fitness}=\frac{1}{1+\text{RMSE}} \qquad \textbf{(4.24)}$$

Setelah melakukan perhitungan nilai *fitness* seperti pada persamaan 4.23 dan 4.24 maka didapatkan nilai *fitness* pada model ARX dan *state-space* adalah sebesar 0,6712 atau 67,12 % dan untuk model dengan sampel 112 data didapatkan nilai *fitness* pada model ARX dan *state-space* saat beban penuh adalah sebesar 0,6902 atau 69,02 %.

Jadi, nilai *fitness* didapatkan lebih bagus dengan menggunakan 112 sampel data yaitu sebesar 69,02% dibandingkan dengan 566 sampe data yaitu sebesar 67,12%. Selisihnya dapat terlihat seperti pada Gambar 4.11 dan 4.12.

Setelah itu dengan tambahan algoritma PSO pada identifikasi model. Maka, didapatkan nilai *fitness* sebesar 69,01% untuk Model dengan 566 sampel data dan 73,44% untuk Model dengan 112 sampel data. Jadi, model PSO lebih baik dikarenakan nilai *fitness*-nya lebih besar.

Maka, dapat dilihat pada Tabel 4.1 merupakan perbandingan nilai fitness dari kedua data yang digunakan.

Tabel 4.1 Hasil Perbandingan Nilai *Fitness* pada Model

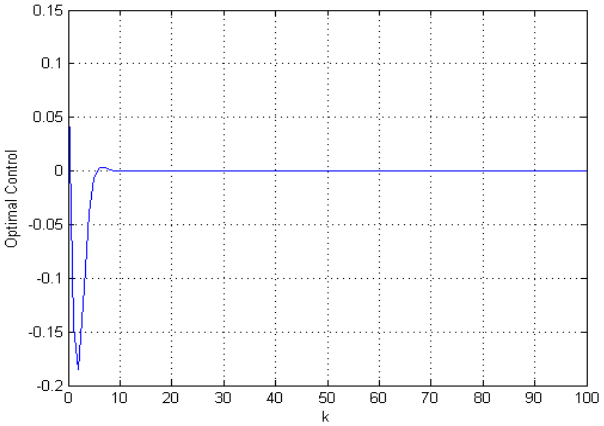
No	Dengan Data Keseluruhan		Dengan Data saat Beban Penuh	
	Model ARX	Model ARX-PSO	Model ARX	Model ARX-PSO
1.	67,12%	69,01%	69,02%	73,44%

Seperti pada Tabel 4.1 terlihat hasil identifikasi dengan data saat beban penuh dengan tambahan algortima PSO dengan nilai fitness sebesar 0.7344 atau 73,44%.

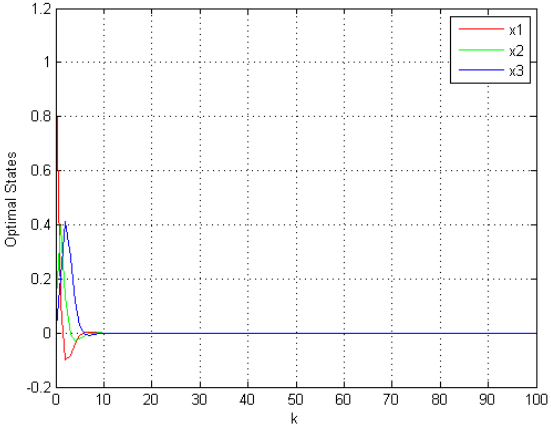
4.4 Hasil Simulasi Kontrol Optimal menggunakan LQG

Simulasi *plant* ini menggunakan model *state-space* PSO bukan pada model regresi dikrit karena nantinya matriks *state-space* digabungkan dengan kontroler LQR dengan tambahan *filter* Kalman yang berupa matriks *gain feedback* dan *gain* Kalman. Maka dari itu gabungan dari keduanya disebut dengan LQG.

Untuk hasil rancangan LQR adalah seperti pada Gambar 4.17 untuk hasil sinyal kontrol dan Gambar 4.18 untuk plot *state* sistem berikut ini.



Gambar 4.17 Hasil *Plot* Sinyal Kontrol dengan Kontroler LQR



Gambar 4.18 Hasil *Plot State* Optimal dengan Kontroler LQR

Terlihat pada Gambar 4.17 dan 4.18 hasil nilainya mendekati nol maka kontroler LQR telah berhasil diterapkan pada *state-space* sistem.

Kemudian didapatkan nilai *gain* Kalman (L) dan *gain feedback* (K). Nilai *gain feedback* dan *gain* Kalman didapatkan dalam persamaan 4.25 dan 4.26 berikut.

$$K=[-0.1165 \quad 0.2119 \quad 0.3525] \tag{4.25}$$

$$L=\begin{bmatrix}0.1511\\-0.1580\\0.6210\end{bmatrix} \tag{4.26}$$

Dengan memasukkan semua nilai parameter seperti matriks A, B, C, D, G, K, L, N dan *noise variance* pada *simulink* LQG seperti pada Gambar 3.7 dan 3.8. Maka didapatkan nilai optimal sinyal kontrol dan keluaran. Hal ini nantinya dibandingkan dengan sinyal kontrol sebelum diberikan kontroler LQG ini atau pada model sebelumnya.

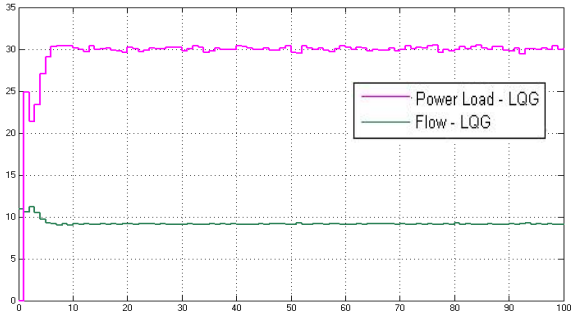
Nilai referensi atau *set point* dari LQG merupakan representasi dari nilai pengaturan daya keluaran (*power load*) pada PLTG. Pada

PLTG daya keluaran diatur sesuai dengan perintah P2B (Pusat Pengatur Beban) ke PLTG.

4.4.1 Hasil Simulasi LQG pada PLTG dengan Daya 30MW

Hasil simulasi saat PLTG dengan referensi daya keluaran 30MW. Dikarenakan unit PLTG maksimal dapat menghasilkan 100MW maka daya yang terbangkit sebesar 30%.

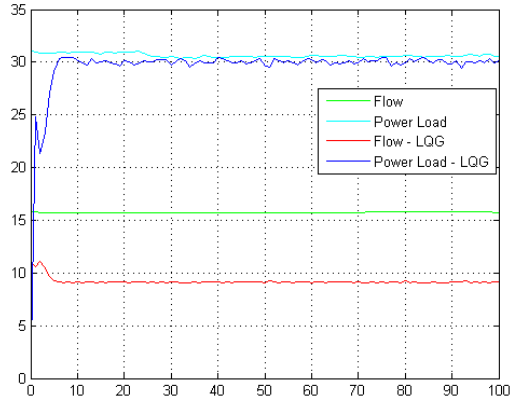
P ada Gambar 4.19 merupakan hasil simulasi dengan referensi daya sebesar 30MW dengan sinyal kontrol atau masukan berupa aliran gas (*Flow*).



Gambar 4.19 Hasil Simulasi dengan LQG pada PLTG 30MW

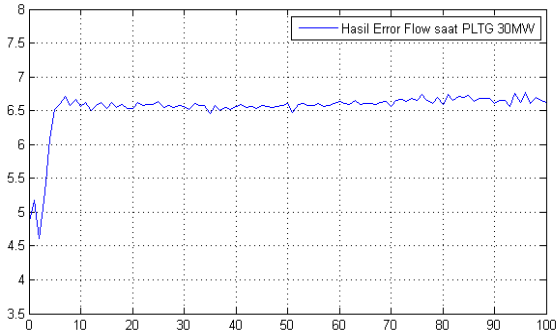
Pada Gambar 4.19, sinyal berwarna ungu merupakan daya yang terbangkit dan sinyal berwarna hijau merupakan masukan sistem atau besarnya aliran gas (*flow*).

Untuk mengetahui perbandingannya dengan sistem sebelumnya dapat dilihat pada Gambar 4.20 berikut.



Gambar 4.20 Hasil Perbandingan Simulasi LQG dengan Kondisi Nyata PLTG 30MW

Pada Gambar 4.21 ditampilkan *error* atau selisih dari nilai *Flow* dikurangkan dengan hasil sinyal kontrol LQG.

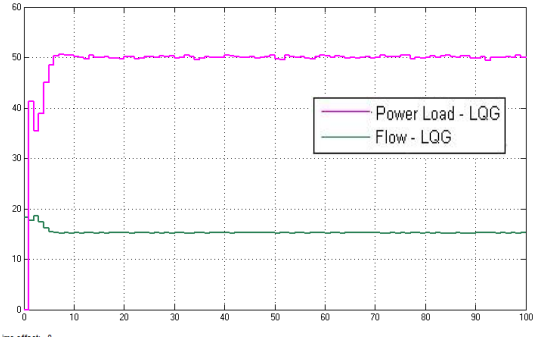


Gambar 4.21 Hasil *Error* pada nilai *Flow* dan sinyal kontrol LQG pada PLTG 30MW

4.4.2 Hasil Simulasi LQG pada PLTG dengan Daya 50MW

Hasil simulasi saat PLTG dengan referensi daya keluaran 50MW. Dikarenakan unit PLTG maksimal dapat menghasilkan 100MW maka daya yang terbangkit sebesar 50%.

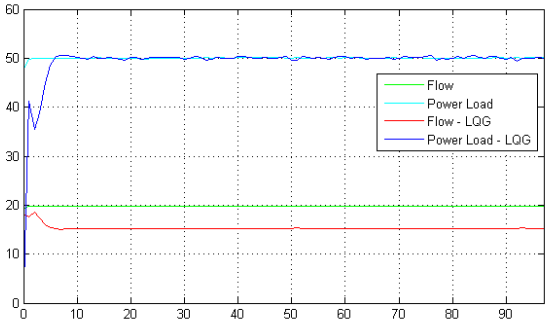
Pada Gambar 4.22 merupakan hasil simulasi dengan referensi daya sebesar 50MW dengan sinyal kontrol atau masukan berupa aliran gas (*Flow*).



Gambar 4.22 Hasil Simulasi dengan LQG pada PLTG 50MW

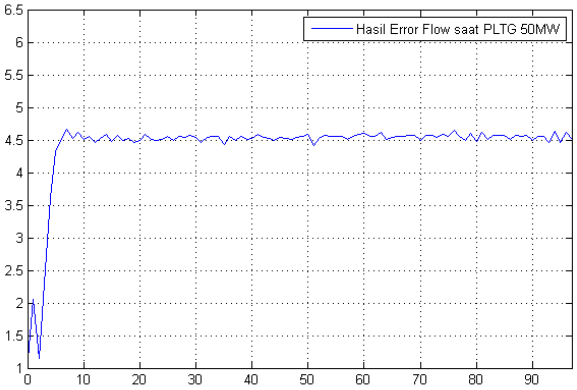
Pada Gambar 4.22, sinyal berwarna ungu merupakan daya yang terbangkit dan sinyal berwarna hijau merupakan masukan sistem atau besarnya aliran gas (*flow*).

Untuk mengetahui perbandingannya dengan sistem sebelumnya dapat dilihat pada Gambar 4.23



Gambar 4.23 Hasil Perbandingan Simulasi LQG dengan Kondisi Nyata PLTG 50MW

Pada Gambar 4.24 ditampilkan *error* atau selisih dari nilai *Flow* dikurangkan dengan hasil sinyal kontrol LQG.

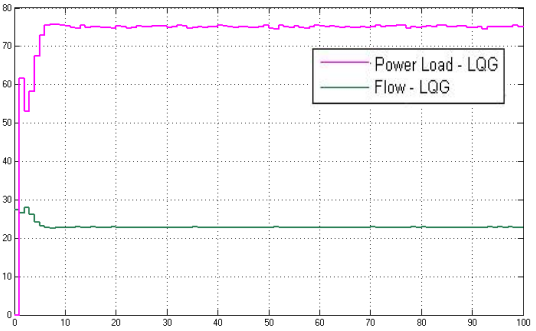


Gambar 4.24 Hasil *Error* pada nilai *Flow* dan sinyal kontrol LQG pada PLTG 50MW

4.4.3 Hasil Simulasi LQG pada PLTG dengan Daya 75MW

Hasil simulasi saat PLTG dengan referensi daya keluaran 75MW. Dikarenakan unit PLTG maksimal dapat menghasilkan 100MW maka daya yang terbangkit sebesar 75%.

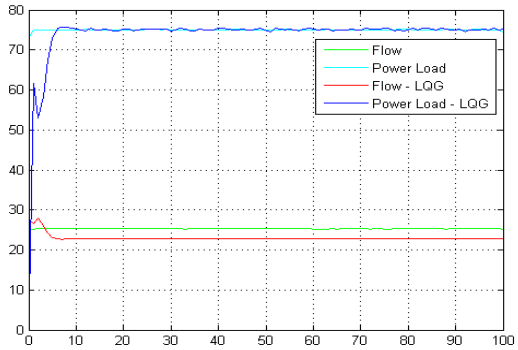
Pada Gambar 4.25 merupakan hasil simulasi dengan referensi daya sebesar 75MW dengan sinyal kontrol atau masukan berupa aliran gas (*Flow*).



Gambar 4.25 Hasil Simulasi dengan LQG pada PLTG 75MW

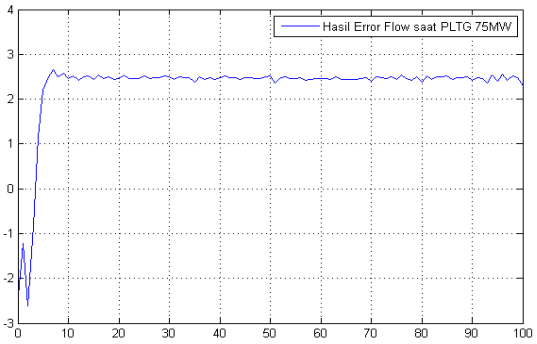
Pada Gambar 4.25, sinyal berwarna ungu merupakan daya yang terbangkit dan sinyal berwarna hijau merupakan masukan sistem atau besarnya aliran gas (*flow*).

Untuk mengetahui perbandingannya dengan sistem sebelumnya dapat dilihat pada Gambar 4.26



Gambar 4.26 Hasil Perbandingan Simulasi LQG dengan Kondisi Nyata PLTG 75MW

Pada Gambar 4.27 ditampilkan *error* atau selisih dari nilai *Flow* dikurangkan dengan hasil sinyal kontrol LQG.

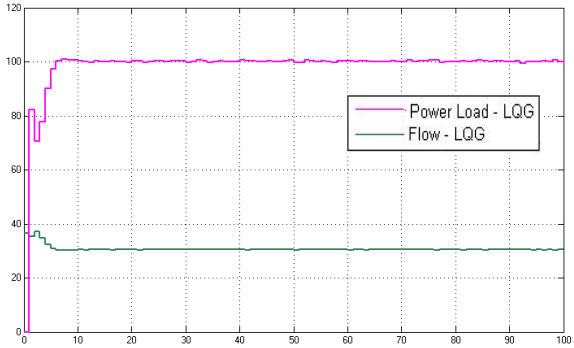


Gambar 4.27 Hasil *Error* pada nilai *Flow* dan sinyal kontrol LQG pada PLTG 75MW

4.4.4 Hasil Simulasi LQG pada PLTG dengan Daya 100MW

Hasil simulasi saat PLTG dengan referensi daya keluaran 100MW. Dikarenakan unit PLTG maksimal dapat menghasilkan 100MW maka daya yang terbangkit sebesar 100% atau dapat dikatakan dengan beban penuh.

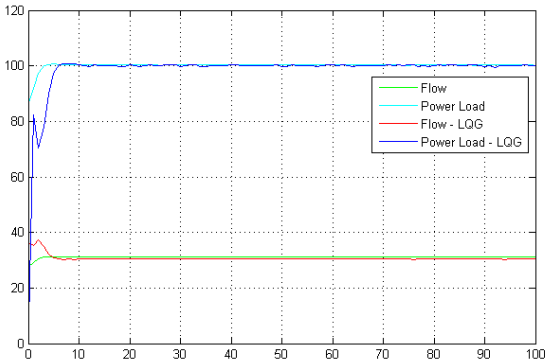
Pada Gambar 4.28 merupakan hasil simulasi dengan referensi daya sebesar 100MW dengan sinyal kontrol atau masukan berupa aliran gas (*Flow*).



Gambar 4.28 Hasil Simulasi dengan LQG pada PLTG 100MW

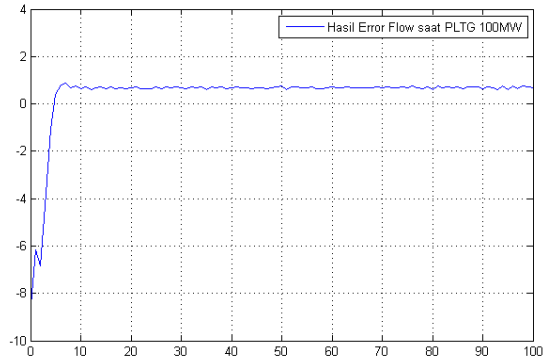
Pada Gambar 4.28, sinyal berwarna ungu merupakan daya yang terbangkit dan sinyal berwarna hijau merupakan masukan sistem atau besarnya aliran gas (*flow*).

Untuk mengetahui perbandingannya dengan sistem sebelumnya dapat dilihat pada Gambar 4.29



Gambar 4.29 Hasil Perbandingan Simulasi LQG dengan Kondisi Nyata PLTG 100MW

Pada Gambar 4.30 ditampilkan *error* atau selisih dari nilai *Flow* dikurangkan dengan hasil sinyal kontrol LQG.



Gambar 4.30 Hasil *Error* pada nilai *Flow* dan sinyal kontrol LQG pada PLTG 100MW

4.5 Analisa Data Hasil Simulasi Kontrol Optimal

Setelah diberikan kontrol optimal LQG pada sistem didapatkan hasil yang lebih efisien dengan nilai *Flow* yang lebih rendah dan hasil daya yang terbangkit sesuai dengan *set point* P2B.

Untuk hasil dengan *set point* 30MW terlihat pada Gambar 4.20 dan 4.21 bahwa nilai simulasi dari LQG lebih efisien dikarenakan energi yang dibutuhkan dalam hal ini *flow* sebagai masukan sistem lebih sedikit dengan hasil *Power Load* yang sesuai dengan referensi. Jadi, nilai *flow* dapat dihemat hingga sekitar 41,35%.

Untuk hasil dengan *set point* 50MW terlihat pada Gambar 4.23 dan 4.24 bahwa nilai simulasi dari LQG lebih efisien dikarenakan energi yang dibutuhkan dalam hal ini *flow* sebagai masukan sistem lebih sedikit dengan hasil *Power Load* sesuai dengan referensi. Jadi, nilai *flow* dapat dihemat sekitar 22,3%.

Untuk hasil dengan *set point* 75MW terlihat pada Gambar 4.26 dan 4.27 bahwa nilai simulasi dari LQG lebih efisien dikarenakan energi yang dibutuhkan dalam hal ini *flow* sebagai masukan sistem lebih sedikit dengan hasil *Power Load* yang sesuai dengan referensi. Jadi, nilai *flow* dapat dihemat hingga sekitar 9,05%.

Sedangkan untuk hasil dengan *set point* 100MW terlihat pada Gambar 4.29 dan 4.30 bahwa nilai simulasi dari LQG lebih efisien dikarenakan energi yang dibutuhkan dalam hal ini *flow* sebagai masukan sistem lebih sedikit dengan hasil *Power Load* yang sesuai dengan referensi. Jadi, nilai *flow* dapat dihemat hingga 1,6%.

Dari hasil yang didapatkan dengan beberapa *set point* berbeda diketahui apabila *set point* semakin rendah maka didapatkan efisiensi daya yang terbangkit semakin besar. Seperti pada Tabel 4.2 berikut ini.

Tabel 4.2 Hasil Perbandingan Nilai *Set Point* dan Efisiensi PLTG

No.	*Set Point* pada PLTG	Nilai *Flow* Aktual pada PLTG (kg/m^3)	Nilai *Flow* pada Simulasi (kg/m^3)	Peningkatan Efisiensi
1	30 MW	16,2	9,5	41,35%
2	50 MW	20,1	15,6	22,3%
3	75 MW	26,5	24,1	9,05%
4	100 MW	31	30,5	1,6%

BAB 5
PENUTUP

Pada bab ini, dibahas mengenai kesimpulan dari pengerjaan Tugas Akhir ini dan juga saran untuk penelitian selanjutnya.

5.1 Kesimpulan

Dari penelitian yang telah dilakukan, dapat ditarik kesimpulan bahwa penerapan algoritma PSO (*Particle Swarm Optimization*) pada identifikasi sistem dengan ARX memberikan hasil yang lebih baik dengan nilai *fitness* sebesar 0,6901 atau 69,01% yang sebelumnya mempunyai nilai *fitness* sebesar 0,6712 atau 67,12% dan dengan menggunakan data beban penuh didapatkan nilai *fitness* sebesar 0.7344 atau 73,44% yang sebelumnya mempunyau nilai *fitness* sebesar 0.6902 atau 69,02%.

Dan juga penerapan kontroler LQG (*Linear Quadratic Gaussian*) pada kontrol optimal memberikan hasil yang lebih efisien terhadap sistem karena dengan nilai masukan yang lebih sedikit dapat menghasilkan keluaran daya yang bernilai sama atau lebih dengan referensi atau daya yang tercatat. Dengan *error* atau selisih masukan aliran gas akan lebih kecil apabila nilai referensi semakin besar.

5.2 Saran

Pada penelitian berikutnya diharapkan pengumpulan data yang lebih detail sehingga dapat melakukan percobaan dan pengamatan melalui pemodelan secara terpisah pada turbin gas dan generator.

Dan juga diharapkan melakukan kajian untuk pemodelan analitis atau matematis yang dapat merepresentasikan besaran-besaran yang ada pada sistem secara lengkap dan utuh.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] Hongjin, Zhang; Yizuo, Zhang; Weidong, Zhang; Tao, Tao, "***Swarm Intelligence Algorithm on Combustion Optimization of Coal-fired Boiler***," in *Chinese Control Conference*, Chengdu, China, 2016.

[2] Soedibyo, **Pembangkit Tenaga Listrik**, Surabaya: ITS Press, 2015.

[3] R. Effendi, **Handout Mata Kuliah Sistem Pengaturan Adaptif**, Surabaya: Institut Sepuluh Nopember Surabaya, 2004.

[4] K. Ogata, ***Modern Control Engineering Third Edition***, Minnesota: Prentice-Hall International, Inc, 1997.

[5] V. Adilla, **Pengendalian Rasio Bahan Bakar dan Udara pada Boiler menggunakan Metode Kontrol Optimal LQR**, Surabaya: Institut Sepuluh Nopember Surabaya, 2013.

[6] D. S. Naidu, ***Optimal Control Systems,*** Idaho: CRC Press, 2002.

[7] F. R. Lewis, ***Optimal Control***, New York: John Wiley & Son, Inc, 1995.

[8] Suyanto, **Algoritma Optimasi**, Bandung: Graha Ilmu, 2010.

Halaman ini sengaja dikosongkan

LAMPIRAN

A. Program Algoritma PSO pada Model ARX

```
%Inisialisasi
npar=3;
n=566;
tsample=50;
t=1:tsample:n*tsample;
x=evalin('base','Flow');
y=evalin('base','PowerLoad');
zero=[0;0;0];
xk=[zero;x];
yk=[zero;y];
DAT=iddata(y,x,tsample);

%Identifikasi Sistem
MODEL=arx(DAT,([npar npar 1]));
par=getpvec(MODEL);
z=lsim(MODEL,x,t);
ei=y-z;
eei=ei(1:n);
E=sqrt((eei'*eei))/(n);
Fitness=1/(1+E);

%State-Space Model
SSMODEL=ss(MODEL);
zss=lsim(SSMODEL,x,t);
eiss=y-zss;
eeiss=eiss(1:n);
Es=sqrt((eeiss'*eeiss))/(n);
Fitnessss=1/(1+Es);

%%Algoritma PSO
%Inisialiasi dan Pembangkitan Partikel
Ew=zeros(1,10);
fit=0;
loop=0;
while fit<0.71
loop=loop+1
```

```
iter=25;
part=10;
va=0.02;
vb=0.025;
vv=0.05;
para=[MODEL.a(2) MODEL.a(3) MODEL.a(4)];
parb=[MODEL.b(2) MODEL.b(3) MODEL.b(4)];
tpp=MODEL;

a1=para(1);a2=para(2);a3=para(3);
b1=parb(1);b2=parb(2);b3=parb(3);
a1loc=(a1)+(randn(part,1)*va);
a2loc=(a2)+(randn(part,1)*va);
a3loc=(a3)+(randn(part,1)*va);
b1loc=(b1)+(randn(part,1)*vb);
b2loc=(b2)+(randn(part,1)*vb);
b3loc=(b3)+(randn(part,1)*vb);

vel1=rand(part,1)*vv;vel2=rand(part,1)*vv;ve
l3=rand(part,1)*vv;
vel4=rand(part,1)*vv;vel5=rand(part,1)*vv;ve
l6=rand(part,1)*vv;

%Inisaliasi pBest

for m=1:part

tpp.a(1)=1;tpp.a(2)=a1loc(m);tpp.a(3)=a2loc(
m);tpp.a(4)=a3loc(m);

tpp.b(1)=0;tpp.b(2)=b1loc(m);tpp.b(3)=b2loc(
m);tpp.b(4)=b3loc(m);
    asdf(m,:)=tpp;
    zmm = lsim(asdf(m),x,t);
    ewi=y-zmm;
    eewi=ewi(1:n);
    Eww=sqrt((eewi'*eewi))/(n);
    Fitnessww(m,:)=1/(1+Eww);
end
```

```
    pbest=Fitnessww;
    pa1loc=a1loc;
    pa2loc=a2loc;
    pa3loc=a3loc;
    pb1loc=b1loc;
    pb2loc=b2loc;
    pb3loc=b3loc;

%Inisialisasi gBest
[value nq]=max(pbest);
gbest=value;
gbest_a1loc=pa1loc(nq);
gbest_a2loc=pa2loc(nq);
gbest_a3loc=pa3loc(nq);
gbest_b1loc=pb1loc(nq);
gbest_b2loc=pb2loc(nq);
gbest_b3loc=pb3loc(nq);

%Looping
i=2;
while i<= iter
    %update location
    new_a1loc=a1loc(:,i-1)+vel1(:,i-1);
new_a2loc=a2loc(:,i-1)+vel2(:,i-1);
    new_a3loc=a3loc(:,i-1)+vel3(:,i-1);
new_b1loc=b1loc(:,i-1)+vel4(:,i-1);
    new_b2loc=b2loc(:,i-1)+vel5(:,i-1);
new_b3loc=b3loc(:,i-1)+vel6(:,i-1);

    %add new location
    a1loc=[a1loc new_a1loc]; a2loc=[a2loc
new_a2loc];
    a3loc=[a3loc new_a3loc]; b1loc=[b1loc
new_b1loc];
    b2loc=[b2loc new_b2loc]; b3loc=[b3loc
new_b3loc];

    temp_pbest=[];
```

```
    temp_pa1loc=[]; temp_pa2loc=[];
temp_pa3loc=[];
    temp_pb1loc=[]; temp_pb2loc=[];
temp_pb3loc=[];

    %fitness evaluation
for l=1:part

tpp.a(1)=1;tpp.a(2)=a1loc(l);tpp.a(3)=a2loc(
l);tpp.a(4)=a3loc(l);

tpp.b(1)=0;tpp.b(2)=b1loc(l);tpp.b(3)=b2loc(
l);tpp.b(4)=b3loc(l);
    asdf(l,:)=tpp;
    zll = lsim(asdf(l),x,t);
    ewi=y-zll;
    eewi=ewi(1:n);
    Eww=sqrt((eewi'*eewi))/(n);
    Fitnesswwl(l,:)=1/(1+Eww);
end

    %update pbest
    for j=1:part
        if pbest(j,i-1)<Fitnesswwl(j)

temp_pbest=[temp_pbest;Fitnesswwl(j)];

temp_pa1loc=[temp_pa1loc;a1loc(j,i)];

temp_pa2loc=[temp_pa2loc;a2loc(j,i)];

temp_pa3loc=[temp_pa3loc;a3loc(j,i)];

temp_pb1loc=[temp_pb1loc;b1loc(j,i)];

temp_pb2loc=[temp_pb2loc;b2loc(j,i)];

temp_pb3loc=[temp_pb3loc;b3loc(j,i)];
        else
```

```
temp_pbest=[temp_pbest;pbest(j,i-1)];

temp_pa1loc=[temp_pa1loc;a1loc(j,i-1)];

temp_pa2loc=[temp_pa2loc;a2loc(j,i-1)];

temp_pa3loc=[temp_pa3loc;a3loc(j,i-1)];

temp_pb1loc=[temp_pb1loc;b1loc(j,i-1)];

temp_pb2loc=[temp_pb2loc;b2loc(j,i-1)];

temp_pb3loc=[temp_pb3loc;b3loc(j,i-1)];
        end
    end

    %update gbest
    [pp xp]=max(temp_pbest);
    if gbest(i-1)<pp
        gbest=[gbest;pp];

gbest_a1loc=[gbest_a1loc;temp_pa1loc(xp)];

gbest_a2loc=[gbest_a2loc;temp_pa2loc(xp)];

gbest_a3loc=[gbest_a3loc;temp_pa3loc(xp)];

gbest_b1loc=[gbest_b1loc;temp_pb1loc(xp)];

gbest_b2loc=[gbest_b2loc;temp_pb2loc(xp)];

gbest_b3loc=[gbest_b3loc;temp_pb3loc(xp)];
     else
        gbest=[gbest;gbest(i-1)];

gbest_a1loc=[gbest_a1loc;gbest_a1loc(i-1)];

gbest_a2loc=[gbest_a2loc;gbest_a2loc(i-1)];
```

```
gbest_a3loc=[gbest_a3loc;gbest_a3loc(i-1)];

gbest_b1loc=[gbest_b1loc;gbest_b1loc(i-1)];

gbest_b2loc=[gbest_b2loc;gbest_b2loc(i-1)];

gbest_b3loc=[gbest_b3loc;gbest_b3loc(i-1)];
     end

    pbest=[pbest temp_pbest];
    pa1loc=[pa1loc temp_pa1loc];
    pa2loc=[pa2loc temp_pa2loc];
    pa3loc=[pa3loc temp_pa3loc];
    pb1loc=[pb1loc temp_pb1loc];
    pb2loc=[pb2loc temp_pb2loc];
    pb3loc=[pb3loc temp_pb3loc];

    %Update Velocity
    r1=rand(); r2=rand();
    co1=2.05; co2=co1;
    phi=co1+co2;
    k=2/abs(2-phi-sqrt(phi^2-4*phi));

    %Add Velocity
    vel1=[vel1 k*(vel1(:,i-
1)+r1*co1*(pa1loc(:,i)-a1loc(:,i))-
r2*co2*(repmat(gbest_a1loc(i),part,1))-
a1loc(:,i))];
    vel2=[vel2 k*(vel2(:,i-
1)+r1*co1*(pa2loc(:,i)-a2loc(:,i))-
r2*co2*(repmat(gbest_a2loc(i),part,1))-
a2loc(:,i))];
    vel3=[vel3 k*(vel3(:,i-
1)+r1*co1*(pa3loc(:,i)-a3loc(:,i))-
r2*co2*(repmat(gbest_a3loc(i),part,1))-
a3loc(:,i))];
    vel4=[vel4 k*(vel4(:,i-
1)+r1*co1*(pb1loc(:,i)-b1loc(:,i))-
```

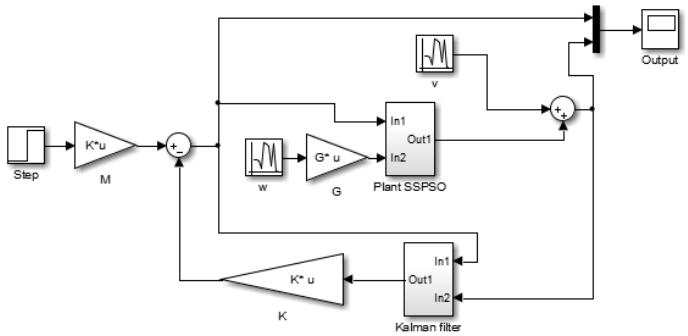
```
r2*co2*(repmat(gbest_b1loc(i),part,1))-
b1loc(:,i))];
    vel5=[vel5 k*(vel5(:,i-
1)+r1*co1*(pb2loc(:,i)-b2loc(:,i))-
r2*co2*(repmat(gbest_b2loc(i),part,1))-
b2loc(:,i))];
    vel6=[vel6 k*(vel6(:,i-
1)+r1*co1*(pb3loc(:,i)-b3loc(:,i))-
r2*co2*(repmat(gbest_b3loc(i),part,1))-
b3loc(:,i))];
    if i==iter
        fit=gbest(iter)
    else
    end
    i=i+1;
end
end
loop=0;
fita1=gbest_a1loc(25);
fita2=gbest_a2loc(25);
fita3=gbest_a3loc(25);
fitb1=gbest_b1loc(25);
fitb2=gbest_b2loc(25);
fitb3=gbest_b3loc(25);
tpp.a(2)=fita1; tpp.a(3)=fita2;
tpp.a(4)=fita3;
tpp.b(2)=fitb1; tpp.b(3)=fitb2;
tpp.b(4)=fitb3;
PSOMODEL=tpp;
zpso=lsim(PSOMODEL,x,t);
plot(t,zss);hold
on;plot(t,zpso,'g');plot(t,y,'r');plot(t,z,'
y');
```

B. Program Kontrol Optimal LQG

```
%Inisialisasi
SSPSO=idss(PSOMODEL);
A=SSPSO.a;
B=SSPSO.b;
C=SSPSO.c;
D=SSPSO.d;
G=SSPSO.k;
Dist=PSOMODEL.noisevariance;
Q=diag([Dist Dist Dist]);
R=0.001;
q=[1 0 0;0 1 0;0 0 1];
r=[1];
x1(1)=1;
x2(1)=0.1;
x3(1)=0;
xk=[x1(1);x2(1);x3(1)];
k0=0;
kf=100;
N=kf+1;
[n,n]=size(A);
I=eye(n);

%Mendapatkan Gain kalman L
[P,e,l,o]=dare(A,B,Q,R);
L=A*P*C'*inv(C*P*C'+R);

%Mndapatkan Gain K
[S,yy,kk,ii]=dare(A,B,q,r);
K=inv(r+B'*S*B)*B'*S*A;

for k=1:N-1
    xk=[x1(k);x2(k);x3(k)];
    xkplus1=(A-B*K)*xk;
    x1(k+1)=xkplus1(1);
    x2(k+1)=xkplus1(2);
    x3(k+1)=xkplus1(3);
    end
```

```
for k=1:N
    xk=[x1(k);x2(k);x3(k)];
    u(k)=-K*xk;
end
k=0:100;
%plot(k,x1,'r',k,x2,'g',k,x3,'b');grid on
xlabel('k')
ylabel('Optimal States')
%plot(k,u,'b')
xlabel('k')
%ylabel('Optimal Control');grid on
J=0.5*[x1(1) x2(1) x3(1)]*S*[x1(1) ;x2(1);
x3(1)]
K
L
```

C. Blok *Simulink* LQG



Halaman ini sengaja dikosongkan

RIWAYAT PENULIS

Hanif Ryanas Putra lahir di kota Madiun, Jawa Timur pada tanggal 19 April 1995. Penulis memulai pendidikannya dari TK An-Nuur di Madiun, kemudian melanjutkan studinya di MI PSM Kota Madiun, SMPN 4 Kota Madiun, dan SMK Negeri 1 Kota Madiun. Setelah menamatkan SMA, penulis melanjutkan studinya di D3 Teknik Elektro FTI-ITS dan lulus pada tahun 2015. Selanjutnya, penulis meneruskan studi sarjana di Teknik Elektro FTE-ITS, kemudian fokus pada bidang studi Teknik Sistem Pengaturan. Pada bulan Juli 2017, penulis mengikuti seminar dan Ujian Tugas Akhir sebagai salah satu persyaratan untuk memperoleh gelar Sarjana Teknik Elektro dari Institut Teknologi Sepuluh Nopember Surabaya.

Halaman ini sengaja dikosongkan