UNIVERSITAS INDONESIA

DESAIN FILTER KALMAN UNTUK MENGESTIMASI VARIABEL KEADAAN YANG TIDAK TERUKUR PADA SISTEM TATA UDARA PRESISI

SKRIPSI

ANTONI ALDILA

0806330693

FAKULTAS TEKNIK

PROGRAM STUDI TEKNIK ELEKTRO

DEPOK

JULI 2012

UNIVERSITAS INDONESIA

DESAIN FILTER KALMAN UNTUK MENGESTIMASI VARIABEL KEADAAN YANG TIDAK TERUKUR PADA SISTEM TATA UDARA PRESISI

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana

ANTONI ALDILA

0806330693

FAKULTAS TEKNIK

PROGRAM STUDI TEKNIK ELEKTRO

DEPOK

JULI 2012

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar

Nama	**: Antoni Aldila**
NPM	**: 0806330693**
Tanda Tangan	**:** 
Tanggal	**: 3 Juli 2012**

LEMBAR PENGESAHAN

Skripsi ini diajukan oleh	:
Nama	: Antoni Aldila
NPM	: 0806330693
Program Studi	: Teknik Elektro
Judul Skripsi	: Desain Filter Kalman untuk Mengestimasi Variabel Keadaan yang Tidak Terukur pada Sistem Tata Udara Presisi

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang dilakukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro, Fakultas Teknik, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing : Ir. Aries Subiantoro, M.SEE. (.....)

Penguji : Dr. Ir. Feri Yusivar, M.Eng. (.....)

Penguji : Prof. Dr.Eng. Drs. Benyamin Kusumoputro, M.Eng. (.....)

Ditetapkan di : Depok

Tanggal : 3 Juli 2012

KATA PENGANTAR

Segala puji dan syukur Penulis panjatkan kehadirat Allah SWT atas segala berkat dan rahmat-Nya sehingga Penulis dapat menyelesaikan penulisan seminar ini. Penulisan seminar ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Jurusan Teknik Elektro pada Fakultas Teknik Universitas Indonesia. Saya menyadari bahwa tanpa bantuan dan bimbingan dari berbagai pihak sangatlah sulit bagi Penulis untuk menyelesaikan penulisan seminar ini. Oleh karena itu Penulis mengucapkan terima kasih kepada:

1. Ir. Aries Subiantoro, M.SEE sebagai dosen pembimbing yang telah dengan sabar membimbing Penulis dalam penyusunan laporan seminar ini.
2. Rizky Prasetya S. T. sebagai senior Penulis yang telah mengajari Penulis mengenai seluk beluk Filter Kalman.
3. Victor S. T. sebagai senior Penulis yang telah mengajari banyak hal tentang identifikasi sistem dalam sistem tata udara presisi.
4. Nur Hidayat S. T. sebagai senoir Penulis yang telah membantu mengajari banyak hal mengenai simulasi dan pengambilan data sistem tata udara presisi.

Masih banyak kekurangan dalam penulisan seminar ini, kritik dan saran yang membangun sangat Penulis harapkan demi kemajuan pengetahuan mengenai seminar ini.

Akhir kata, Penulis sangat berharap laporan seminar ini berguna bagi pembaca dan pengembangan ilmu pengetahuan di Universitas Indonesia.

Depok, 3 Juli 2012

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama	: Antoni Aldila
NPM	: 0806330693
Program Studi	: Teknik Elektro
Departemen	: Teknik Elektro
Fakultas	: Teknik
Jenis Karya	: Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty-Free Right*)** atas karya ilmiah saya yang berjudul :

Desain Filter Kalman untuk Mengestimasi Variabel Keadaan yang Tidak Terukur pada Sistem Tata Udara Presisi

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok

Pada tanggal : 3 Juli 2012

Yang menyatakan

(Antoni Aldila)

ABSTRAK

Nama : Antoni Aldila
Program Studi : Teknik Elektro
Judul : Desain Filter Kalman untuk Mengestimasi Variabel Keadaan yang Tidak Terukur pada Sistem Tata Udara Presisi

Sistem tata udara presisi merupakan mesin refrigerasi yang digunakan di ruang pusat data untuk menjaga temperatur di dalam kabinet berkisar antara 20º - 22ºC, dan kelembaban antara 45-55%. Untuk mencapai keadaan tersebut, delapan variabel tak terukur belum dapat diestimasi sehingga dibutuhkan observer. Proses estimasi *state* dilakukan menggunakan model ruang keadaan. Persamaan untuk Filter Kalman dibagi menjadi persamaan *time update* dan *measurement update*. Penggunaan metode ini diharapkan diperoleh nilai matriks *prediction error covarians* yang konvergen pada nilai sekecil mungkin. Selain itu juga dibandingkan *state* hasil estimasi dengan *state* aktual model untuk mengetahui nilai kuadrat kesalahan estimasi yang terjadi.

Kata kunci: *Observer*, Filter Kalman, Sistem Tata Udara Presisi, Model Linier

ABSTRACT

Name : Antoni Aldila
Major : Electrical Engineering
Title : Design of Kalman Filter to Estimate Unmeasured State Variables of Prescision Air Conditioning

Precision air conditioning is a refrigeration machine that used in the data center to keep the temperature inside the cabinet ranged from 20 º - 22 º C, and humidity between 45-55%. To reach that state, the eight variables not measured can not be estimated so that the observer is required. State estimation process is done using a state space model. The equation for the Kalman Filter equations are divided into time update and measurement update. Use of this method is expected to obtain the prediction error matrix covarians which converges on the value as small as possible. It also compared to the estimated state with the actual state of the model to determine the value of the square of estimation error that occurred.

Keywords : Observer, Kalman Filter, Prescision Air Conditioning, Linear Model

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Sistem tata udara presisi (*Precision Air Conditioning* atau PAC) merupakan sistem refrigerasi yang bekerja berdasarkan konsep termodinamika. Mesin refrigerasi adalah alat yang melakukan proses perpindahan kalor dari media bersuhu tinggi ke media bersuhu rendah dengan memanfaatkan siklus refrigerasi *(vapor-compression cycle).* PAC banyak digunakan di berbagai kebutuhan industri maupun rumah tangga. PAC ini digunakan untuk mengendalikan suhu dan kelembaban udara relatif pada kabinet yang ada di ruang pusat data, sehingga suhu dan kelembabannya terjaga konstan di nilai tertentu. Hal ini bertujuan untuk menjaga peralatan IT bisa beroperasi secara kontinu dengan meminimalkan kemungkinan kerusakan. Tujuan lainnya adalah untuk menjaga usia pemakaian peralatan IT tersebut agar bertahan lama. Tujuan lain yang seperti telah disebutkan di atas adalah pengefisienan energi, sehingga pengeluaran perusahaan bisa ditekan menjadi lebih murah. Suhu ideal untuk peralatan IT sekitar 20-22oC dan kelembaban relatif ideal untuk peralatan IT sekitar 45-55%.

Untuk dapat mengendalikan PAC ini dibutuhkan algoritma cerdas yang dapat membuat alat ini bekerja pada nilai yang diinginkan. Untuk itu didesain suatu algoritma MPC yang dapat mengendalikan sistem tata udara presisi secara optimal sehingga masalah konsumsi energi yang cukup besar dalam sistem tata udara presisi dapat teratasi. Dalam MPC sendiri tidak semua *state* variabelnya dapat terukur, oleh karena itu mengestimasi *state* variabel yang takterukur ini dibutuhkan obserbver yang handal yang dapat bekerja optimal. Sehingga digunakan observer *kalman filter* untuk dapat mengestimasi *state* sistem tata udara presisi.

Digunakan observer *kalman filter* dalam penelitian ini karena observer ini terbukti optimal dalam mengestimasi variabel keadaan yang takterukur dalam penerapannya di berbagai sistem. Untuk observer lain selain *Kalman Filter,* masih Keunggulan *Kalman Filter* adalah kemampuannya untuk mengestimasi *state* pada

waktu lampau, sekarang, maupun di waktu mendatang, bahkan ketika karakteristik spesifik dari model yang akan diestimasi tidak diketahui. Keunggulan lainnya adalah metode ini dapat diimplementasikan dengan mudah pada sistem tata udara presisi ini.

1.2 Perumusan Masalah

Masalah pokok dalam riset ini adalah mengimplementasikan algoritma *Kalman Filter* untuk memecahkan masalah estimasi variabel takterukur pada sistem tata udara presisi karena di dalam sistem tata udara presisi masih banyak variabel tak terukur seperti temperatur di evaporator, temperatur di kondenser, temperatur di peralatan IT, dan lain-lain. Variabel-variabel tak terukur ini akan diestimasi nilainya menggunakan *Kalman Filter*. Algoritma *observer Kalman Filter* ini terdiri dari *time udpate* dan *measurement update* yang didalamnya terdapat algoritma untuk mengkalkulasi *kalman gain*, meng*update* nilai *error covariance*, dan juga algoritma untuk meng*update state* estimasi. Diharapkan melalui metode Filter Kalman ini, diperoleh *state* yang cukup akurat yang digunakan dalam perancangan pengendali pada sistem tata udara presisi.

1.3 Tujuan Penelitian

Penelitian ini dilakukan untuk merancang suatu observer untuk estimator *state* dengan menggunakan metode Filter Kalman yang diterapkan pada sistem tata udara presisi agar dapat mengestimasi variabel keadaan yang takterukur sehingga dapat dicapai temperatur dan kelembaban sistem tata udara presisi berada pada nilai yang diinginkan. Dengan algoritma yang sesuai nantinya diharapkan dapat membuat sistem tata udara presisi yang digunakan dapat bekerja pada nilai temperatur dan kelembaban yang diinginkan sehingga dapat mengurangi konsumsi daya yang digunakan oleh alat tersebut. Algoritma pengendali ini juga mengikutsertakan RH (derajat kelembaban), disamping temperatur pada desain pengendalian.

1.4 Pembatasan Masalah

Masalah pada skripsi ini dibatasi pada perancangan algoritma Kalman Filter untuk mengestimasi variabel ruang keadaan takterukur pada sistem tata udara presisi kompresor DC dengan menggunakan perangkat lunak Matlab (R2009a), yang pada desain pengendaliannya keluaran hasil estimasi *state* ini akan menjadi nilai *state* untuk pengendali MPC untuk mengendalikan temperatur dan kelembaban sistem tata udara presisi berada pada batasan yang diinginkan.

Model yang dipakai adalah model linier diskrit hasil dari idetifikasi menggunakan N4SID pada penelitian sebelumnya. Hasil estimasi variabel takterukur menggunakan algoritma Filter Kalman akan dibandingkan dengan nilai sebenarnya dari variabel ruang keadaan tersebut.

Pada penelitian ini dilakukan dengan beberapa batasan masalah, antara lain :

1. Peneliti tidak melakukan proses pemodelan dan identifikasi sistem tata udara presisi dengan menggunakan kompresor DC. Model yang digunakan telah didapat dari penelitian yang telah dilakukan sebelumnya (Victor, 2011).
2. Peneliti hanya mendesain algoritma *Kalman Filter* karena model yang digunakan adalah model linier.
3. Peneliti tidak membahas algoritma pengendali *Model Predictive Control (MPC)* maupun LQR yang digunakan untuk mengendalikan keluaran dari sistem tata udara presisi.

1.5 Metodologi Penelitian

Metologi penelitian yang dipakai pada skripsi ini di antaranya:

1. Studi literatur, yaitu dengan membaca jurnal dan skripsi maupun tesis mengenai Filter Kalman serta sistem tata udara presisi.
2. Konsultasi dengan dosen pembimbing dan berdiskusi dengan teman yang melakukan penelitian mengenai sistem tata udara presisi.

3. Merancang observer Filter Kalman menggunakan perangkat lunak Matlab (2009a), yaitu dengan menggunakan M-File dan menggunakan C-Mex.
4. Menguji observer Filter Kalman pada sistem yang lebih sederhana dengan menggunakan model sistem CSTR *(Continuous Stirred* Tank *Reactor).*

1.6 Sistematika Penulisan

Laporan skripsi ini akan dibagi menjadi lima bab, di mana masing-masing memuat hal berikut:

1. Bab 1: Pendahuluan

   Bab ini menjelaskan tentang latar belakang, perumusan masalah, tujuan penelitian, pembatasan masalah, metodologi penelitian, dan sistematika penulisan.

2. Bab 2: Dasar Teori

   Bab ini menjelaskan dasar teori mengenai konsep dasar algoritma Filter Kalman dan penentuan nilai dari matriks kovarian sistem, serta membahas sistem tata udara presisi beserta persamaan matematisnya.

3. Bab 3: Perancangan Observer Filter Kalman untuk Sistem Tata Udara Presisi

   Pada bab ini, Penulis menjelaskan langkah-langkah yang dilakukan di dalam merancang observer Filter Kalman. Pertama dijelaskan mengenai perancangan algoritma Filter Kalman dengan variasi noise yang digunakan, selanjutnya dijelaskan mengenai penggunaan variasi nilai matriks kovarian error proses dan matriks kovarian error pengukuran. Dijelaskan pula penggunaan algoritma penetuan optimasi nilai matriks kovarian error proses dan matriks kovarian error pengukuran yang didapat dari jurnal. Pada bab ini juga dijelaskan tentang penerapaan algoritma Filter Kalman pada sistem tata udara presisi yang sebelumnya diujikan terlebih dahulu untuk sistem yang lebih sedeerhana yaitu dengan dengan

menggunakan model sistem CSTR *(Continuous Stirred Tank Reactor)* baik rancangan menggunakan M-File maupun menggunakan C-Mex.

4. Bab 4: Hasil Simulasi dan Analisis

   Bab ini membahas hasil estimasi *state* takterukur menggunakan Filter Kalmanpada sistem tata udara presisi maupun sistem yang lebih sederhana yaitu model sistem CSTR. Variasi noise maupun kovarian error proses dan pengukuran juga dibahas dalam bab ini. Akan dijelaskan analisis dari setiap percobaan yang dilakukan tersebut. Semua hasil estimasi *state* takterukur tersebut dianalisis berdasarkan nilai kesalahannya.

5. Bab 5: Kesimpulan dan Saran

   Pada bab ini, Penulis menyimpulkan hasil percobaan dan analisis yang dilakukan serta menuliskan saran-saran praktis yang berguna bagi pembaca yang menggunakan, mempelajari, melanjutkan, ataupun mengembangkan percobaan yang telah dilaporkan pada laporan skripsi ini.

BAB 2
DASAR TEORI

2.1 *State of the Art* Kalman Filter

Kalman Filter banyak digunakan dalam berbagai aplikasi. Fungsi dari Kalman Filter itu sendiri adalah sebagai estimator yang handal dalam berbagai sistem yang digunakan. Modelnya yang sederhana sehingga mudah diterapkan dalam berbagai sistem. Dengan memperhitungkan white noise yang merupakan noise yang diestimasi pada seluruh cakupan frekuensi, sehingga kalman filter langsung dapat digunakan sebagai estimator tanpa perlu menghitung noise yang terjadi pada sistem secara detil terlebih dahulu. Kalman filter dapat digunakan untuk mengestimasi sistem yang linear. Dalam perkembangannya, untuk sistem yang lebih kompleks dengan persamaan matematis yang linear, kalman filter dimodifikasi agar dapat mengestimasi sistem yang non linear, modifikasi kalman filter ini ada yang dinamakan *extended kalman filter (EKF), fractional kalman filter (FKF),* dan juga *uncented kalman filter* (UKF).

Berbagai aplikasi *kalman filter* dapat diterapkan dalam banyak sistem. Di tahun 2003 John Valasek dan Wei Chen [7], menggunakan observer kalman filter untuk mengidentifikasi secara online sistem pesawat. Masalah sistem identifikasi online muncul dari keakurasian, locally linear, serta model dinamik pesawat dari nonlinear pesawat. Metode identifikasi kalman filter ini cocok untuk mengidentifikasi secara online sistem pesawat dari model pesawat locally linear dan secara umum cukup intensif mengendalikan intensitas *white noise* sensor gaussian dan untuk menyalakan intensitas hembusan diskrit.

Di tahun yang sama, Zhuang Xu dan M. F. Rahman [25] menggunakan *extended kalman filter (EKF)* untuk mengestimasi kecepatan rotor pada saat kecepatan yang sangat rendah. Untuk dapat mengestimasi pada kecepatan yang sangat rendah diperlukan sensitivitas yang tinggi dari estimator untuk model nonlinear, gangguan *(disturbance),* dan model parameter detuning. Telah dilakukan riset mengenai prinsip dari *direct torque control (DTC)* diimplementasikan pada *interior permanent magnet (IPM).* Di tahun sebelumnya

telah dilakukan penelitian mengenai IPM namun belum dapat mengestimasi kecepatan rotor pada saat kecepatan yang sangat rendah. Di penelitian ini estimator *extended kalman filter (EKF)* terbukti memilki *dynamic behavior* yang lebih baik, serta memiliki kemampuan estimasi resitansi gangguan dan memilki akurasi tinggi.

Di tahun 2003 juga kalman filter dipakai oleh D. Loebis, R. Sutton, J. Chudley, dan W. Naeem [4] untuk melakukan riset mengenai penerapan sistem navigasi cerdas, didasarkan pada penggunaan yang terintegrasi dari *global positioning system (GPS)* dan beberapa sistem navigasi inersia *(inertia system navigation/INS)* sensor, untuk aplikasi kendaraan otonom di bawah air (AUV). Dalam riset ini SKF dan EKF digunakan untuk memadukan data dari sensor INS dan untuk mengintegrasikannya dengan data GPS. Selain itu juga digunakan teknik logika fuzzy untuk adaptasi dari asumsi statistik awal dari keduanya (SKF dan EKF), disebabkan oleh kemungkinan perubahan karakteristik noise sensor. Setelah dilakukan estimasi, perbaikan estimasi dari SKF dan EKF dan meningkatkan akurasi keseluruhan dari integrasi GPS.

Pada tahun 2004, Pratap R.[13], melakukan penelitian mengenai *Extended Kalman Filter (EKF)* yang digunakan untuk menyaring masuknya noise ke dalam reaktor biologis, penelitian ini dilakukan karena mikroba yang ada di dalam reaktor biologis bisa terpengaruh oleh noise apabila noise ini tidak difilter. Noise ini disebabkan oleh dua sumber yang mempengaruhi kinerja yaitu noise pengukuran dari proses sinyal sensor (pH, suhu, kecepatan agitasi, laju aliran, dan lain-lain) dan noise dari lingkungan. Penelitian sebelumnya yang dilakukan oleh M. L. Shuler dan G. Liden menunjukkan bahwa kompleksitas dan efek pH berpotensi berbahaya pada perilaku mikroba, karena efek ekonomis, kinetik, maupun benefit. Dari hasil riset ini EKF telah terbukti dapat menyelamatkan osilasi periodik yang stabil yang telah terdistorsi oleh noise, serta EKF telah terbukti efektif dalam menyaring noise dari aliran tersebut bahwa sekitar osilasi bebas noise dapat dipulihkan.

Di tahun 2005, *kalman filter* digunakan untuk mengestimasi suhu internal, dengan menggunakan Kalman filter, diterapkan pada sistem hibrida linier oleh L. Boillereaux, H. Fibrianto, dan J. M. Flaus [9]. Metode ini diterapkan karena

keterbatasan penggunaan sensor invasif, sehingga untuk memperkirakan suhu internal dari makanan hanya dapat diperoleh dengan pengukuran di permukaan.

Setahun kemudian, Jose dan Wan Yu [8] membandingkan algoritma pembelajaran normal *(backpropagation)* dengan kalman filter untuk mengidentifikasi sistem nonlinier dimana modifikasi *dead-zone robust* diterapkan pada kalman filter. Kalman filter diterapkan untuk melatih *state space* jaringan saraf tiruan berulang untuk identifikasi sistem nonlinier. Dimana riset serupa telah dilakukan di tahun-tahun sebelumnya yaitu mengenai analisa konvergensi neural network, beberapa teknik modifikasi robust pada algoritma *least square*, analisa kestabilan, dan konvergensi kalman filter untuk model linear *stochastic regresi time-varying*. Dari hasil penelitiannya algoritma kalman filter memiliki beberapa sifat yang lebih baik, seperti konvergensi yang yang lebih cepat, meskipun algoritma ini lebih kompleks dan sensitif terhadap sifat noise. Serta metode Lyapunov yang digunakan untuk membuktikan bahwa pelatihan (training) kalman filter stabil.

Di tahun 2006 juga, A. Tianoa, R. Suttonb, A. Lozowicki, dan W. Naeem [1] menggunakan observer kalman filter untuk identifikasi model linier waktu diskrit multivariabel dari kendaraan bawah air (AUV). Dimana *Observer Kalman Filter Identification (OKID)* digunakan untuk mengevaluasi efektivitas untuk identifikasi eksperimental perilaku dinamis dari sebuah AUV. Penggunaan observer ini karena pengendalian yang belum optimal pada AUV karena kesulitan untuk menentukan model matematika online perilaku dinamis dari AUV yang dapat diandalkan sehingga diperlukan suatu algoritma yang mampu mengestimasinya. Hasil yang dicapai dalam riset ini menunjukkan bahwa metode OKID dapat menjadi alat yang efisien untuk identifikasi eksperimen dinamika AUV.

Brendan M. Quine [3] pada tahun 2006, mengembangkan model estimasi nonlinier baru yang memiliki operasi yang sama dengan EKF tetapi sangat lebih mudah diimplementasikan pada model aplikasi kompleks. Yang bertujuan untuk mengatasi estimasi *mean dan covarian* pada *state* sistem yang harus diketahui ketika filter diinisialisasi. Karena menurut penelitian sebelumnya filter rekursif pada umumnya menghasilkan optimasi optimal minimum kesalahan *state* dan

dalam banyak kasus analisis (Bar-Shalom dan Fortmann, 1988). Dari riset ini didapat kesimpulan bahwa EKF ideal untuk implementasi sistem kompleks nonlinier dan model observasi.

Di tahun 2007, A.Vasebi, S. M. T. Bathaee, dan M. Partovibakhsh [2] menggunakan metode extended kalman filter untuk memperkirakan *state of charge* (SoC) dari baterai asam timbal pada kendaraan listrik hybrid (HEV/hybrid electric vehicle). Estimasi ini dilakukan karena banyak masalah terjadi pada indikator SoC tradisional, seperti offset, drift dan *state* divergensi panjang. Dengan menggunakan EKF pada penelitian ini, menunjukkan bahwa metode EKF teknik yang lebih unggul daripada metode tradisional, dengan akurasi dalam memperkirakan SoC mencapai 3%.

Di tahun ini juga Weihua Li, Sirish L. Shaha, dan Deyun Xiao [20], mengembangkan sebuah data *driven kalman filter* untuk sebuah sistem *Non-Uniformly Sampled Multirate (NUSM)* yang bertujuan untuk menyelidiki metode kalman filter untuk deteksi tunggal dan isolasi dari sensor, aktuator, dan kesalahan proses dalam sistem NUSM dengan analisa kemampuan deteksi kesalahan dan isolabilitas. Dikarenakan adanya non-uniformly sampled multirate pada sistem.

Selain itu pada tahun 2007 Mickael Hilairet, Francois Auger, dan Eric Berthelot [10] memodifikasi Kalman filter, yang digunakan untuk mengestimasi fluks rotor dan kecepatan rotor pada motor induksi. Estimasi ini diperlukan untuk menentukan kecepatan dan posisi rotor dari tegangan dan arus stator apabila tanpa menggunakan sensor kecepatan dan sensor posisi. Estimator *kalman filter* modifikasi ini dapat mengurangi jumlah operasi aritmatika sampai 25% daripada menggunakan kalman filter biasa. Serta estimator ini memperbolehkan menggunakan *sampling rate* yang lebih tinggi menggunakan sebuah *mikrokontroller* yang lebih murah.

Di tahun 2008, J. Kim [6] mengenai *extended kalman filter* yang digunakan untuk mengidentifikasi gaya dinamik ban lateral, identifikasi ini dilakukan karena adanya interaksi antara ban dengan permukaan jalan yang merupakan fungsi nonlinier pada beberapa variabel, seperti slip longitudinal, sudut sampling slip, beban normal, sudut camber, tekanan ban, temperatur, dan

karakteristik permukaan jalan. Dengan menggunakan *extended kalman filter*, didapat keakurasian model estimasi ban lateral.

Di tahun yang sama X. Luoa , I. M. Moroz [21], memodifikasi skema *ensemble kalman filter (EnKF)* menggunakan konsep uncented transform yang merupakan sebuah konsep metode baru untuk transformasi nonlinear dari mean dan kovarian dalam filter dan estimator. Hal ini disebabkan oleh adanya error distribusi analisis simetri (tidak membutuhkan gaussian) apabila menggunakan EnKF biasa menghasilkan estimasi yang kurang akurat. Metode EnKF ini terbukti dapat memberikan estimasi yang akurat.

Di tahun 2009, *kalman filter* digunakan oleh W. J. Sung, S. C. Lee, dan K. H. You [17] untuk memprediksi gangguan untuk sistem posisi. Tujuannya adalah untuk mendesain *hybrid controller* berdasarkan *adaptive fuzzy* Kalman filter observer untuk memprediksi *noise* pengukuran. Sehingga metode yang digunakan merupakan gabungan antara *kalman filter* dan *adaptive fuzzy*.

Pada tahun 2009 Salvatore, Velardi, Hassan, dan Antonello [15] melakukan pemantauan tahap pengeringan utama dari proses lyophilisasi obat-obatan dalam botol. Pemantauan ini diperlukan untuk memastikan bahwa suhu maksimum produk dalam botol dipertahankan pada nilai yang aman untuk menghindari terjadinya denaturasi. Namun untuk mencapai hal tersebut terjadi permasalahan karena komputasi yang dilakukan sistem terlalu kompleks, sehingga dibutuhkan suatu algoritma observer untuk mengatasinya. Di sini EKF berfungsi untuk menyederhanakan model proses yang digunakan untuk mengurangi beban komputasi.

Murat Barut [11] pada tahun 2009 juga menerapkan extended kalman filter pada penelitiannya. Tujuan penelitiannya yaitu mengestimasi secara online masalah yang berkaitan dengan ketidakpastian dalam stator rotor dan resistensi melekat dengan kontrol sensorless dengan efisiensi motor induksi (IM) yang tinggi dalam rentang kecepatan yang luas serta memperluas jumlah *state* yang terbatas dan estimasi parameter menggunakan algoritma EKF tunggal dengan eksekusi berturut-turut dari dua input yang berasal dari dua model IM berdasarkan resistansi stator dan estimasi resistansi rotor. Penelitian ini dilakukan karena adanya ketidakpastian estimasi *state* parameter elektrik dan mekanik dari motor

induksi, serta suhu dan frekuensi bergantung pada variasi resistansi rotor dan stator yang terdapat di seluruh bagian penting dari ketidakpastian elektrik dalam sebuah motor induksi selama torsi beban dan friksi menentukan mekanik utamanya. Dan juga *speed sensorless control* saat titik operasi pada kecepatan sangat rendah atau kecepatan nol saat steady *state* dibawah kondisi tanpa beban (no load). Hasil simulasi menunjukkan bahwa sistem kendali speed sensorless direct vector menggunakan teknik estimasi EKF cukup berhasil.

Di tahun 2010 Wang Jianlin [20] menggunakan EKF untuk sistem estimasi online variabel biologis terukur, dimana EKF digunakan untuk model *state* space untuk mengurangi gangguan *noise* dalam proses fermentasi.

Di tahun 2011 Yongjin Kwon dan Yongmin Park [23], menerapkan kalman filter yang digunakan untuk memperbaiki sifat seragam distorsi gambar untuk meningkatkan keakurasian robot. Hal ini dilakukan karena dalam kamera, lensa yang kurang sempurna menginduksi distorsi gambar, sumber utama dalam akurasi posisi, efek distorsi lensa dapat diperbaiki dengan menerapkan algoritma koreksi. Serta sulitnya dalam kalibrasi visi, yaitu akurasi jarak jauh. Ketidakseragaman distorsi gambar terjadi akibat kelengkungan lensa tidak sempurna hampir di semua area kerja robot, yang pada gilirannya menginduksi sebuah bimbingan visi yang tidak akurat. Dengan menggunakan teknik kalman filter, sifat seragam non distorsi gambar secara efektif dan juga akurasi posisi robot secara signifikan ditingkatkan.

Di tahun yang sama juga, Zongbo Xie dan Jiuchao Feng [24] memperkenalkan teknik baru yaitu *Iterated Uncented Kalman Filter* (IUKF) yang merupakan modifikasi dari UKF biasa, untuk identifikasi sistem nonlinier struktural (NSSI). Modifikasi ini dilakukan karena sulit menerapkan UKF untuk sistem struktur yang sangat nonlinear terutama yang dikenakan beban berat. Penelitian mengenai identifikasi sistem nonlinier structural telah banyak digunakan, diantaranya X. J. Hu, Berana, serta R. M. Crujeiras menggunakan metode least square estimation (LSE), R. Kandepu, K. Xiong, R. Van Der Merwe, dan R. Zhan menggunakan uncented kalman filter [UKF), H. Gao dan H. Urkowitz menggunakan metode H filter, I. Yoshida, Y. Tanaka, S. J. Li menggunakan metode *sequential Monte Carlo* untuk sistem tersebut. IUKF

menghasilkan estimasi keadaan yang lebih baik daripada identifikasi parameter dari UKF, dan IUKF juga lebih robust untuk pengukuran tingkat noise.

Selain itu juga, di tahun 2011 dilakukan uji keakurasian tracking dan teknik estimasi untuk besaran, frekuensi, dan fasa tegangan kerdipan (flicker) menggunakan kalman filter, yang dilakukan oleh H. M. Al-Hamadi [5], dimana menggunakan model extended *state* space untuk mengestimasi parameter. Dengan menggunakan algoritma kalman filter, konvergensi dari estimasi parameter yang didapat, nilai parameter estimasinya sangat dekat dengan nilai aslinya.

Secara umum, kalman filter banyak digunakan sebagai estimator dalam berbagai permasalahan, karena kalman filter sendiri memiliki keunggulan yaitu kemampuannya untuk mengestimasi *state* pada waktu lampau, sekarang, maupun di waktu mendatang, bahkan ketika karakteristik spesifik dari model yang akan diestimasi tidak diketahui.

Tabel 2.1 Daftar Paper Perkembangan Filter Kalman

No.	Judul Paper	Kata Kunci	Jenis Filter Kalman Yang diaplikasikan	Model (Jenis Sistem)
1.	In line monitoring of the primary drying phase of the Freeze drying process in vial by means of a Kalman filter based observer	Freeze-drying; Lyophilisation; Primary drying; Modelling; Monitoring; Extended Kalman filter	EKF	Nonlinier
2.	Non linear system identification with recurrent neural networks and dead-zone Kalman filter algorithm	Identification; Neural networks; Kalman filter; Stability	EKF	Nonlinier
3.	Observer Kalman filter identification of an autonomous	System identification; Parameter	Kalman Filter	Linier

	Underwater vehicle	identification; Numerical methods; Vehicle dynamics; Underwater		
4.	A switched Kalman filter dedicated to Assisted pressure food thawing	Switched linear systems; Kalman filter; Heat transfer; Phase change; High pressure	EKF	Nonlinier
5.	Adaptive tuning of a Kalman filter via fuzzy logic for an intelligent AUV navigation system	Autonomous underwater vehicles; Navigation; Sensorfusion; Kalman filters; Extended Kalman filters; Fuzzy logic Probalistic neural network	SKF dan EKF	Nonlinier
6.	The extended Kalman filter as a noise modulator for continuous yeast Cultures under monotonic, oscillating and chaotic conditions	Extended Kalman filter; Continuous culture; Inflow noise; Oscillations; Chaos	EKF	Nonlinier
7.	Bi Input-extended Kalman filter based estimation technique for speed-sensorless control of induction motors	Induction motor; Extended Kalman filter; Rotor and stator resistance estimation; Load torque estimation; Sensorless	EKF	Nonlinier

		control; Zero speed operation		
8.	On-line Estimation in Fed-batch Fermentation Process Using *State* Space Model and Unscented Kalman Filter	online estimation; simplified mechanistic model; support vector machine; particle swarm optimization; unscented Kalman Filter	EKF	Nonlinier
9.	Improvement of vision guided robotic accuracy using Kalman filter	Remote vision calibration; Kalman filter; Operational efficiency EQM (e-quality for manufacture)	Kalman Filter	Linear
10.	Predicting *state* of charge of lead-acid batteries for hybrid electric vehicles by extended Kalman filter	Batteries; Extended Kalman filter; Hybrid electric vehicle; *State* of charge	EKF	Nonlinier
11.	Identification of lateral tyre force dynamics using an extended Kalman filter From experimental road test data	Extended Kalman filter; Lateral tyre force; Magic formula; Vehicle dynamics; Relaxation length	EKF	Nonlinier
12.	Speed and rotor flux estimation of induction machines using a two-stage extended Kalman filter	Keywords: Induction machine; Non-linear system; Kalman estimator;	EKF	Nonlinier

		Estimation theory		
13.	Real-time nonlinear structural system identification via iterated unscented Kalman filter	Keywords: Nonlinear structural system identification; Iterated unscented Kalman filter; Real-time	UKF, IUKF	Nonlinier
14.	Real-time free way traffic *state* estimation based on Extended Kalman filter : a general approach	Freeway; Traffic *state* estimation; Stochastic macroscopic traffic flow model; Extended Kalman filter; Model Parameter estimation	UKF	Nonlinier
15.	Ensemble Kalman filter with the unscented transform	Ensemble Kalman filter; Unscented transform; Ensemble unscented Kalman filter	Ensemble Kalman Filter (EnKF)	Nonlinier
16.	A derivative-free implementation of the extended Kalman filter	Stochastic modelling and nonlinear models; *State* and parameter estimation; Kalman filters	EKF	Nonlinier
17.	Kalman filters in non-uniformly sampled multirate systems: For FDI and beyond	Non-uniformly	Kalman Filter	*non-uniformly sampled multirate*

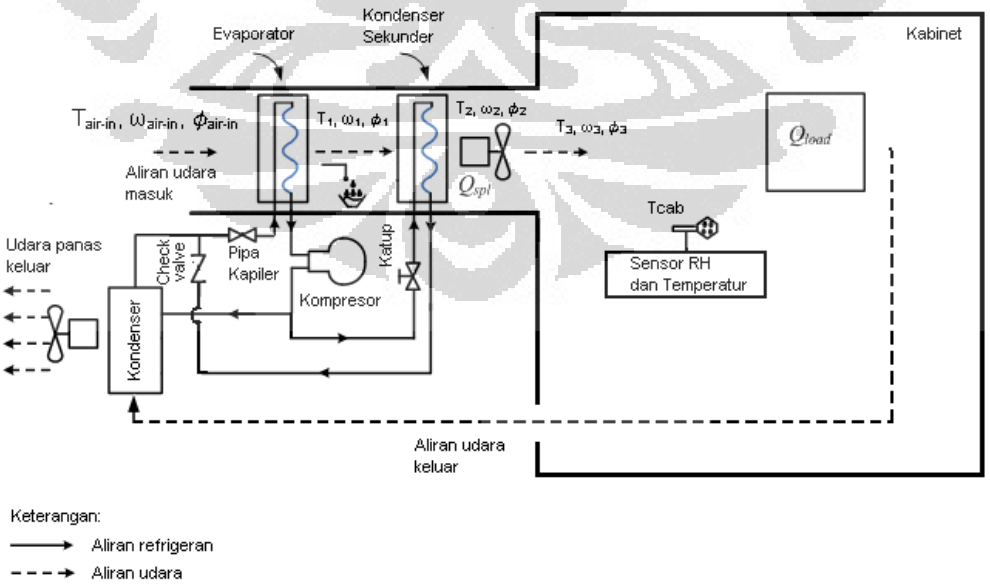
		sampled multirate systems; Kalman filters; One-step predictor; Filtering; Unified fault detection and isolation		
18.	Fuzzy logic voltage flicker estimation using Kalman filter	Fuzzy logic; Voltage flicker; Power quality; Adaptive filters; Kalman filter	Kalman Filter, Extended *state* space	Nonlinier
19.	An adaptive high-gain observer for nonlinear systems	Nonlinear observer; Adaptive high gain observer; Kalman filtering	EKF	Nonlinear, Adaptive high gain
20.	Ultra-precision positioning using adaptive fuzzy-Kalman filter observer	Ultra-positioning; Kalman Filter; Covariance matching; Fuzzy control; sliding-mode control	Fuzzy Kalman Filter	Nonlinier
21.	Observer/Kalman Filter Identication for Online System Identcation of Aircraft		EKF	Nonlinier
22.	An Extended Kalman Filter Observer for the Direct Torque Controlled Interior Permanent Magnet Synchronous Motor Drive	IPMSM; direct torque control; EKF	EKF	Nonlinier

2.2 Sistem Tata Udara Presisi

Sistem tata udara presisi (*Precision Air Conditioning* atau PAC) merupakan sistem refrigerasi yang bekerja berdasarkan konsep termodinamika. Mesin refrigerasi adalah alat yang melakukan proses perpindahan kalor dari media bersuhu tinggi ke media bersuhu rendah dengan memanfaatkan siklus refrigerasi (*vapor-compression cycle*).

PAC banyak digunakan di berbagai kebutuhan industri maupun rumah tangga. Penggunaan PAC ini dapat mengefisiensikan penggunaan energi. Pada skripsi ini, penggunaan PAC difokuskan pada penggunaan industri, terutama di ruang pusat data (*data center*). PAC ini digunakan untuk mengendalikan suhu dan kelembaban udara relatif pada kabinet yang ada di ruang pusat data, sehingga suhu dan kelembabannya terjaga konstan di nilai tertentu. Hal ini bertujuan untuk menjaga peralatan IT bisa beroperasi secara kontinu dengan meminimalkan kemungkinan kerusakan. Tujuan lainnya adalah untuk menjaga usia pemakaian peralatan IT tersebut agar bertahan lama. Tujuan lain yang seperti telah disebutkan di atas adalah pengefisienan energi, sehingga pengeluaran perusahaan bisa ditekan menjadi lebih murah. Suhu ideal untuk peralatan IT sekitar 20-22^{o}C dan kelembaban relatif ideal untuk peralatan IT sekitar 45-55%.

Berikut adalah bagan PAC yang akan diidentifikasi pada skripsi ini:



Gambar 2.1 Bagan Sistem Tata Udara Presisi

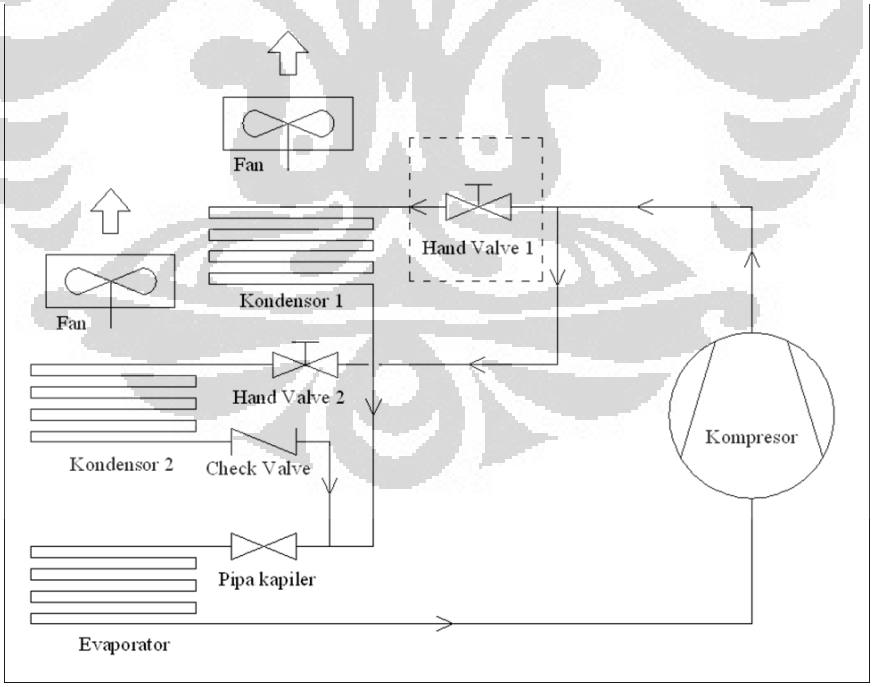
Skripsi ini merupakan modifikasi dari skripsi maupun jurnal tentang PAC yang telah dilakukan sebelumnya oleh Rise Hapshary Surayuda (Surayuda, 2010), Sutarna (Sutarna, 2008), dan Victor (Victor, 2011). Pembahasan yang dilakukan dalam skripsi ini pun hanya meliputi bagian-bagian yang seperlunya yang berkaitan dengan pemodelan PAC.

2.2.1 Prinsip Kerja Tata Udara Presisi

Ada beberapa komponen yang ada dalam sistem tata udara presisi ini, yaitu:

- Kompresor
- Evaporator
- Dua buah kondenser
- Dua buah kipas (*fan*)
- Pipa Kapiler
- Katup (electronic valve)

Berikut adalah skema PAC beserta penjelasan cara kerja sistem tersebut:

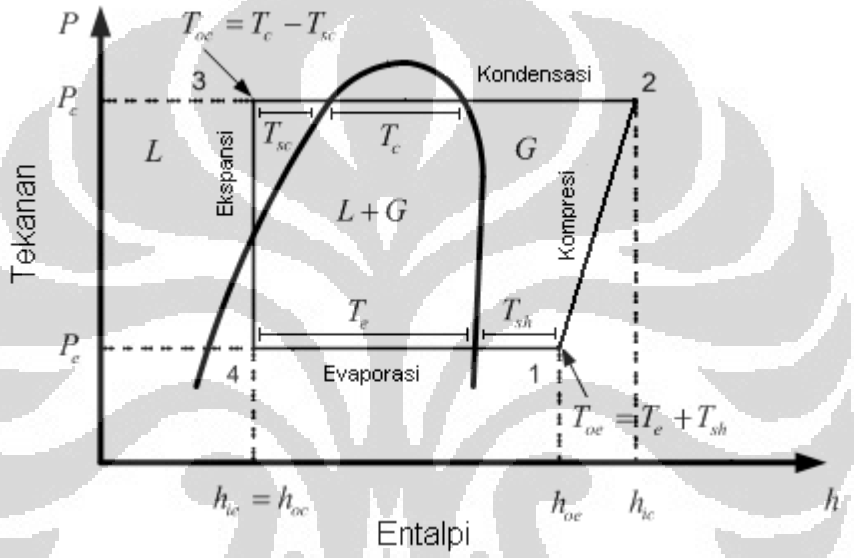


Gambar 2.2 Diagram Pipa Sistem Tata Udara Presisi

Pada sistem tata udara presisi ini, ada dua aliran fluida yang perlu diperhatikan. Pertama adalah aliran udara dan kedua adalah aliran refrigeran. Fluida yang digunakan sebagai refrigeran pada PAC ini adalah R134a.

Aliran udara ditarik masuk oleh *fan* dari ruang pusat data ke dalam kabinet, melalui PAC. Udara tersebut pertama akan melalui evaporator, lalu melewati kondenser kedua dan akhirnya masuk ke dalam kabinet. Udara dari dalam kabinet akan dibuang keluar oleh kondenser kedua.

Berikut akan dijelaskan siklus refrigeran yang mengalir pada PAC:



Gambar 2.3 Diagram P-h Siklus Refrigerasi

Ada 4 tahap yang dialami oleh refrigeran, yaitu:

- Tahap Kompresi

Saat PAC dinyalakan, kompresor mulai bekerja menaikkan tekanan refrigeran dan mengalirkannya ke kondenser pertama. Refrigeran yang keluar dari kompresor dalam fasa gas dengan tekanan dan suhu yang tinggi.

- Tahap Kondensasi

Tahap ini berlangsung di kondenser pertama. Pada kondenser pertama suhu refrigeran lebih tinggi dibandingkan suhu udara di luar, sehingga terjadi perpindahan kalor dari refrigeran ke lingkungan luar yang menyebabkan udara yang dibuang oleh kondenser pertama menjadi lebih panas. Akibat pembuangan

panas dari refrigeran ke lingkungan luar, terjadi proses kondensasi sehingga terjadi perubahan fasa pada refrigeran dari gas ke cair.

- Tahap Ekspansi

Tahap ini terjadi di pipa kapiler. Refrigeran R134a cair yang keluar dari kondenser pertama kemudian mengalir ke dalam pipa kapiler. Di sini tekanan refrigeran menurun drastis karena ada efek penghambatan oleh alat ekspansi.

- Tahap Evaporasi

Tahap evaporasi ini terjadi di evaporator. Refrigeran yang tekanan dan suhunya telah rendah itu masuk ke dalam evaporator. Di evaporator suhu refrigeran lebih rendah dibandingkan suhu udara dalam PAC. Oleh karena itu, terjadi penyerapan kalor oleh refrigeran yang mengakibatkan suhu udara yang keluar dari evaporator (T_1) lebih rendah dibanding dengan suhu udara pada ruang pusat data ($T_{air\text{-}in}$) yang masuk ke dalam evaporator dengan kelembaban relatif (RH) yang tinggi. Akibat penyerapan kalor ini, terjadi proses evaporasi sehingga refrigeran berubah fase dari cair ke gas.

Refrigeran dalam bentuk gas yang keluar dari evaporator kemudian masuk kembali ke kompresor untuk dikompresi, dan begitu seterusnya. Siklus ini dikenal dengan siklus refrigerasi atau *vapor-compression cycle* seperti ditunjukkan pada Gambar 2.3. PAC yang yang digunakan pada skripsi ini menggunakan kondenser tambahan, sehingga refrigeran yang keluar dari kompresor tidak hanya mengalir ke kondenser pertama, tapi juga ke kondenser kedua jika katup kondenser kedua dibuka. (lihat Gambar 2.2).

Pada kondenser kedua, terjadi tahap kondensasi sepertu yang terjadi pada kondenser pertama. Di kondenser kedua, suhu refrigeran lebih tinggi dibandingkan suhu udara di dalam PAC, sehingga terjadi pembuangan panas dari refrigeran ke udara luar. Hal ini menyebabkan suhu udara yang keluar dari kondenser kedua (T_2) lebih tinggi dibanding suhu udara yang masuk ke kondenser pertama (T_1) dengan kelembaban relatif yang lebih rendah juga ($\phi 2 < \phi 1$). Adanya kenaikan suhu tersebut, maka kondenser kedua ini berfungsi sebagai kondenser reheat, yaitu untuk memanaskan kembali udara yang keluar dari evaporator karenaumumnya udara yang keluar dari evaporator memiliki suhu yang rendah dan kelembaban yang tinggi.

2.2.2 Persamaan Matematis Sistem Tata Udara Presisi

Dalam membahas persamaan matematis dan selama proses identifikasi, ada beberapa asumsi yang perlu diperhatikan, antara lain:

- Campuran udara terjadi di dalam evaporator, kondenser dan lingkungan
- Suhu evaporasi di evaporator dianggap konstan
- Sisi udara di evaporator meliputi daerah kering (*dry region*) dan daerah basah (*wet region*)
- Perbandingan volume udara *dry region* terhadap volume udara sisi *wet region* adalah 1:4
- Suhu kondensasi di kondenser dianggap konstan
- Sisi udara di kondenser hanya meliputi daerah kering saja
- Aliran refrigerant yang mengalir ke kondenser kedua diasumsikan sebanyak 10% dari *mass flow refrigerant total*, yaitu aliran refrigerant yang keluar dari kompresor atau aliran refrigerant yang masuk ke evaporator ataupun keluar dari evaporator
- Tekanan di kompresor dianggap konstan
- Beban diangap konstan
- Kecepatan aliran udara volumetric (*air volumetric flow*) dalam sistem dianggap konstan
- Rugi-rugi panas pada daerah aliran udara diabaikan

Sistem tata udara presisi ini, keluaran yang akan kita kendalikan adalah suhu kabinet (T_{cab}) dan kelembaban relatif kabinet (ω_{cab}). Persamaan matematis yang dipakai adalah model kompresor, model evaporator, model kondenser kedua, model udara masuk kabinaet (*supply air*) dan model kabinet. Sedangkan model kondenser pertama diabaikan karena kondenser pertama tidak berpengaruh terhadap suhu dan kelembaban relatif di kabinet. T_{cab} dan ω_{cab} didapat tanpa memerlukan informasi mengenai udara yang dibuang keluar oleh kondenser pertama.

2.2.3 Model Kompresor

Persamaan matematis dari model kompresor yang digunakan untuk sistem tata udara presisi ini adalah sebagai berikut:

$$M_{ref} = \frac{sV_{com}}{v_s}\left(1 = 0{,}015\left[\left(\frac{P_c}{P_e}\right)^{\frac{1}{\beta}} - 1\right]\right) \tag{2.1}$$

di mana

M_{ref} : aliran massa refrigeran total keluaran kompresor (kg/s)

s : kecepatan kompresor (rps)

V_{com} : *swept volume* kompresor (m^3)

v_s : volume spesifik dari *superheat refrigerant* (m^3/kg)

P_c : tekanan kondensasi (kPa)

P_e : tekanan evaporasi (kPa)

β : indeks kompresi

Swept volume pada persamaan (2.1) dicari dengan:

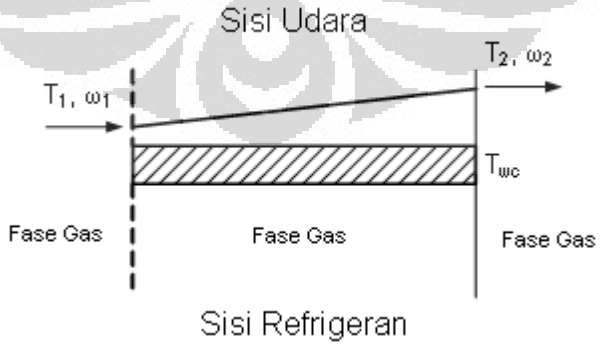
$$V_{com} = \frac{V_d}{n_c} \tag{2.2}$$

di mana

V_d : displacement volume compressor (m^3)

n_c : jumlah silinder pada kompresor

2.2.4 Model Kondenser Kedua



Gambar 2.4 Skema Aliran Udara di Kondenser Kedua

Gambar 2.4 menunjukkan skema diagram aliran udara di kondenser kedua, yang berbeda dengan skema aliran udara evaporator karena kondenser hanya memiliki daerah *dry region*.

Persamaan matematis untuk kondenser kedua pada sistem tata udara presisi adalah:

$$C_{pu}\rho_u V_{wc2}\frac{dT_2}{dt} = C_{pu}\rho_u f\left(T_1 - T_2\right) + UA_3\left(T_{wc2} - \frac{T_1 + T_2}{2}\right) \tag{2.3}$$

Persamaan matematis dinding kondenser kedua pada sistem tata udara presisi adalah:

$$C_{pw}\rho_w V_{wc2}\frac{dT_{wc2}}{dt} = UA_3\left(\frac{T_1 + T_2}{2} - T_{wc2}\right) - M_{ref\,2}\left(h_{oc2} - h_{ic2}\right) \tag{2.4}$$

di mana

V_{wc2} : volume sisi udara kondenser kedua (m^3)

UA_3 : perpindahan kalor keseluruhan di kondenser kedua (kW/^oC)

T_2 : suhu udara keluaran kondenser kedua (^oC)

T_{wc2} : suhu dinding kondenser kedua (^oC)

M_{ref2} : aliran massa refrigerant di kondenser kedua (kg/s)

(M_{ref2} = 10% *Mref*)

h_{ic2} : entalpi di input kondenser kedua (kJ/kg)

h_{oe2} : entalpi di output kondenser kedua (kJ/kg)

Pada Gambar 2.6 terlihat bahwa udara yang keluar dari kondenser kedua melewati *fan* sebelum masuk ke kabinet. *Fan* memiliki panas yang dapat menyebabkan suhu udara naik sedikit walaupun tidak signifikan dan dapat diabaikan. Akan tetapi, karena diasumsikan tidak terdapat beban kelembaban yang dihasilkan oleh *fan*, maka kelembaban spesifik udara setelah melewati *fan* (ω_3) dianggap sama dengan kelembaban spesifik keluaran kondenser sekunder (ω_2), sehingga persamaannya menjadi:

$$\omega_3 = \omega_2 = \omega_1 \tag{2.5}$$

Persamaan matematis model udara masuk (*supply air*)

$$T_3 = \frac{C_{pu}\rho_u f T_2 + Q_{spl}}{C_{pu}\rho_u f} \tag{2.6}$$

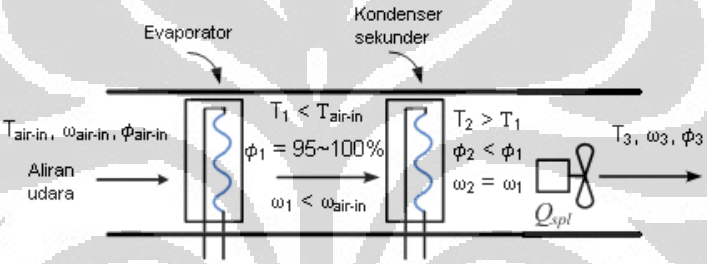
di mana

T_3 : Suhu udara setelah melewati *fan* (^oC)

Q_{spl} : *heat* dari *fan* (kW)

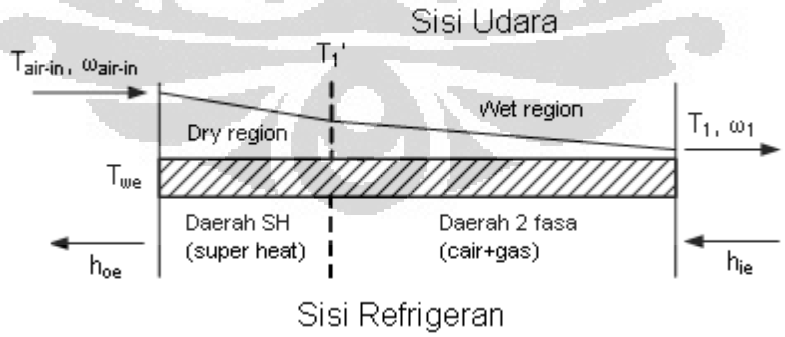
2.2.5 Model Evaporator

Gambar 3.8 menunjukkan skema aliran udara yang melewati evaporator dan kondenser kedua beserta keterangan perubahan suhu dan kelembaban selama melewati kedua komponen tersebut



Gambar 2.5 Skema Aliran Udara di Evaporator dan Kondenser Kedua

Evaporator sendiri memiliki skema aliran udara yang terbagi mejadi *dry region* dan *wet region*.



Gambar 2.6 Skema Aliran Udara dan Refrigeran di Evaporator

Persamaan matematis di *dry region* pada evaporator adalah*:*

$$C_{pu}\rho_u V_1 \frac{dT_1'}{dt} = C_{pu}\rho_u f\left(T_{air-in} - T_1'\right) + UA_1\left(T_{we} - \frac{T_{air-in} + T_1'}{2}\right) \quad (2.7)$$

Persamaan matematis di *wet region* pada evaporator adalah*:*

$$C_{pu}\rho_u V_2 \frac{dT_1}{dt} + \rho_u V_2 h_{fg} \frac{d\omega_1}{dt} = C_{pu}\rho_u f\left(T_1' - T_1\right) + \rho_u f h_{fg}\left(\omega_{air-in} - \omega_1\right) + UA_2\left(T_{we} - \frac{T_1' + T_1}{2}\right)$$

$$\omega_1 = \frac{\left(0{,}0198T_1^{\,2} + 0{,}085T_1 + 4{,}4984\right)}{1000} \quad (2.8)$$

$$\frac{d\omega_1}{dt} = \frac{\left(2 \times 0{,}0198T_1 + 0{,}085\right)\frac{dT_1}{dt}}{1000} \quad (2.9)$$

Persamaan matematis di dinding evaporator sistem tata udara presisi yang digunakan adalah:

$$C_{pw}\rho_w V_{we} \frac{dT_{we}}{dt} = UA_1\left(\frac{T_{air-in} + T_1'}{2} - T_{we}\right) + UA_2\left(\frac{T_1' + T_1}{2} - T_{we}\right) - M_{ref1}\left(h_{oe} - h_{ie}\right) \quad (2.10)$$

di mana

C_{pu} : kalor spesifik udara (kJ/kg^{o}C)

C_{pw} : kalor spesifik dinding evaporator/kondenser (kJ/kg^{o}C)

ρ_u : kerapatan udara (kg/m)

ρ_u : kerapatan dinding evaporator/kondenser (kg/m^3))

V_1 : volume sisi udara evaporator di *dry region* (m^3)

V_2 : volume sisi udara evaporator di *wet region* (m^3)

V_{we} : volume sisi udara evaporator total (m^3)

f : kecepatan aliran udara volumetris (m^3/s)

h_{fg} : kalor laten dari vaporasi udara (kJ/kg)

UA_1 : perpindahan kalor keseluruhan di *dry region* evaporator (kW/^{o}C)

UA_2 : perpindahan kalor keseluruhan di *wet region* evaporator (kW/^{o}C)

$\omega_{air\text{-}in}$: kelembaban spesifik udara di ruang pusat data (kg/kg)

ω_1 : kelembaban spesifik udara keluaran evaporator (kg/kg)

$T_{air\text{-}in}$: suhu udara di ruang pusat data (^{o}C)

T_1' : suhu udara di antara *dry region* dan *wet region* evaporator (^{o}C)

T_1 : suhu udara keluaran evaporator (^{o}C)

T_{we} : suhu dinding evaporator (^{o}C)

M_{ref1} : aliran massa refrigeran di evaporator (kg/s) ($M_{\text{ref1}} = M_{\text{ref}}$)

h_{ie} : entalpi di input evaporator (kJ/kg)

h_{oe} : entalpi di output evaporator (kJ/kg)

2.2.6 Model Kabinet

Persamaan matematis suhu kabinet:

$$C_{pu}\rho_u V_{cab}\frac{dT_{cab}}{dt} = C_{pu}\rho_u f\left(T_3 - T_{cab}\right) + Q_{load} \tag{2.11}$$

Persamaan matematis kelembaban kabinet:

$$\rho_u V_{cab}\frac{d\omega_{cab}}{dt} = \rho_u f\left(\omega_3 - \omega_{cab}\right) + M \tag{2.12}$$

di mana

V_{cab} : volume kabinet (m^3)

T_{cab} : suhu udara kabinet (^{o}C)

ω_{cab} : kelembaban spesifik udara kabinet (kg/kg)

Qload : beban *heat sensible* dari peralatan

M : beban kelembaban di kabinet (kg/s)

Informasi salah satu keluaran PAC yang dibutuhkan adalah kelembaban relatif, maka digunakan persamaan yang mengkonversi kelembaban spesifik (ω) menjadi kelembaban relatif (ϕ)

$$\phi = \frac{\omega P}{(0{,}622 + \omega)P_g} \tag{2.83}$$

Dengan

$$P_g = 0{,}6108 \exp\left(\frac{17{,}27T}{T + 237{,}3}\right) \tag{2.14}$$

Di mana

P : tekanan atmosefer (kPa)

P_g : tekanan uap saturasi (kPa)

T : suhu udara (^{o}C)

2.3 Algoritma Kalman Filter

Algoritma Kalman Filter ini digunakan untuk mengestimasi proses linier dinamis seperti yang terlihat pada (2.15)

$$\boldsymbol{x}_k = A\boldsymbol{x}_{k-1} + B\boldsymbol{u}_{k-1} + \boldsymbol{w}_{k-1} \tag{2.15}$$

dengan persamaan *measurement* yang diperlihatkan pada (2.16)

$$\boldsymbol{z}_k = H\boldsymbol{x}_k + \boldsymbol{v}_k \tag{2.16}$$

dimana wk dan vk adalah variabel acak yang masing-masing merepresentasikan *process noise* dan *measurement noise*. *Noise* ini diasumsikan merupakan *white noise* dengan kovarians masing-masing **Q** dan **R** yang diasumsikan konstan. A adalah matriks yang menghubungkan *state* waktu sebelumnya dengan *state* waktu sekarang. B adalah matriks yang menghubungkan sinyal kendali atau *input* dengan *state* waktu sekarang. H adalah matriks yang menghubungkan *state* waktu sekarang dengan hasil pengukuran.

Untuk menurunkan algoritma Kalman Filter, pertama definisikan $\hat{\boldsymbol{x}}_k^-$ sebagai *a-priori state estimate* dan $\hat{\boldsymbol{x}}_k$ sebagai *a-posteriori state estimate*. *A-priori state estimate* disini berarti estimasi *state* pada *step* ke- k yang diperoleh dengan menggunakan pengetahuan yang ada sampai sebelum *step* ke- k. *A-posteriori state estimate* adalah koreksi dari *a-priori state estimate* setelah data hasil pengukuran diperoleh. Definisikan juga vektor *state-error a-priori* (e_k^-) dan *a-posteriori* (e_k). Persamaan untuk dua parameter ini ditunjukkan oleh (2.17) dan (2.18)

$$e_k^- = \boldsymbol{x}_k - \hat{\boldsymbol{x}}_k^- \tag{2.17}$$

$$e_k = \boldsymbol{x}_k - \hat{\boldsymbol{x}}_k \tag{2.18}$$

dengan kovarians *error* masing-masing P_k^- dan P_k

Tujuan dari algoritma Kalman Filter adalah mengupdate estimasi *state* yang tidak diketahui dengan menggunakan informasi yang terkandung pada hasil pengukuran, $\mathbf{z}_k$ yang didapat setiap time *step* k. Disini, *estimator* yang diinginkan berjenis *linear estimator*. Dengan demikian, *a-posteriori estimate* dapat direpresentasikan sebagai kombinasi linier antara *a-priori estimate* dan hasil pengukuran. Persamaan yang terbentuk kemudian adalah (2.19)

$$\hat{\boldsymbol{x}}_k = \boldsymbol{K}_k^{(1)}\hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k \boldsymbol{z}_k \tag{2.19}$$

dengan $\boldsymbol{K}_k^{(1)}$ dan $\boldsymbol{K}_k$ adalah *gain* yang nilainya akan dicari.

Untuk mencari nilai *gain* $\boldsymbol{K}_k^{(1)}$ dan $\boldsymbol{K}_k$, digunakanlah prinsip ortogonalitas. Prinsip ini menyatakan bahwa pada sebuah *estimator*, nilai estimasi $\hat{\boldsymbol{x}}_k$ akan memiliki nilai *mean square error* minimal jika dan hanya jika (2.20) terpenuhi.

$$E[\boldsymbol{e}_k\, \boldsymbol{y}_i^T] = \boldsymbol{0} \quad \text{untuk i = 1,2,3,…k-1} \tag{2.20}$$

Dengan menggunakan (2.16), (2.18), (2.19) dan (2.20) didapat (2.21)

$$E\left[(\boldsymbol{x}_k - \boldsymbol{K}_k^{(1)}\hat{\boldsymbol{x}}_k^- - \boldsymbol{K}_k H\boldsymbol{x}_k - \boldsymbol{K}_k \boldsymbol{w}_{k-1})\boldsymbol{z}_i^T\right] = \boldsymbol{0} \tag{2.21}$$

Karena $\boldsymbol{v}_k$ dan $\boldsymbol{w}_{k-1}$ *uncorrelated*, maka

$$E[\boldsymbol{w}_k\, \boldsymbol{y}_i^T] = \boldsymbol{0}$$

Sehingga (2.21) dapat ditulis kembali menjadi (2.22)

$$E\left[(\boldsymbol{I} - \boldsymbol{K}_k H - \boldsymbol{K}_k^{(1)})\boldsymbol{x}_k \boldsymbol{y}_i^T + \boldsymbol{K}_k^{(1)}(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k^-)\boldsymbol{z}_i^T\right] = \boldsymbol{0} \tag{2.22}$$

Dari prinsip ortogonalitas, dapat diperoleh hubungan

$$E[(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k^-)\boldsymbol{z}_i^T] = \boldsymbol{0} \tag{2.23}$$

Dengan demikian, (2.18) dapat disederhanakan menjadi bentuk (2.24)

$$(\boldsymbol{I} - \boldsymbol{K}_k H - \boldsymbol{K}_k^{(1)})E\left[\boldsymbol{x}_k \boldsymbol{z}_i^T\right] = \boldsymbol{0} \tag{2.24}$$

Untuk sembarang nilai $\boldsymbol{x}_k$ dan $\boldsymbol{y}_i^T$, (2.24) hanya dapat terpenuhi jika berlaku hubungan

$$\boldsymbol{I} - \boldsymbol{K}_k H - \boldsymbol{K}_k^{(1)} = \boldsymbol{0}$$

atau dengan kata lain

$$\boldsymbol{K}_k^{(1)} = \boldsymbol{I} - \boldsymbol{K}_k H \tag{2.25}$$

Substitusi (2.25) ke (2.19) akan menghasilkan representasi lain dari *a-posteriori estimate* seperti tergambar pada (2.26)

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k(\boldsymbol{z}_k - H\hat{\boldsymbol{x}}_k^-) \tag{2.26}$$

Persamaan (2.26) adalah inti dari proses estimasi *state* pada algoritma Kalman Filter. Parameter $\boldsymbol{K}_k$ memiliki peran sangat penting untuk mengendalikan proses koreksi estimasi *state* berdasarkan data hasil pengukuran. Parameter ini biasa disebut sebagai Kalman *Gain*.

Persamaan eksplisit untuk menghasilkan $\boldsymbol{K}_k$ akan didiskusikan lebih lanjut. Dari prinsip ortogonalitas dapat diperoleh (2.27)

$$E[(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k^-)\hat{\boldsymbol{z}}_i^T] = \boldsymbol{0} \tag{2.27}$$

dengan $\hat{\boldsymbol{z}}_i^T$ adalah estimasi dari data hasil pengukuran dan memiliki persamaan seperti terlihat pada (2.28)

$$\hat{\boldsymbol{z}}_k = H\hat{\boldsymbol{x}}_k^- \tag{2.28}$$

Definisikan residu atau proses inovasi sebagai (2.28)

$$\tilde{\boldsymbol{z}}_k = \boldsymbol{z}_k - \hat{\boldsymbol{z}}_k \tag{2.29}$$

Dengan mensubstitusikan (2.28) dan (2.26) ke (2.29), maka didapat (2.30) yang merupakan representasi lain dari (2.29)

$$\tilde{\boldsymbol{z}}_k = \boldsymbol{H}\boldsymbol{e}_k^- + \boldsymbol{v}_k \tag{2.30}$$

Mengurangkan (2.23) dengan (2.27) dan menggunakan definisi (2.29) menghasilkan (2.31)

$$E[(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k^-)\tilde{\boldsymbol{z}}_i^T] = \boldsymbol{0} \tag{2.31}$$

Dengan menggunakan persamaan *measurement* pada (2.16) dan persamaan *state estimate update* pada (2.26), maka vektor *state-error* e_k dapat ditulis ulang menjadi (2.32)

$$\boldsymbol{e}_k = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{e}_k^- - \boldsymbol{K}_k\boldsymbol{v}_k \tag{2.32}$$

Mensubstitusi (2.32) dan (2.30) ke (2.31) menghasilkan (2.33)

$$E[\{(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{e}_k^- - \boldsymbol{K}_k\boldsymbol{v}_k\}(\boldsymbol{H}\boldsymbol{e}_k^- + \boldsymbol{v}_k)] = \boldsymbol{0} \tag{2.33}$$

Karena *measurement noise* $\boldsymbol{v}_k$ independen terhadap *state* dan juga terhadap *error* $\boldsymbol{e}_k^-$, maka (2.19) dapat direduksi menjadi (2.20)

$$(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})E[\boldsymbol{e}_k^-\boldsymbol{e}_k^{-T}]\boldsymbol{H}^T - \boldsymbol{K}_k E[\boldsymbol{v}_k\,\boldsymbol{v}_k{}^T] = \boldsymbol{0} \tag{2.34}$$

Dengan mendefinisikan $E[\boldsymbol{e}_k^-\boldsymbol{e}_k^{-T}]$ sebagai $\boldsymbol{P}_k^-$ dan memperhatikan sifat kovarians, (2.34) dapat direduksi menjadi (2.35)

$$(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{P}_k^-\boldsymbol{H}^T - \boldsymbol{K}_k\boldsymbol{R} = \boldsymbol{0} \tag{2.35}$$

Menyelesaikan (2.35) untuk $\boldsymbol{K}_k$ akan menghasilkan persamaan eksplisit untuk $\boldsymbol{K}_k$ pada (2.36)

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}^T [\boldsymbol{H}\boldsymbol{P}_k^- \boldsymbol{H}^T + \boldsymbol{R}]^{-1} \quad (2.36)$$

Tahap terakhir untuk mendapatkan algoritma Kalman Filter adalah menemukan efek waktu terhadap matriks *error* kovarians. Proses ini disebut *error covariance propagation* dan memiliki dua langkah, yaitu

a. Menemukan perhitungan $\boldsymbol{P}_k$ jika diketahui $\boldsymbol{P}_k^-$

b. Menemukan perhitungan$\boldsymbol{P}_k^-$ jika diketahui $\boldsymbol{P}_{k-1}$

Langkah pertama dilakukan dengan mendefinisikan $\boldsymbol{P}_k$ dengan persamaan (2.37)

$$\boldsymbol{P}_k = E[\boldsymbol{e}_k \boldsymbol{e}_k{}^T] \quad (2.37)$$

Dengan menggunakan persamaan *state error* vector pada (2.32) dan memperhatikan bahwa *process noise* $\boldsymbol{v}_k$ independen terhadap *a-priori estimate error* $\boldsymbol{e}_k^-$, maka dari (2.37) didapat (2.38)

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}) \boldsymbol{P}_k^- (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H})^T - \boldsymbol{K}_k \boldsymbol{R} \boldsymbol{K}_k{}^T \quad (2.38)$$

Menggunakan hubungan pada (2.36), (2.38) dapat disederhanakan menjadi (2.39)

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}) \boldsymbol{P}_k^- \quad (2.39)$$

Persamaan (2.39) merupakan persamaan untuk menghitung $\boldsymbol{P}_k$ yang akan digunakan dalam langkah-langkah algoritma Kalman Filter.

Langkah kedua dilakukan dengan mendefinisikan *a-priori estimate* sebagai fungsi *a-posteriori estimate* pada waktu sebelumnya. Persamaan yang digunakan untuk merepresentasikan hal tersebut dijelaskan pada (2.40)

$$\widehat{\boldsymbol{x}}_k^- = \boldsymbol{A} \widehat{\boldsymbol{x}}_{k-1} \quad (2.40)$$

Dengan menggunakan (2.33) dan (2.40), dapat dibuat definisi baru dari *state error vector* yang dijelaskan pada (2.41)

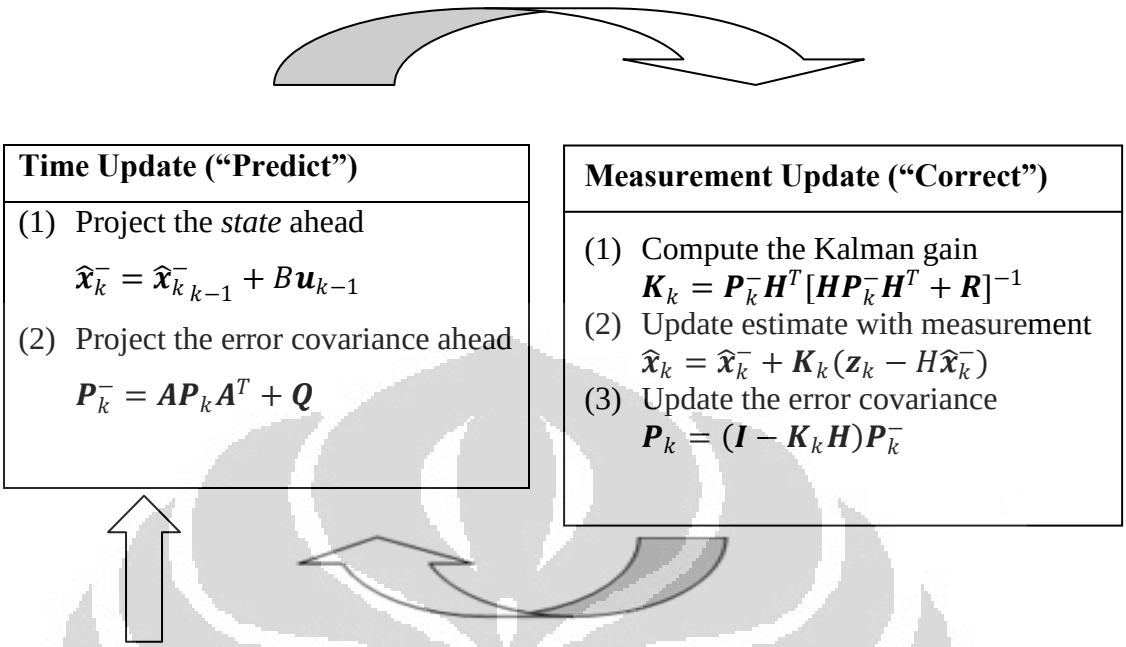
$$\boldsymbol{e}_k^- = \boldsymbol{A}\boldsymbol{e}_k^- + \boldsymbol{w}_{k-1} \quad (2.41)$$

Berikutnya, dengan menggunakan (2.41) pada definisi $\boldsymbol{P}_k^-$, didapatlah (2.42)

$$\boldsymbol{P}_k^- = \boldsymbol{A}\boldsymbol{P}_k \boldsymbol{A}^T + \boldsymbol{Q} \quad (2.42)$$

Setelah persamaan-persamaan yang dibutuhkan telah didapatkan, maka langkah-langkah dalam algoritma Kalman Filter dapat dirumuskan.

Langkah-langkah algoritma ini dijelaskan pada gambar 2.7.



Gambar 2.7 Representasi Visual Algoritma Kalman Filter

Pada algoritma Kalman Filter, terutama pada bagian perhitungan Kalman *Gain*, terdapat karakteristik yang perlu diperhatikan. Untuk nilai $\boldsymbol{R}$ yang semakin kecil mendekati nol, maka $\boldsymbol{K}_k$ akan memboboti residu dengan lebih berat. Dapat dikatakan, untuk nilai *measurement error covariance* yang semakin kecil, maka data hasil pengukuran akan semakin dipercaya. Sebaliknya, ketika nilai *a-priori estimate error covariance* semakin kecil, bobot residu akan semakin kecil. Dapat dikatakan, untuk nilai *estimate error covariance* yang semakin kecil, *state* hasil estimasi akan semakin dipercaya.

2.3.1 Extended Kalman Filter

Metode *Extended Kalman Filter* digunakan untuk mengestimasi sistem dengan persamaan *difference* yang diperlihatkan pada (2.43)

$$\boldsymbol{x}_k = f(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{w}_{k-1}) \tag{2.43}$$

dengan persamaan *measurement* yang diperlihatkan pada (2.44)

$$\boldsymbol{z}_k = h(\boldsymbol{x}_k, \boldsymbol{v}_k) \tag{2.44}$$

dimana $f(.)$ adalah persamaan nonlinier yang menghubungkan *state* waktu sebelumnya, *process noise* dan *input* dengan *state* waktu sekarang. Pada persamaan *measurement*, $h(.)$ adalah persamaan nonlinier yang menghubungkan *state* sekarang dan *measurement noise* dengan hasil pengukuran. Estimasi untuk mendapatkan *state* $\boldsymbol{x}_k$, dilakukan dengan menggunakan (2.45) dan (2.46)

$$\hat{\boldsymbol{x}}_k^- = f(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{w}_{k-1}) \tag{2.45}$$

$$\boldsymbol{z}_{\boldsymbol{k}}^- = h(\hat{\boldsymbol{x}}_k^-, \boldsymbol{v}_k) \tag{2.46}$$

Untuk persamaan eksplisit estimasi *state* pada (2.43), pertama persamaan (2.45) dan (2.46) harus dilinierisasi. Linierisasi ini dilakukan menggunakan deret Taylor di sekitar titik kerja yaitu $\boldsymbol{w}_k = \boldsymbol{0}$ dan $\boldsymbol{v}_k = \boldsymbol{0}$. Asumsi ini digunakan karena nilai *noise* yang terjadi pada proses dan pada pengukuran tidak dapat dihitung. Estimasi dilakukan dengan menganggap *noise-noise* tersebut bernilai nol. Dengan demikian, persamaan sistem berubah menjadi (2.47) dan (2.48)

$$\boldsymbol{x}_k \approx f(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_{k-1}, 0) + \boldsymbol{F}(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_{k-1}, 0) + \boldsymbol{W}\boldsymbol{w}_k \tag{2.47}$$

$$\boldsymbol{z}_k \approx h(\hat{\boldsymbol{x}}_k^-, 0) + \boldsymbol{H}(\hat{\boldsymbol{x}}_k^-, 0) + \boldsymbol{V}\boldsymbol{v}_k \tag{2.48}$$

dimana

$$\boldsymbol{F}_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_{k-1}, 0)$$

$$\boldsymbol{W}_{[i,j]} = \frac{\partial f_{[i]}}{\partial w_{[j]}}(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_{k-1}, 0)$$

$$\boldsymbol{H}_{[i,j]} = \frac{\partial H_{[i]}}{\partial w_{[j]}}(\hat{\boldsymbol{x}}_k^-, 0)$$

$$\boldsymbol{V}_{[i,j]} = \frac{\partial H_{[i]}}{\partial v_{[j]}}(\hat{\boldsymbol{x}}_k^-, 0)$$

Berikutnya dengan mendefinisikan *measurement residual* sebagai (2.49) dan memperhatikan persamaan *a-priori state error*, didapat persamaan *error* proess seperti terlihat pada (2.50) dan (2.51)

$$\hat{e}_{z_k}^- = \boldsymbol{z}_k - \hat{\boldsymbol{z}}_k^- \tag{2.49}$$

$$\hat{e}_{x_k}^- \approx A(\boldsymbol{x}_{k-1} - \hat{\boldsymbol{x}}_{k-1}^-) + \varepsilon_k \tag{2.50}$$

$$e_{z_k}^- \approx \boldsymbol{H}\hat{e}_{x_k}^- + \eta_k \tag{2.51}$$

Pada (2.50) dan (2.51), terdapat dua variabel baru, yaitu ε_k dan η_k. Variabel-variabel ini adalah variabel acak baru yang memiliki *mean* nol dan

matriks kovarians masing-masing WQW^T dan VRV^T. Variabel-variabel acak pada (2.49)-(2.51) memiliki karakteristik :

$$p(\hat{e}_{x_k}^-) \sim N(0, E[\hat{e}_{x_k}^- \hat{e}_{x_k}^{-\,T}])$$
$$p(\varepsilon_k) \sim N(0, E[\boldsymbol{W Q_k W^T}])$$
$$p(\eta_k) \sim N(0, E[\boldsymbol{V R_k V^T}])$$

Selanjutnya, akan dicari estimasi dari $\hat{e}_{x_k}^-$ dengan menggunakan Kalman Filter hipotetis. Hasil estimasi ini dinamakan $\hat{\boldsymbol{e}}_k$ dan akan digunakan untuk mendapatkan *a-posteriori state estimate* berdasarkan (2.52)

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \hat{\boldsymbol{e}}_k \tag{2.52}$$

Dengan memperhatikan karakteristik $\hat{e}_{x_k}^-$, ε_k, dan η_k, men-set nilai prediksi $\hat{\boldsymbol{e}}_k$ menjadi nol, dan mempertimbangkan data $\hat{e}_{z_k}^-$ didapatlah persamaan Kalman Filter hipotetis untuk memperoleh nilai $\hat{\boldsymbol{e}}_k$ pada (2.53)

$$\hat{\boldsymbol{e}}_k = \boldsymbol{K}_k \hat{e}_{z_k}^- \tag{2.53}$$

Mensubstitusikan (2.53) ke (2.52) dan mempertimbangkan persamaan *measurement residual*, didapat (2.54)

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k(\boldsymbol{z}_k - \hat{\boldsymbol{z}}_k^-) \tag{2.54}$$

Persamaan (2.54) adalah persamaan *state estimate update* untuk sistem nonlinier.

Persamaan-persamaan yang digunakan dalam algoritma *Extended Kalman Filter* kemudian adalah sesuai dengan yang tertera pada tabel 2.1. Persamaan-persamaan ini diperoleh dari persamaan Kalman Filter dengan beberapa penyesuaian.

Tabel 2.2 Persamaan-Persamaan *Extended Kalman Filter*

Time Update ("Prediction") Equations $\hat{\boldsymbol{x}}_k^- = f(\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{w}_{k-1})$ $\boldsymbol{P}_k^- = \boldsymbol{F P}_k \boldsymbol{F}^T + \boldsymbol{WQW^T}$
Measurement Update ("Correction") Equations $\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}^T [\boldsymbol{H P}_k^- \boldsymbol{H}^T + \boldsymbol{VRV^T}]^{-1}$ $\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k(\boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_k^-, 0))$ $\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}) \boldsymbol{P}_k^-$

2.3.2 Estimasi Kovarian *Noise* pada Sistem dengan Bias [16]

Berdasarkan sistem stokastik *controllable/unobservable* linier diskrit,

$$\begin{bmatrix} \boldsymbol{x_0(k+1)} \\ \boldsymbol{x_u(k+1)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{F} & \boldsymbol{M} \\ 0 & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x_0(k)} \\ \boldsymbol{x_u(k)} \end{bmatrix} + \begin{bmatrix} \boldsymbol{G} \\ 0 \end{bmatrix} \boldsymbol{w(k)}$$

$$\boldsymbol{y(k)} = [\boldsymbol{H} \quad 0] \begin{bmatrix} \boldsymbol{x_0(k)} \\ \boldsymbol{x_u(k)} \end{bmatrix} + \boldsymbol{v(k)} \tag{2.55}$$

Dimana $\boldsymbol{x_o} \epsilon \boldsymbol{R}^n$ adalah *state* vektor dinamik, $\boldsymbol{x_u} \epsilon \boldsymbol{R}^m$ merupakan vektor yang tersusun dari *state* bias konstan dan $\boldsymbol{y(k)} \epsilon \boldsymbol{R}^r$ merupakan vektor pengukuran. $\boldsymbol{F} \epsilon \boldsymbol{R}^{n X n}$ diketahui, matriks transisi konstan dari $\boldsymbol{x_0(k)}$ sampai $\boldsymbol{x_0(k+1)}$. $\boldsymbol{M} \epsilon \boldsymbol{R}^{n X m}$ tidak diketahui namun merupakan matriks transisi konstan dari $\boldsymbol{x_u(k)}$ sampai $\boldsymbol{x_u(k+1)}$. $\boldsymbol{G} \epsilon \boldsymbol{R}^{n X q}$ diketahui, merupakan matriks input konstan untuk $\boldsymbol{w(k)}$, dan $\boldsymbol{H} \epsilon \boldsymbol{R}^{r X n}$ diketahui merupakan matrik output konstan untuk $\boldsymbol{x_o}$.

Misalkan noise proses $\boldsymbol{w(k)} \epsilon \boldsymbol{R}^q$ dan noise pengukuran $\boldsymbol{v(k)} \epsilon \boldsymbol{R}^r$ memenuhi asumsi-asumsi berikut ini:

Asumsi pertama, $\{\boldsymbol{w(k)}\}$ dan $\{\boldsymbol{v(k)}\}$ merupakan *individually stasionary*, *zero-mean, white processes* dengan kovarian

$$E[\boldsymbol{w(i)w(j)}^T] = \boldsymbol{Q}\delta_{ij}$$

$$E[\boldsymbol{v(i)v(j)}^T] = \boldsymbol{R}\delta_{ij} \quad \nabla\, i,j$$

dimana δ_{ij} menunjukkan *Kroneker delta function.* Keduanya $\boldsymbol{Q} \epsilon \boldsymbol{R}^{q X q}$ dan $\boldsymbol{R} \epsilon \boldsymbol{R}^{r X r}$ merupakan matriks definit positif simetris.

Asumsi kedua, $\{\boldsymbol{w(k)}\}$ dan $\{\boldsymbol{v(k)}\}$ merupakan *mutually uncorelated process*, sebagai contoh:

$$E[\boldsymbol{w(i)v(j)}^T] = E[\boldsymbol{w(i)}]E[\boldsymbol{w(j)}^T] \quad \nabla\, i,j$$

Asumsi ketiga, orde keempat dari $\{\boldsymbol{w(k)}\}$ dan $\{\boldsymbol{v(k)}\}$ terbatas.

Dalam mendesain Filter Kalman, noise proses dan noise pengukuran diasumsikan sebagai *gaussian* proses. Karena sebuah *zero-mean white gaussian prosess* memiliki saat orde empat yang terbatas, asumsi ketiga kurang membatasi daripada asumsi Gaussian. Sebuah asumsi tambahan diperlukan untuk sistem (2.55). Dengan kata lain, sistem bebas bias, yang terdiri hanya dari $\boldsymbol{x_0}$ sebagai *state* vektornya jika tidak ada *state* bias tersedia di (2.55), diasumsikan menjadi *controllable* dan *observable*, sebagai contoh,

$$rank[\boldsymbol{G}, \boldsymbol{FG}, \ldots, \boldsymbol{F}^{n-1}\boldsymbol{G}] = n$$

$$rank[\boldsymbol{H}, (\boldsymbol{HF})^T, \dots, (\boldsymbol{HF}^{n-1})^T] = n$$

Kovarians noise **Q** dan **R** merupakan matriks konstan yang tidak diketahui. Untuk menyelesaikan (2.55) untuk **Q** dan **R** dimana *state* matriks transisi secara parsial tidak diketahui. Karena **M** tidak diketahui, sebuah Filter Kalman digagas dari (2.55) mungkin divergen selama sebuah model error dari **M**. Oleh karena itu, dalam pengestimasian **Q** dan **R**, sebuah rangkaian yang tidak berhubungan dengan matriks **M** harus diciptakan.

Perlakuan *state* vektor bias $\boldsymbol{x_u}$ dalam (2.55) sebagai sebuah vektor input $\boldsymbol{u}$, sebuah pengurangan orde ke-n sistem dapat ditulis sebagai:

$$\boldsymbol{x_0}(\boldsymbol{k}+\boldsymbol{1}) = \boldsymbol{F}\,\boldsymbol{x_0}(\boldsymbol{k}) + \boldsymbol{Mu} + \boldsymbol{Gw}(\boldsymbol{k}) \tag{2.56}$$
$$\boldsymbol{y}(\boldsymbol{k}) = \boldsymbol{Hx_0}(\boldsymbol{k}) + \boldsymbol{v}(\boldsymbol{k})$$

Jika α vektor pengukuran tersedia dari waktu $k-\alpha+1$ sampai k, kemudian sebuah alternatif merepresentasikan dari (2.56) memungkinkan dengan jalan $\boldsymbol{Y}(\boldsymbol{k})$ dipekerjakan sebagai sebuah kombinasi linier dari *state observable*, input dan output konstan tidak diketahui dan noise.

$$\boldsymbol{Y}(\boldsymbol{k}) = \boldsymbol{\Gamma x_0}(\boldsymbol{k}+\boldsymbol{\alpha}+\boldsymbol{1}) + \underline{\mathrm{N}}\boldsymbol{U} + \boldsymbol{BW}(\boldsymbol{k}) + \boldsymbol{V}(\boldsymbol{k}) \tag{2.57}$$

$$\boldsymbol{Y}(\boldsymbol{k}) \equiv \begin{bmatrix} \boldsymbol{y}(\boldsymbol{k}-\boldsymbol{\alpha}+\boldsymbol{1}) \\ \boldsymbol{y}(\boldsymbol{k}-\boldsymbol{\alpha}+\boldsymbol{2}) \\ \cdot \\ \cdot \\ \cdot \\ \boldsymbol{y}(\boldsymbol{k}) \end{bmatrix} \tag{2.58}$$

dimana α adalah sebuah user-specified integer yang disebut sliding window size. $\boldsymbol{Y}(\boldsymbol{k})$ adalah sebuah $\alpha r\ X\ 1$ blok vektor pengukuran. Tiga vektor dengan cara yang sama didefinisikan sebagai (2.584). $\boldsymbol{W}(\boldsymbol{k})$ adalah sebuah $\alpha q\ X\ 1$ blok vektor noise proses dan $\boldsymbol{V}(\boldsymbol{k})$ adalah $\alpha r\ X\ 1$ blok vektor noise pengukuran. $\boldsymbol{U}$ adalah $\alpha m\ X\ 1$ vektor konstan, yang terdiri dari *state* biar. **Γ** adalah sebuah matriks $\alpha n\ X\ n$ sebagai berikut:

$$\boldsymbol{\Gamma} = \begin{bmatrix} \boldsymbol{H} \\ \boldsymbol{HF} \\ \cdot \\ \cdot \\ \cdot \\ \boldsymbol{HF}^{\alpha-1} \end{bmatrix} \tag{2.59}$$

Hal ini sama untuk matriks observability dari (2.56) dalam kasus $\alpha = n$. Karena (2.56) observable, **Γ** memilki rank penuh n dalam kasus $\alpha \geq n$. $\boldsymbol{B} \epsilon \boldsymbol{R}^{\alpha r X \alpha q}$ yang merupakan sebuah matriks Toeplitz diekspresikan sebagai,

$$\boldsymbol{B} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ \boldsymbol{HG} & 0 & 0 & 0 & 0 \\ \boldsymbol{HFG} & \boldsymbol{HG} & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ \boldsymbol{HF^{\alpha-2}G} & \boldsymbol{HF^{\alpha-3}G} & \boldsymbol{HF^{\alpha-4}G} & \dots & 0 \end{bmatrix}$$

Dan $\underline{N} \epsilon \boldsymbol{R}^{\alpha r X \alpha m}$ juga sebuah matriks Toeplitz dengan cara yang sama terdiri dari **H**, **F**, dan **M**.

Sekarang berdasarkan pendahuluan dari sebuah matriks proyeksi $\boldsymbol{\Pi}^T$ merupakan sebuah matriks yang berubah-ubah yang terdiri dari beberapa vektor dalam null space dari $\boldsymbol{\Gamma}^T$. Sebagai contoh, menggunakan singular value decomposition.

Dari $\boldsymbol{\Gamma}$ dengan rank penuh, $\boldsymbol{\Pi}$ dapat dibangun sebagai sebuah matriks berdimensi $(\alpha r - n)\, X\, \alpha r$ dengan rank $(\alpha r - n)$. Dengan mengalikan $\boldsymbol{\Pi}$ oleh persamaan diferensial antara (2.57) dan blok vektor pengukuran $\boldsymbol{Y}(\boldsymbol{k} - \boldsymbol{l})$ pada waktu $k - l$, sebuah variabel baru $Z(k, l)$ dinotasikan $\boldsymbol{\Pi}\{\boldsymbol{Y}(\boldsymbol{k}) - \boldsymbol{Y}(\boldsymbol{k} - \boldsymbol{l})\}$ diekspresikan sebagai

$$\boldsymbol{Z}(\boldsymbol{k}, \boldsymbol{l}) = \boldsymbol{\Pi}[\boldsymbol{B}\{(\boldsymbol{k}) - \boldsymbol{Y}(\boldsymbol{k} - \boldsymbol{l})\} + \{\boldsymbol{V}(\boldsymbol{k}) - \boldsymbol{V}(\boldsymbol{k} - \boldsymbol{l})\}] \qquad (2.60)$$
$$\nabla\, l > 0$$

Sebagai catatan bahwa (2.60) diekspresikan dalam noise proses dan pengukuran hanya tanpa $\boldsymbol{x_0}(\boldsymbol{k} - \boldsymbol{\alpha} + \boldsymbol{1})$, $\boldsymbol{x_0}(\boldsymbol{k} - \boldsymbol{l} - \boldsymbol{\alpha} + \boldsymbol{1})$, dan $\boldsymbol{U}$. Dan juga bahwa (2.60) bebas dari M. Dari manipulasi aljabar dan asumsi pertama dan kedua, hal ini dapat dideterminasikan bahwa mean dan kovarians dari $Z(k, l)$ adalah

$$E[\boldsymbol{Z}(\boldsymbol{k}, \boldsymbol{l})] = 0 \qquad (2.61)$$

$$E[\boldsymbol{Z}(\boldsymbol{k}, \boldsymbol{l}), (\boldsymbol{k}, \boldsymbol{l})^T] = \boldsymbol{\Pi}[\boldsymbol{B}\boldsymbol{Q}(\boldsymbol{k}, \boldsymbol{l})\boldsymbol{B}^T + \boldsymbol{R}(\boldsymbol{k}, \boldsymbol{l})]\boldsymbol{\Pi}^T \qquad (2.62)$$

$\boldsymbol{Q}(\boldsymbol{k}, \boldsymbol{l})$ dan $\boldsymbol{R}(\boldsymbol{k}, \boldsymbol{l})$ menotasikan kovarian dari $\boldsymbol{W}(\boldsymbol{k}) - \boldsymbol{W}(\boldsymbol{k} - \boldsymbol{l})$ dan $\boldsymbol{V}(\boldsymbol{k}) - \boldsymbol{V}(\boldsymbol{k} - \boldsymbol{l})$. Yang dihitung sebagai

$$\boldsymbol{Q}(\boldsymbol{k}, \boldsymbol{l}) = \boldsymbol{Q}(\boldsymbol{l}) = \begin{matrix} 2\boldsymbol{Q_b} - \boldsymbol{Q_b}\boldsymbol{S_Q^l} - (\boldsymbol{Q_b}\boldsymbol{S_Q^l})^T, & l \leq \alpha - 1 \\ \boldsymbol{2Q_b}, & > \alpha - 1 \end{matrix}$$

$$\boldsymbol{R}(\boldsymbol{k}, \boldsymbol{l}) = \boldsymbol{R}(\boldsymbol{l}) = \begin{matrix} 2\boldsymbol{R_b} - \boldsymbol{R_b}\boldsymbol{S_R^l} - (\boldsymbol{R_b}\boldsymbol{S_R^l})^T, & l \leq \alpha - 1 \\ 2\boldsymbol{R_b}, & > \alpha - 1 \end{matrix} \qquad (2.63)$$

dengan

$$\boldsymbol{Q_b} = block\ diag\{Q, \dots, Q\} \quad \alpha\ deret$$

$$\boldsymbol{R_b} = block\ diag\{R, \dots, R\} \quad \alpha\ deret$$

$$\boldsymbol{S_q} = \begin{bmatrix} 0_{qXq} & \boldsymbol{I}_{qXq} & 0_{qXq} & \cdots & 0_{qXq} \\ 0_{qXq} & 0_{qXq} & \boldsymbol{I}_{qXq} & \cdots & 0_{qXq} \\ 0_{qXq} & 0_{qXq} & 0_{qXq} & \cdots & 0_{qXq} \\ \vdots & \vdots & \vdots & \ddots & \boldsymbol{I}_{qXq} \\ 0_{qXq} & 0_{qXq} & 0_{qXq} & 0_{qXq} & 0_{qXq} \end{bmatrix} \tag{2.64}$$

Matriks $\alpha r\ X\ \alpha r$ dari matriks $\boldsymbol{S_r}$ sama halnya didefinisikan oleh substitusi 0_{rXr} dan I_{rXr} untuk 0_{qXq} dan $\boldsymbol{I}_{qXq}$. Kovarians dari $\boldsymbol{Z(k,l)}$ tidak tergantung di waktu k, tetapi perubahan waktu l. $\{\boldsymbol{Z(k,l)}\}$ adalah deret baru untuk mengestimasi **Q** dan **R**.

Menganggap estimasi kovarian noise **Q** dan **R** dari (2.68). Karena sebuah manipulasi dari sebuah vektor adalah sedikit lebih kompleks daripada matriks tersebut, hal ini tepat untuk memperkenalkan sebuah vektor yang terdiri dari kovarians noise yang bisa diestimasi. **Q** dan **R** keduanya dapat ditulis sebagai fungsi linier dari p komponen dari sebuah vektor θ

$$\boldsymbol{Q} = \sum_{i=l}^{p} \boldsymbol{Q_i \theta_i}, \qquad \boldsymbol{R} = \sum_{i=l}^{p} \boldsymbol{R_i \theta_i} \tag{2.65}$$

Dimana $\boldsymbol{Q_i}$ dan $\boldsymbol{R_i}$ merupakan *user-specified matrices*, $\boldsymbol{\theta_i}$ merupakan element ke i dari vektor $\boldsymbol{\theta}$ yang tidak diketahui, dan p merupakan jumlah total dari parametet yang diestimasi dalam matriks kovarians noise. Hal ini mengikuti dari persamaan (2.65) bahwa kovarians dari $\boldsymbol{Z(k,l)}$ dapat juga diekspresikan sebagai sebuah kombinasi linier dari ketidaktahuan parameter θ_i seperti

$$\boldsymbol{E[Z(k,l), Z(k,l)^T]} = \sum_{i=l}^{p} \boldsymbol{D_i(l)\theta_i} \tag{2.66}$$

$$\boldsymbol{D_i(l)} = \Pi[\boldsymbol{BQ_i(l)B^T + R_i(l)}]\Pi^T \tag{2.67}$$

Dimana $\boldsymbol{Q_i(l)}$ dan $\boldsymbol{R_i(l)}$ dihitung dari (2.63) setelah menyisipkan $\boldsymbol{Q_i}$, $\boldsymbol{R_i}$ malahan dari $\boldsymbol{Q}$ dan $\boldsymbol{R}$ dalam (2.64). Untuk mengidentifikasi vektor $\boldsymbol{\theta}$ dengan satu deretan sampel data. Jika $\boldsymbol{z(k)}$ tetap, proses *moving average* (MA) dijalankan

oleh sebuah *zero-mean white process* dengan *finite variance* dan *finite fourth-order moment.*

Kemudian dengan persamaan

$$\lim_{N \to \infty} \frac{1}{N} \sum_{k=1}^{N} \boldsymbol{z(k)z(k-i)} = E[\boldsymbol{z(k)z(k-i)}]$$

Dengan probabilitas satu dan dalam akar rata-rata. Menggunakan kebenaran ini, dapat ditunjukkan bahwa

$$\begin{aligned} \lim_{N \to \infty} \frac{1}{N} \sum_{k=1}^{N} \boldsymbol{W(k)W(k-i)^T} &= \boldsymbol{Q_b S_Q^l} \\ \lim_{N \to \infty} \frac{1}{N} \sum_{k=1}^{N} \boldsymbol{V(k)V(k-i)^T} &= \boldsymbol{R_b S_R^l} \\ \lim_{N \to \infty} \frac{1}{N} \sum_{k=1}^{N} \boldsymbol{W(k)V(k-i)^T} &= 0 \quad \forall \quad i \geq 0. \end{aligned} \tag{2.68}$$

Setelah mengalikan $\boldsymbol{Z(k,l)}$ di (2.60) dengan $\boldsymbol{Z(k,l)^T}$, kemudian mengambil rata-rata waktu dengan data takhingga dan menjalankan (2.68), waktu rata-rata dari $\boldsymbol{Z(k,l)Z(k,l)^T}$ ditulis sebagai

$$\lim_{N \to \infty} \frac{1}{N} \sum_{k=1}^{N} \boldsymbol{Z(k,l)Z(k,l)^T} = \boldsymbol{\Pi[BQ(k,l)B^T + R(k,l)]\Pi^T} \tag{2.69}$$

Dari (2.62) dan (2.69), hal tersebut mengikuti $\{\boldsymbol{Z(k,l)Z(k,l)^T}\}$ merupakan *mean-ergodic process* dan dapat dimodelkan dengan

$$\begin{aligned} \boldsymbol{Z(k,l)Z(k,l)^T} &= \boldsymbol{\Pi[BQ(k,l)B^T + R(k,l)]\Pi^T} + \boldsymbol{e(k,l)} \\ &= \sum_{i=l}^{p} \boldsymbol{D_i(l)\theta_i} + \boldsymbol{e(k,l)} \end{aligned} \tag{2.70}$$

Dimana $\boldsymbol{e(k,l)}$ merupakan sebuah matriks error yang memuaskan

$$\lim_{N \to \infty} \frac{1}{N} \sum_{k=1}^{N} \boldsymbol{e(k,l)} = 0 \tag{2.71}$$

2.3.2.1 Algoritma *Least Square*

Pendeskripsian sebuah metode untuk mengidentifikasi θ dari (2.70) berdasarkan algoritma *least square* konvensional. Untuk mengkonversi (2.70) menjadi sebuah bentuk vektor, sebuah tumpukkan vektor vec(.) pertama kali didefinisikan untuk elemen di bagian segitiga atas dari sebuah matriks simetris. Untuk sebuah $m\,X\,m$ matriks persegi **P**, **vec (P)** dinotasikan sebagai sebuah vektor kolom dari dimensi $m(m+1)/2$ seperti

$$\boldsymbol{vec(P)} = [p_{11,}\, p_{12,}\, p_{22}\;\; p_{13,}\, \dots, p_{mm}\,]^T$$

dimana p_{ij} dinotasikan (i,j) elemen dari matriks **P**. Dari definisi vec(.), (2.70) dapat disusun dengan

$$\begin{aligned}\boldsymbol{T_k(l)} &= \left[\boldsymbol{vec\left(D_1(l)\right), vec\left(D_2(l)\right), \dots, vec\left(p(l)\right)}\right]\boldsymbol{\theta} + \boldsymbol{vec\left(e(k,l)\right)} \\ &\equiv \boldsymbol{D(l)\theta} + \boldsymbol{vec\left(e(k,l)\right)} \qquad (2.72a)\end{aligned}$$

dengan

$$\begin{aligned}\boldsymbol{T_k(l)} &\equiv vec(\boldsymbol{Z(k,l)Z(k,l)}^T) \\ &= vec\left(\boldsymbol{\Pi}\{\boldsymbol{Y(k)} - \boldsymbol{Y(k-l)}\}\{\boldsymbol{Y(k)} - \boldsymbol{Y(k-l)}\}^T\boldsymbol{\Pi}^T\right) \qquad (2.72b)\end{aligned}$$

Dimana $\boldsymbol{vec(e(k,l))}$ menotasikan sebuah persamaan error vektor dengan *zero-mean.* Misalkan sebuah priori informasi dalam kovarians noise tidak tersedia, agar meminimumkan persamaan error di (2.72a), indeks performa quadratik dipilih:

$$\boldsymbol{J_N(\theta)} = \sum_{k=1}^{N}\sum_{l=1}^{L}\{\boldsymbol{T_k(l)} - \boldsymbol{D_i(l)\theta}\}^T\{\boldsymbol{T_k(l)} - \boldsymbol{D_i(l)\theta}\} \qquad (2.73)$$

Dimana **L** merupakan waktu korelasi maksimum. $\boldsymbol{J_N(\theta)}$ merupakan minimasi untuk

$$\boldsymbol{\theta(N)} = \left[\sum_{l=1}^{L}\boldsymbol{D(l)}^T\boldsymbol{D(l)}\right]^{-1}\frac{1}{N}\sum_{k=1}^{N}\sum_{l=1}^{L}\boldsymbol{D(l)}^T\boldsymbol{T_k(l)} \qquad (2.74)$$

Vektor **θ** yang tidak diketahui pada (2.72a) memiliki solusi unik ketika *sliding window size* α dipilih, seperti

$$\frac{(\alpha r - n)(\alpha r - n + 1)}{2} \geq p$$

Karena harus ada persamaan lebih daripada parameter yang tidak diketahui.

Untuk memeriksa sifat dari (2.74), masukkan (2.72a) ke (2.74):

$$\boldsymbol{\theta(N)} = \boldsymbol{\theta} + \left[\sum_{l=1}^{L} \boldsymbol{D(l)^T D(l)}\right]^{-1} \sum_{l=1}^{L} \boldsymbol{D(l)^T} \frac{1}{N} \sum_{k=1}^{N} \boldsymbol{vec(e(k,l))} \tag{2.75}$$

Diharapkan nilai dari **vec(e(k,l))** adalah nol berdasarkan pada (2.66) dan (2.70), $E\{\boldsymbol{\theta(N)}\} = \boldsymbol{\theta}$, sebagai contoh estimasinya tanpa bias. Sebagai tambahan, karena ***e(k,l)*** memiliki (2.71), $\boldsymbol{\theta(N)}$ konvergen untuk nilai **θ** sebagai **N** mendekati takhingga.

$$\boldsymbol{\theta(k)} = \frac{k-1}{k}\boldsymbol{\theta(k-1)} + \Theta(\boldsymbol{k}) \sum_{l=1}^{L} \boldsymbol{D(l)^T T_k(l)} \tag{2.76}$$

$$\Theta(\boldsymbol{k}) = \frac{1}{k}\left[\sum_{l=1}^{L} \boldsymbol{D(l)^T D(l)}\right]^{-1}$$

θ(k) diperoleh dari (2.76), estimasi kovarians noise **Q** dan **R** tersedia dari persamaan berikut:

$$\boldsymbol{Q(k)} = \sum_{i=1}^{p} \boldsymbol{Q_i \theta_i(k)}\ , \qquad \boldsymbol{R(k)} = \sum_{i=1}^{p} \boldsymbol{R_i \theta_i(k)}$$

dimana $\boldsymbol{\theta_i(k)}$ merupakan element ke i dari $\boldsymbol{\theta(k)}$.

2.4 *Linear Quadratic Regulator*

Linear-quadratic regulator (LQR) adalah algoritma yang digunakan untuk menghasilkan penguat optimal (*K*) sebagai pengganti blok pengendali sehingga persamaan (2.77) membuat fungsi kriteria pada persamaan (2.78) sekecil mungkin.

$$u_k = -K_N x_k \tag{2.77}$$

$$J(u) = \int_0^{\infty} \left(x^T Q x + u^T R u + 2x^T N u\right) dt \tag{2.78}$$

Nilai K pada persamaan (2.77) diperoleh melalui persamaan

$$K = R^{-1}\left(B^T S + N^T\right) \tag{2.79}$$

dimana nilai S merupakan solusi dari persamaan *Riccati*

$$A^T S + SA - (SB + N)R^{-1}(B^T S + N^T) + Q = 0 \tag{2.80}$$

Untuk menghasilkan sinyal pengendali pada satu langkah ke depan u_{N+1}, maka digunakan variabel keadaan untuk satu langkah ke depan $\hat{x}_{N+1|N}$ yang diperoleh melalui algoritma yang diuji pada penelitian ini. Dengan menggunakan persamaan (2.77), maka diperoleh

$$u_{N+1|N} = -K_N x_{N+1|N} \tag{2.81}$$

Pada perancangan pengendali ini juga terdapat blok *pre-compensator (V)* untuk mengatur agar sinyal keluaran sama dengan sinyal referensi. Berdasarkan diagram blok di atas, persamaan sinyal kendalinya :

$$u(k) = Vw(k) - Kx(k) \tag{2.82}$$

Pada saat kondisi tunak, yaitu saat nilai $y_s(k) = w(k)$, nilai keadaan saat pencuplikan setelahnya sama dengan nilai keadaan sekarang, yaitu $x(k+1) = x(k) = x_s(k)$. Dengan mensubstitusi persamaan (2.82) ke dalam persamaan model proses (2.1) dan (2.2), maka diperoleh

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ &= Ax(k) + BVw(k) - BKx(k) \\ x_s(k) &= (A - BK)x_s(k) + BVw(k) \\ (I - A + BK)x_s(k) &= BVw(k) \\ x_s(k) &= (I - A + BK)^{-1} BVw(k) \end{aligned} \tag{2.83}$$

dengan $x_s(k)$ merupakan nilai keadaan tunak. Substitusi persamaan (2.82) dan (2.83) ke persamaan model proses dalam keadaan tunak menghasilkan :

$$\begin{aligned} y_s(k) &= Cx_s(k) + Du(k) \\ &= Cx_s(k) + DVw(k) - DKx_s(k) \\ &= (C - DK)x_s(k) + DVw(k) \\ &= (C - DK)(I - A + BK)^{-1} BVw(k) + DVw(k) \\ y_s(k) &= \left((C - DK)(I - A + BK)^{-1} B + D\right)Vw(k) \end{aligned} \tag{2.84}$$

Dalam keadaan tunak, keluaran sistem diharapkan sama dengan referensi, yaitu $y_s(k) = w(k)$ sehingga :

$$V = \left((C - DK)(I - A + BK)^{-1}B + D\right)^{-1} \tag{2.85}$$

BAB 3
PERANCANGAN OBSERVER FILTER KALMAN UNTUK SISTEM TATA UDARA PRESISI

3.1 Perancangan Filter Kalman dengan Menggunakan Matlab

Observer Filter Kalman dirancang menggunakan m-file dan C-Mex Matlab. M-file Matlab yang digunakan pada perancangan ini ada dua macam: satu m-file berisi program utama algoritma Filter Kalman yang nantinya dapat dipanggil pada program yang ada pada m-file satu lagi berisi niali inputan matriks **A**, **B**, **C**, **D**, serta sinyal kendali **u**, *state* baru, output **y**, matriks **P**, matriks **Q**, dan matriks **R**. Berikut ini akan dibahas algoritma observer Filter Kalman yang diterapkan pada setiap m-file.

3.1.1 Algoritma Observer Filter Kalman pada M-file 'kalman.m'

Program yang ada pada M-file ini berisi tentang algoritma Filter Kalman yang terdiri dari *measurement update* dan *time update*. M-file 'kalman.m' ini yang akan dipanggil dalam M-file 'kalmanTest.m' yang nantinya akan dilakukan komputasi nilai *state* estimasi dari model sistem tata udara presisi yang sudah diidentifikasi dan didapatkan nilai dari matriks **A**, **B**, **C**, dan **D** nya. Dalam M-file ini inputan yang dibutuhkan berupa nilai-nilai dari matriks **A**, **B**, **C**, **D**, matriks **u**, matriks **x** prediksi, matriks **y** *(output)*, matriks **P** (prediksi error kovarian), matriks **Q** (kovarians error pengukuran), dan matriks **R** (kovarian error proses). Sedangkan keluaran yang dihasilkan dari M-file ini adalah *update* dari **x** prediksi dan *update* dari matriks **P**.

Algoritma Filter Kalman yang ditulis dalam M-File ini mengikuti pola algoritma Filter Kalman pada umumnya. Penerapan algoritma ini sama seperti yang telah dibahas dalam Bab 2. Algoritmanya adalah sebagai berikut:

Algoritma *Time update*

$$\hat{\boldsymbol{x}}_k^- = \hat{\boldsymbol{x}}_{k\,k-1}^- + B\boldsymbol{u}_{k-1}$$

$$\boldsymbol{P}_k^- = \boldsymbol{A}\boldsymbol{P}_k\boldsymbol{A}^T + \boldsymbol{Q}$$

Algoritma *Measurement Update*

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}^T [\boldsymbol{H}\boldsymbol{P}_k^- \boldsymbol{H}^T + \boldsymbol{R}]^{-1}$$

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k(\boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_k^-)$$

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{P}_k^-$$

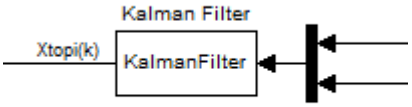
Untuk nilai awal $\hat{\boldsymbol{x}}_{k-1}$ dan $\boldsymbol{P}_{k-1}$ adalah nol yang nanti akan dipanggil pada M-file 'kalmanTest.m'.

3.1.2 Algoritma Observer Filter Kalman pada M-file 'kalmanTest.m'

M-file ini berisi inisialisasi *state* dan output. Nilai awal untuk *state* adalah matriks nol dimensi 8x1 sedangkan untuk nilai awal untuk output adalah matriks noll dimensi 2x1. Atau dengan kata lain *state* dan output diinisialisasi dengan nilai nol. Di M-file ini juga nilai dari matriks **A**, **B**, **C**, dan **D** dari hasil identifikasi didefinisikan. Inisialisasi untuk matriks P adalah matriks identitas berdimensi 8x8. Nilai untuk matriks kendali **u** ditentukan antara nilai 0 sampai 2,55. Sehingga dalam penelitian ini untuk mendapatkan nilai tersebut digunakan fungsi random yang dikalikan dengan konstanta 2,55. Untuk nilai dari matriks **Q** dan matriks **R** dibagi menjadi dua cara penentuan yaitu diset secara manual dengan berbagai variasi nilai yang nilainya semakin mengecil yang bertujuan untuk mencari nilai terbaik dari variasi nilai **Q** dan **R** dari penentuan manual. Yang kedua untuk menentukan nilai dari matriks **Q** dan **R** adalah dengan menggunakan algoritma dari paper [16].

3.1.3 Perancangan Filter Kalman Menggunakan C-Mex S-Function

Dalam blok S-Function KalmanFilter ini ada satu port masukan dan satu port keluaran. Dalam satu port inputan terdapat dua input yang dimasukkan dalam mux, serta setiap input masing-masing terdapat dua outputan. Setiap *port* outputan terdiri dari dua output. Jadi total masukan untuk input terdapat 4 masukan. Dua input awal dihasilkan dari matriks kendali $\boldsymbol{u_1}$ dan $\boldsymbol{u_2}$, sedangkan outputan kedua dihasilkan dari *update* nilai untuk y yang dihasilkan. Inputan kendali yang dihasilkan nilainya antara 0-2.55 karena sebelum input ini masuk ke dalam blok S-Function FilterKalman, terdapat blok saturasi yang nilainya diset antara 0-2.55. Sampling time yang digunakan adalah sebesar 5 sekon.



Gambar 3.1 Blok S-Function Kalman Filter

Algoritma yang ditulis dalam blok S-Function Kalman Filter sama seperti yang ditulis dalam M-File. Dalam blok KalmanFilter ini, nilai matriks **P** di*update* setiap kali pencuplikan. Besarnya sampling time yang digunakan dalam blok ini adalah 5 sekon.

3.1.4 Membandingkan Keluaran *State* pada M-File dengan Keluaran *State* pada C-Mex

Untuk mendapatkan keluaran yang sama maka digunakan inputan yang sama pula antara algoritma Filter Kalman pada C-Mex dengan algoritma Filter Kalman pada M-File. Sinyal inputan yang digunakan merupakan sinyal sejumlah data yang direkam dalam workspace yang nantinya dapat digunakan sebagai input dalam C-Mex S-Function maupun sebagi input dalam M-File.

Nilai sinyal input $u1$ adalah sebesar 0.9 sedangkan nilai $u2$ sebesar 2.55. Sinyal ini mempunyai nilai yang konstan. Input yang digunakan mempunyai dimensi sebesar 801x2. Hal ini dikarenakan sinyal input yang dibutuhkan oleh algoritma Filter Kalman untuk memprediksi *state* pada sistem tata udara presisi memiliki jumlah input dua. Dalam prosesnya, setiap satu baris input (dua kolom nilai) diambil sebagai masukan dalam algortima Filter Kalman baik dalam C-Mex.

Sedangkan untuk input pada M-File, terlebih dahulu disimpan dalam data_input.mat yang nantinya dengan fungsi load data_input.mat, nilai dari inputan ini dapat digunakan dalam program M-File. Kedelapan *state* hasil keluaran dari M-File dan C-Mex kemudian dibandingkan untuk mengetahui kesamaan dari *state* hasil C-Mex dengan *state* hasil dari M-File.

3.2.1 Pengujian Menggunakan Model CSTR *(Continuous Stirred Tank Reactor)*

Dalam perancangan desain observer Filter Kalman ini, untuk menguji apakah Filter Kalman memiliki performa yang baik, maka desain observer Filter Kalman ini diujikan pada model sistem CSTR *(Continuous Stirred Tank Reactor)*

merupakan suatu tangki reaktor yang digunakan untuk mencampur dua atau lebih bahan kimia dalam bentuk cairan dengan menggunakan pengaduk *(mixer).* Pada *Continuous Stirred-Tank Reactor* terdapat heater yang akan menghasilkan panas untuk mengatur temperatur cairan pada harga tertentu. Model sistem CSTR merupakan sistem LTI *(Linear Time Invariant).* CSTR ini juga merupakan sistem multivariabel yang memiliki dua input dan dua output. Model sistem dari CSTR didapat dari Help MATLAB sebagai berikut,

$$\boldsymbol{A} = \begin{bmatrix} -0.0285 & -0.0014 \\ -0.0371 & -0.1476 \end{bmatrix}$$

$$\boldsymbol{B} = \begin{bmatrix} -0.0850 & 0.0238 \\ 0.0802 & 0.4462 \end{bmatrix}$$

$$\boldsymbol{C} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$\boldsymbol{D} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Dari model ruang keadaan sistem tersebut, kemudian dilihat pula nilai *state* asli dari sistem CSTR dan kemudian dibandingkan nilai *state*nya dengan nilai *state* prediksi dari algoritma Filter Kalman. Dengan nilai kovarians error matriks **Q** dan **R** yang nilainya divariasikan dengan menggunakan nilai-nilai:

$$\boldsymbol{Q} = \begin{bmatrix} 0.2 & 0 \\ 0 & 0.2 \end{bmatrix} \quad \boldsymbol{R} = \begin{bmatrix} 0.2 & 0 \\ 0 & 0.2 \end{bmatrix}$$

$$\boldsymbol{Q} = \begin{bmatrix} 0.02 & 0 \\ 0 & 0.02 \end{bmatrix} \quad \boldsymbol{R} = \begin{bmatrix} 0.02 & 0 \\ 0 & 0.02 \end{bmatrix}$$

$$\boldsymbol{Q} = \begin{bmatrix} 0.002 & 0 \\ 0 & 0.002 \end{bmatrix} \quad \boldsymbol{R} = \begin{bmatrix} 0.002 & 0 \\ 0 & 0.002 \end{bmatrix}$$

$$\boldsymbol{Q} = \begin{bmatrix} 0.0002 & 0 \\ 0 & 0.0002 \end{bmatrix} \quad \boldsymbol{R} = \begin{bmatrix} 0.0002 & 0 \\ 0 & 0.0002 \end{bmatrix}$$

Serta nilai **Q** dan **R** yang didapat dari optimasi menggunakan algoritma penentuan matriks kovarians pada sistem dengan bias.

3.2.2 Penerapan Algoritma Pencari Nilai Optimasi Q dan R pada Sistem CSTR

Penerapan algoritma ini pada sistem CSTR akan dicari nilai dari matriks kovarians **Q** dan **R** dari sistem CSTR dengan menggunakan persamaan berikut:

$$\boldsymbol{Q}(\boldsymbol{k}) = \sum_{i=1}^{p} \boldsymbol{Q}_i \boldsymbol{\theta}_i(k) \ , \qquad \boldsymbol{R}(k) = \sum_{i=1}^{p} \boldsymbol{R}_i \boldsymbol{\theta}_i(k)$$

Dan didapat nilai dari parameter θ dan Θ dari:

$$\boldsymbol{\theta(k)} = \frac{k-1}{k}\theta(k-1) + \Theta(k)\sum_{l=1}^{L} D(l)^T T_k(l)$$

$$\Theta(\boldsymbol{k}) = \frac{1}{k}\left[\sum_{l=1}^{L} \boldsymbol{D(l)^T D(l)}\right]^{-1}$$

Dengan menentukan nilai parameter dari k adalah 3 dan nilai parameter α sebesar 2 sehingga didapat nilai paramater p sebesar 3 dengan menggunakan persamaan $\frac{(\alpha r - n)(\alpha r - n + 1)}{2} \geq p$. Dengan r dan n merupakan dimensi matriks pada sistem CSTR yang masing-masing bernilai 2.

Nilai dari $\boldsymbol{D_i(l)}$ didapat dengan menggunakan persamaan

$$\boldsymbol{D_i(l) = \Pi[BQ_i(l)B^T + R_i(l)]\Pi^T}$$

Dengan i merupakan banyaknya iterasi yang dilakukan. Dalam penelitian ini digunakan iterasi sebanyak 50 kali untuk mendapatkan nilai parameter $\boldsymbol{D_i(l)}$. Dari hasil perhitungan, didapat nilai dari matriks $\boldsymbol{\Pi}$ adalah sebesar

$$\begin{bmatrix} 0.1459 & 0.0366 & 0.9886 & -0.0027 \\ 0.0016 & 0.0285 & 0.0014 & 0.9996 \end{bmatrix}$$

Sedangkan nilai dari matriks **B** didapat dari hasil perhitungan $\boldsymbol{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}$

Nilai dari parameter **Q** dan **R** diset awal pada nilai

$\boldsymbol{Q} = \begin{bmatrix} 0.0001 & 0 \\ 0 & 0.0001 \end{bmatrix}$ dan $\boldsymbol{R} = \begin{bmatrix} 0.0001 & 0 & 0.000 & 0.000 \\ 0 & 0.0001 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.0001 & 0 \\ 0.000 & 0.000 & 0 & 0.0001 \end{bmatrix}$

Sehingga dengan melakukan komputasi penentuan nilai $\boldsymbol{Q_i(l)}$ dan $\boldsymbol{R_i(l)}$ didapat nilai

$$\boldsymbol{Q(k,l) = Q(l)} = \begin{matrix} 2\boldsymbol{Q_b} - \boldsymbol{Q_b S_Q^l} - (\boldsymbol{Q_b S_Q^l})^T, & l \leq \alpha - 1 \\ 2\boldsymbol{Q_b}, & > \alpha - 1 \end{matrix}$$

$$\boldsymbol{R(k,l) = R(l)} = \begin{matrix} 2\boldsymbol{R_b} - \boldsymbol{R_b S_R^l} - (\boldsymbol{R_b S_R^l})^T, & l \leq \alpha - 1 \\ 2\boldsymbol{R_b}, & > \alpha - 1 \end{matrix}$$

dengan

$$\boldsymbol{S_q} = \begin{bmatrix} 0_{qXq} & I_{qXq} & 0_{qXq} & \cdots & 0_{qXq} \\ 0_{qXq} & 0_{qXq} & I_{qXq} & \cdots & 0_{qXq} \\ 0_{qXq} & 0_{qXq} & 0_{qXq} & \cdots & 0_{qXq} \\ \vdots & \vdots & \vdots & \ddots & I_{qXq} \\ 0_{qXq} & 0_{qXq} & 0_{qXq} & 0_{qXq} & 0_{qXq} \end{bmatrix}$$

dan

$$\boldsymbol{S_r} = \begin{bmatrix} 0_{rXr} & I_{rXr} & 0_{rXr} & \cdots & 0_{rXr} \\ 0_{rXr} & 0_{rXr} & I_{rXr} & \cdots & 0_{rXr} \\ 0_{rXr} & 0_{rXr} & 0_{rXr} & \cdots & 0_{rXr} \\ \vdots & \vdots & \vdots & \ddots & I_{rXr} \\ 0_{rXr} & 0_{rXr} & 0_{rXr} & 0_{rXr} & 0_{rXr} \end{bmatrix}$$

dari berbagai persamaan tersebut kemudian didapat nilai dari $\boldsymbol{D_i(l)}$ adalah

$$\boldsymbol{D_i(1)} = 0.001 \times \begin{bmatrix} 0.3866 & 0.0953 \\ 0.0953 & 0.4001 \end{bmatrix}$$

$$\boldsymbol{D_i(2)} = 0.001 \times \begin{bmatrix} 0.3944 & 0.1974 \\ 0.1974 & 0.4004 \end{bmatrix}$$

Komputasi dilakukan secara terus-menerus sampai mencapai iterasi sebanyak 50 kali. Sehingga didapatkan $\sum_{l=1}^{50}[\boldsymbol{D(l)}^T\boldsymbol{D(l)}]$ sebesar $10^{-6}\text{X}\begin{bmatrix} 0.5476 & 0.3888 \\ 0.3888 & 0.5677 \end{bmatrix}$

Untuk nilai $\Theta(\mathbf{1})$, $\Theta(\mathbf{2})$, dan $\Theta(\mathbf{3})$ didapat sebagai berikut:

$$\Theta(\mathbf{1}) = 10^{-6} \times \begin{bmatrix} 3.5535 & -2.4333 \\ -2.4333 & 3.4276 \end{bmatrix}$$

$$\Theta(\mathbf{2}) = 10^{-6} \times \begin{bmatrix} 1.7767 & -1.2166 \\ -1.2166 & 1.7138 \end{bmatrix}$$

$$\Theta(\mathbf{3}) = 10^{-6} \times \begin{bmatrix} 1.1845 & -0.8111 \\ -0.8111 & 1.1425 \end{bmatrix}$$

Sedangkan untuk nilai $\boldsymbol{\theta(1)}$, $\boldsymbol{\theta(2)}$, dan $\boldsymbol{\theta(3)}$ didapat sebagai berikut:

$$\boldsymbol{\theta(1)} = \begin{bmatrix} 1.000 & 0.000 \\ 0.000 & 1.000 \end{bmatrix}$$

$$\boldsymbol{\theta(2)} = \begin{bmatrix} 1.000 & 0.000 \\ 0.000 & 1.000 \end{bmatrix}$$

$$\boldsymbol{\theta(3)} = \begin{bmatrix} 1.000 & 0.000 \\ 0.000 & 1.000 \end{bmatrix}$$

Sehingga didapat ni $\boldsymbol{Q}(1)$, $\boldsymbol{Q}(2)$, dan $\boldsymbol{Q}(3)$ didapat sebagai berikut:

$$\boldsymbol{Q}(1) = 10^{-3} \times \begin{bmatrix} 0.6 & -0.1 \\ -0.1 & 0.6 \end{bmatrix}$$

$$\boldsymbol{Q}(2) = 10^{-3} \times \begin{bmatrix} 0.6 & 0 \\ 0 & 0.6 \end{bmatrix}$$

$$\boldsymbol{Q}(3) = 10^{-3} \times \begin{bmatrix} 0.6 & -0.1 \\ -0.1 & 0.6 \end{bmatrix}$$

Dengan cara serupa didapat nilai dari $\boldsymbol{R}(1)$, $\boldsymbol{R}(2)$, dan $\boldsymbol{R}(3)$ sebagai berikut:

$$\boldsymbol{R}(1) = 10^{-3} \times \begin{bmatrix} 1 & -0.1667 \\ -0.1667 & 1 \end{bmatrix}$$

$$\boldsymbol{R}(2) = \begin{bmatrix} 0.0030 & 0.0000 \\ 0.0000 & 0.0030 \end{bmatrix}$$

$$\boldsymbol{R}(3) = 10^{-3} \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Digunakan matriks $\boldsymbol{Q}(3)$ dan $\boldsymbol{R}(3)$ sebagai kovarians matriks $\boldsymbol{Q}$ dan $\boldsymbol{R}$ untuk mengestimasi nilai *state* CSTR.

$$\boldsymbol{Q} = 10^{-3} \times \begin{bmatrix} 0.6 & -0.1 \\ -0.1 & 0.6 \end{bmatrix} \text{ dan } \boldsymbol{R} = 10^{-3} \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Dalam pengujian observer Filter Kalman, dilakukan variasi pada noise yang diberikan. Ada tiga variasi noise pengukuran dan noise proses yaitu noise dengan nilai 0,1; 0,001; dan 0,0001 untuk masing-masing noise.

3.3 Pengujian Pada Model Sistem Tata Udara Presisi

Dalam penelitian ini akan dicari nilai *state* variabel takterukur dari kedelapan *state* sistem tata udara presisi, kedelapan *state* tersebut adalah:
State 1 adalah kabinet (^0C); *State* 2 adalah kelembaban relatif kabinet (kg/kg); *State* 3 adalah suhu udara keluaran evaporator (^0C); *State* 4 adalah suhu udara diantara *dry region* dan *wet region* evaporator (^0C); *State* 5 adalah suhu udara keluaran kondenser kedua (^0C); *State* 6 adalah suhu dinding evaporator (^0C); *State* 7 adalah suhu dinding kondenser kedua (^0C); *State* 8 adalah kelembaban spesifik keluaran evaporator (kg/kg).

3.3.1 Model Linier Sistem Tata Udara

Model proses dari sistem tata udara presisi yang dipakai pada perancangan observer Filter Kalman ini merupakan hasil dari identifikasi N4SID secara *offline* terhadap sistem tata udara presisi, yang didapat persamaan ruang keadaan dari sistem tata udara presisi pada penelitian sebelumnya. Model ruang keadaan yang didapat merupakan model linear.

Matriks **A**, **B**, **C**, **D** dari model linear yang didapat dari metode N4SID secara *offline* adalah sebagai berikut:

$$A = \begin{bmatrix} 0.974525 & -0.01366 & -0.00441 & -0.00099 & -0.01133 & 0.018842 & -0.00418 & -0.01468 \\ -0.07666 & 0.832216 & 0.012794 & 0.17223 & 0.1086 & -0.03933 & 0.092703 & 0.020134 \\ 0.068861 & 0.075798 & 0.963319 & -0.0198 & 0.023203 & -0.01762 & 0.000708 & -0.00013 \\ 0.079615 & 0.070479 & 0.012201 & 0.516287 & -0.75091 & -0.17064 & -0.50661 & 0.317302 \\ 0.060414 & 0.008054 & -0.06385 & 0.552407 & 0.528967 & 0.143632 & -0.04947 & -0.10849 \\ 0.098599 & 0.11378 & 0.019149 & 0.097361 & 0.088844 & -0.67208 & -0.12845 & -0.32955 \\ 0.067682 & 0.054275 & 0.000742 & -0.04147 & -0.10308 & 0.393316 & 0.026605 & -0.02373 \\ -0.02239 & -0.0486 & -0.00423 & -0.08236 & -0.01574 & -0.17013 & -0.37892 & -0.45597 \end{bmatrix}$$

$$B = \begin{bmatrix} -0.0033 & 0.008158 \\ -0.00057 & -0.00557 \\ 0.006841 & -0.01526 \\ 0.014192 & -0.03765 \\ 0.010281 & -0.02706 \\ 0.017131 & -0.04519 \\ 0.013584 & -0.03685 \\ -0.00208 & 0.004475 \end{bmatrix}$$

$$C = \begin{bmatrix} -12.1395 & -0.69088 & 0.114385 & 0.038945 & 0.328561 & -0.06891 & -0.21067 & 0.096712 \\ -0.20531 & 0.024634 & 0.395873 & 0.011623 & 0.022705 & -0.01259 & -0.01089 & 0.007233 \end{bmatrix}$$

$$D = \begin{bmatrix} 0.03306 & -0.08744 \\ 0.0019 & -0.00504 \end{bmatrix}$$

Nilai eigen A dari model linear N4SID *offline* adalah

$$\lambda(A) = \begin{bmatrix} -0.6116 + 0.1608i \\ -0.6116 - 0.1608i \\ 0.5226 + 0.6412i \\ 0.5226 - 0.6412i \\ 0.1295 \\ 0.8102 \\ 0.9761 + 0.0109i \\ 0.9761 - 0.0109i \end{bmatrix}$$

Hasil tes matriks *Observability* dari model hasil linearisasi adalah 8 dari 8, yang artinya adalah keadaan (*state*) yang dapat diobservasi pada model linear sistem tata udara presisi ini adalah sebanyak 8 *state* dari totalnya 8 *state*. Hasil uji matriks *Controllability* dari model hasil linearisasi adalah 8 dari 8, yang artinya

adalah kedaan (*state*) yang dapat dikendalikan pada model linear sistem tata udara presisi ini adalah sebanyak 8 *state* dari total 8 *state*.

3.3.2 Penerapan algoritma Penentuan Matriks Kovarian Error Q dan R Sistem Tata Udara Presisi

Penerapan algoritma ini pada sistem tata udara presisi akan dicari nilai dari matriks kovarians **Q** dan **R** dari sistem sistem tata udara presisi dengan menggunakan persamaan berikut:

$$Q(k) = \sum_{i=1}^{p} Q_i \theta_i(k) \ , \qquad R(k) = \sum_{i=1}^{p} R_i \theta_i(k)$$

Dan didapat nilai dari parameter θ dan Θ dari:

$$\theta(k) = \frac{k-1}{k}\theta(k-1) + \Theta(k)\sum_{l=1}^{L} D(l)^T T_k(l)$$

$$\Theta(k) = \frac{1}{k}\left[\sum_{l=1}^{L} D(l)^T D(l)\right]^{-1}$$

Dengan menentukan nilai parameter dari k adalah 3 dan nilai parameter α sebesar 8 sehingga didapat nilai paramater p sebesar 36 dengan menggunakan persamaan $\frac{(\alpha r - n)(\alpha r - n + 1)}{2} \geq p$. Dengan r dan n merupakan dimensi matriks pada sistem tata udara presisi yang masing-masing bernilai 2.

Nilai dari $D_i(l)$ didapat dengan menggunakan persamaan $D_i(l) = \Pi[BQ_i(l)B^T + Ri(l)\Pi T$

Dengan i merupakan banyaknya iterasi yang dilakukan. Dalam penelitian ini digunakan iterasi sebanyak 50 kali untuk mendapatkan nilai parameter $D_i(l)$. Dari hasil perhitungan, didapat nilai dari matriks Π adalah sebesar

$$\begin{bmatrix} 0.1459 & 0.0366 & 0.9886 & -0.0027 \\ 0.0016 & 0.0285 & 0.0014 & 0.9996 \end{bmatrix}$$

Sedangkan nilai dari matriks **B** didapat dari hasil perhitungan $\boldsymbol{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}$

Nilai dari parameter **Q** dan **R** diset awal pada nilai

$$\boldsymbol{Q} = 10^{-5} \times \begin{bmatrix} 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{bmatrix}$$

$$\boldsymbol{R} = 10^{-3} \times \begin{bmatrix} 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \end{bmatrix}$$

Sehingga dengan melakukan komputasi penentuan nilai $\boldsymbol{Q_i(l)}$ dan $\boldsymbol{R_i(l)}$ didapat nilai

$$\boldsymbol{Q(k,l)} = \boldsymbol{Q(l)} = \begin{matrix} 2\boldsymbol{Q_b} - \boldsymbol{Q_b S_Q^l} - (\boldsymbol{Q_b S_Q^l})^T, & l \leq \alpha - 1 \\ 2\boldsymbol{Q_b}, & > \alpha - 1 \end{matrix}$$

$$\boldsymbol{R(k,l)} = \boldsymbol{R(l)} = \begin{matrix} 2\boldsymbol{R_b} - \boldsymbol{R_b S_R^l} - (\boldsymbol{R_b S_R^l})^T, & l \leq \alpha - 1 \\ 2\boldsymbol{R_b}, & > \alpha - 1 \end{matrix}$$

dengan

$$\boldsymbol{S_q} = \begin{bmatrix} 0_{qXq} & I_{qXq} & 0_{qXq} & \cdots & 0_{qXq} \\ 0_{qXq} & 0_{qXq} & I_{qXq} & \cdots & 0_{qXq} \\ 0_{qXq} & 0_{qXq} & 0_{qXq} & \cdots & 0_{qXq} \\ \vdots & \vdots & \vdots & \ddots & I_{qXq} \\ 0_{qXq} & 0_{qXq} & 0_{qXq} & 0_{qXq} & 0_{qXq} \end{bmatrix}$$

dan

$$\boldsymbol{S_r} = \begin{bmatrix} 0_{rXr} & I_{rXr} & 0_{rXr} & \dots & 0_{rXr} \\ 0_{rXr} & 0_{rXr} & I_{rXr} & \dots & 0_{rXr} \\ 0_{rXr} & 0_{rXr} & 0_{rXr} & \cdots & 0_{rXr} \\ \vdots & \vdots & \vdots & \ddots & I_{rXr} \\ 0_{rXr} & 0_{rXr} & 0_{rXr} & 0_{rXr} & 0_{rXr} \end{bmatrix}$$

dari berbagai persamaan tersebut kemudian didapat nilai dari $\boldsymbol{D_i(l)}$ adalah

$$\boldsymbol{D_i(1)} = 10^{-3} \times \begin{bmatrix} 0.1414 & -0.0199 & -0.0859 & -0.0822 & 0.0223 & 0.0394 & 0.0084 & -0.0331 \\ -0.0199 & 0.1719 & 0.0089 & -0.0768 & -0.0051 & 0.0839 & 0.0037 & 0.0089 \\ -0.0859 & 0.0089 & 0.1427 & 0.0191 & -0.0984 & -0.0174 & 0.0228 & 0.0084 \\ -0.0822 & -0.0768 & 0.0191 & 0.1741 & -0.0093 & -0.0520 & -0.0016 & 0.0323 \\ 0.0223 & -0.0051 & -0.0984 & -0.0093 & 0.1497 & 0.0043 & -0.0993 & -0.0004 \\ 0.0394 & 0.0839 & -0.0174 & -0.0520 & 0.0043 & 0.1330 & -0.0028 & -0.0821 \\ 0.0084 & 0.0037 & 0.0228 & -0.0016 & -0.0993 & -0.0028 & 0.1252 & 0.0001 \\ -0.0331 & 0.0089 & 0.0084 & 0.0323 & -0.0004 & -0.0821 & 0.0001 & 0.1147 \end{bmatrix}$$

$$\boldsymbol{D_i(2)} = 10^{-3} \times \begin{bmatrix} 0.1816 & -0.0697 & 0.0043 & 0.0142 & -0.0905 & -0.0003 & 0.0023 & 0.0747 \\ -0.0697 & 0.1424 & -0.0524 & 0.0337 & 0.0096 & -0.0138 & -0.0002 & 0.0597 \\ 0.0043 & -0.0524 & 0.1767 & -0.0857 & 0.0011 & 0.0175 & -0.0938 & -0.0325 \\ 0.0142 & 0.0337 & -0.0857 & 0.1789 & -0.0056 & 0.0821 & 0.0100 & -0.0117 \\ -0.0905 & 0.0096 & 0.0011 & -0.0056 & 0.1229 & -0.0060 & 0.0008 & -0.0034 \\ -0.0003 & -0.0138 & 0.0175 & 0.0821 & -0.0060 & 0.0943 & -0.0104 & -0.0564 \\ 0.0023 & -0.0002 & -0.0938 & 0.0100 & 0.0008 & -0.0104 & 0.1225 & 0.0021 \\ 0.0747 & 0.0597 & -0.0325 & -0.0117 & -0.0034 & -0.0564 & 0.0021 & 0.1984 \end{bmatrix}$$

Komputasi dilakukan secara terus-menerus sampai mencapai iterasi sebanyak 50 kali. Sehingga didapatkan $\sum_{l=1}^{50}[\boldsymbol{D(l)}^T \boldsymbol{D(l)}]$ sebesar

$$10^{-3} \times \begin{bmatrix} 1.20 & 0.00 & -0.1 & -0.1 & -0.1 & 0.10 & -0.1 & 0.10 \\ 0.00 & 1.20 & -0.1 & -0.1 & 0.00 & 0.10 & -0.1 & 0.10 \\ -0.1 & -0.1 & 1.20 & 0.00 & -0.1 & 0.10 & 0.00 & 0.10 \\ -0.1 & -0.1 & 0.00 & 1.20 & -0.1 & 0.00 & 0.00 & 0.00 \\ -0.1 & 0.00 & -0.1 & -0.1 & 1.20 & 0.10 & -0.1 & 0.10 \\ 0.00 & 0.10 & 0.10 & 0.00 & 0.10 & 1.10 & 0.00 & -0.1 \\ -0.1 & -0.1 & 0.00 & 0.00 & -0.1 & 0.00 & 1.20 & 0.00 \\ 0.10 & 0.10 & 0.10 & 0.00 & 0.10 & -0.1 & 0.00 & 1.20 \end{bmatrix}$$

Untuk nilai $\theta(\mathbf{1})$, $\theta(\mathbf{2})$, dan $\theta(\mathbf{3})$ didapat sebagai berikut:

$$\theta(\mathbf{1}) = \begin{bmatrix} 890.9829 & 72.16660 & 71.98330 & 56.02650 & 71.59750 & -126.837 & 75.46630 & -96.2845 \\ 72.16660 & 895.7857 & 73.95980 & 61.51630 & 66.10920 & -126.029 & 114.6552 & -92.8118 \\ 71.98330 & 73.95980 & 866.3511 & 35.10880 & 69.65060 & -93.1320 & 56.69080 & -85.3074 \\ 56.02650 & 61.51630 & 35.10880 & 829.0270 & 62.02770 & -56.6603 & 41.62850 & -41.6730 \\ 71.59750 & 66.10920 & 69.65060 & 62.02770 & 884.2625 & -122.1550 & 71.85680 & -102.328 \\ -126.837 & -126.029 & -93.1320 & -56.6603 & -122.155 & 998.0644 & -56.2348 & 162.7986 \\ 75.46630 & 114.6552 & 56.69080 & 41.62850 & 71.85680 & -56.2348 & 896.1110 & -52.7568 \\ -96.2845 & -92.8118 & -85.3074 & -41.6730 & -102.3289 & 162.7986 & -52.7568 & 912.4134 \end{bmatrix}$$

$$\theta(\mathbf{2}) = \begin{bmatrix} 445.4915 & 36.08330 & 35.99170 & 28.01330 & 35.79870 & -63.4187 & 37.73320 & -48.1423 \\ 36.08330 & 447.8928 & 36.97990 & 30.75810 & 33.05460 & -63.0149 & 57.32760 & -46.4059 \\ 35.99170 & 36.97990 & 433.1756 & 17.55440 & 34.82530 & -46.5660 & 28.34540 & -42.6537 \\ 28.01330 & 30.75810 & 17.55440 & 414.5135 & 31.01390 & -28.3301 & 20.81420 & -20.8365 \\ 35.79870 & 33.05460 & 34.82530 & -28.3301 & 442.1312 & -61.0775 & 35.92840 & -51.1645 \\ -63.4187 & -63.0149 & -46.5660 & -56.6603 & -61.0775 & 499.0322 & -28.1174 & 81.39930 \\ 37.73320 & 57.32760 & 28.34540 & 20.81420 & 35.92840 & -28.1174 & 448.0555 & -26.3784 \\ -48.1423 & -46.4059 & -42.6537 & -20.8365 & -51.1645 & 81.39930 & -26.3784 & 456.2067 \end{bmatrix}$$

$$\theta(\mathbf{3}) = \begin{bmatrix} 296.9943 & 24.0555 & 23.9944 & 18.6755 & 23.8658 & -42.2791 & 25.1554 & -32.0948 \\ 24.0555 & 298.5952 & 24.6533 & 20.5054 & 22.0364 & -42.0099 & 38.2184 & -30.9373 \\ 23.9944 & 24.6533 & 288.7837 & 11.7029 & 23.2169 & -31.0440 & 18.8969 & -28.4358 \\ 18.6755 & 20.5054 & 11.7029 & 276.3423 & 20.6759 & -18.8868 & 13.8762 & -13.8910 \\ 23.8658 & 22.0364 & 23.2169 & 20.6759 & 294.7542 & -40.7183 & 23.9523 & -34.1096 \\ -42.2791 & -42.0099 & -31.0440 & -18.8868 & -40.7183 & 332.6881 & -18.7449 & 54.2662 \\ 25.1554 & 38.2184 & 18.8969 & 13.8762 & 23.9523 & -18.7449 & 298.7037 & -17.5856 \\ -32.0948 & -30.9373 & -28.4358 & -13.8910 & -34.1096 & 54.2662 & -17.5856 & 304.1378 \end{bmatrix}$$

Sedangkan nilai untuk $\boldsymbol{\theta}(\mathbf{1})$, $\boldsymbol{\theta}(\mathbf{2})$, dan $\boldsymbol{\theta}(\mathbf{3})$ didapat sebagai berikut:

$$\boldsymbol{\theta}(\mathbf{1}) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{\theta}(\mathbf{2}) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{\theta}(\mathbf{3}) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Sehingga didapat ni $\boldsymbol{Q}(\mathbf{1})$, $\boldsymbol{Q}(\mathbf{2})$, dan $\boldsymbol{Q}(\mathbf{3})$ didapat sebagai berikut:

$$\boldsymbol{Q}(\mathbf{1}) = 10^{-3} \times \begin{bmatrix} 3.60 & 0.01 & 0.00 & 0.00 & 0.01 & 0.00 & 0.00 & 0.00 \\ 0.00 & 3.60 & 0.01 & 0.00 & 0.00 & 0.01 & 0.01 & 0.00 \\ -0.1 & 0.01 & 3.60 & 0.01 & 0.00 & 0.00 & 0.01 & 0.00 \\ 0.00 & 0.01 & 0.00 & 3.60 & 0.00 & 0.01 & 0.00 & 0.00 \\ 0.00 & 0.01 & 0.00 & 0.01 & 3.60 & 0.01 & 0.00 & 0.00 \\ 0.00 & 0.01 & 0.00 & 0.01 & 0.01 & 3.60 & 0.00 & 0.00 \\ 0.00 & 0.01 & 0.00 & 0.01 & 0.01 & 0.00 & 3.60 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 00.0 & 0.01 & 3.60 \end{bmatrix}$$

$$\boldsymbol{Q}(\mathbf{2}) = 10^{-3} \times \begin{bmatrix} 3.60 & 0.01 & 0.00 & 0.00 & 0.01 & 0.00 & 0.00 & 0.00 \\ 0.00 & 3.60 & 0.01 & 0.00 & 0.00 & 0.01 & 0.01 & 0.00 \\ -0.1 & 0.01 & 3.60 & 0.01 & 0.00 & 0.00 & 0.01 & 0.00 \\ 0.00 & 0.01 & 0.00 & 3.60 & 0.00 & 0.01 & 0.00 & 0.00 \\ 0.00 & 0.01 & 0.00 & 0.01 & 3.60 & 0.01 & 0.00 & 0.00 \\ 0.00 & 0.01 & 0.00 & 0.01 & 0.01 & 3.60 & 0.00 & 0.00 \\ 0.00 & 0.01 & 0.00 & 0.01 & 0.01 & 0.00 & 3.60 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 00.0 & 0.01 & 3.60 \end{bmatrix}$$

$$\boldsymbol{Q}(\mathbf{3}) = 10^{-3} \times \begin{bmatrix} 3.60 & 0.01 & 0.00 & 0.00 & 0.01 & 0.00 & 0.00 & 0.00 \\ 0.00 & 3.60 & 0.01 & 0.00 & 0.00 & 0.01 & 0.01 & 0.00 \\ -0.1 & 0.01 & 3.60 & 0.01 & 0.00 & 0.00 & 0.01 & 0.00 \\ 0.00 & 0.01 & 0.00 & 3.60 & 0.00 & 0.01 & 0.00 & 0.00 \\ 0.00 & 0.01 & 0.00 & 0.01 & 3.60 & 0.01 & 0.00 & 0.00 \\ 0.00 & 0.01 & 0.00 & 0.01 & 0.01 & 3.60 & 0.00 & 0.00 \\ 0.00 & 0.01 & 0.00 & 0.01 & 0.01 & 0.00 & 3.60 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 00.0 & 0.01 & 3.60 \end{bmatrix}$$

Nilai dari matriks $\boldsymbol{Q}(\mathbf{1})$, $\boldsymbol{Q}(\mathbf{2})$, dan $\boldsymbol{Q}(\mathbf{3})$ merupakan matriks yang bersifat definit positif, sehingga memenuhi syarat matriks kovarians yang bisa dipakai pada Filter Kalman. Dengan cara serupa didapat nilai dari $\boldsymbol{R}(\mathbf{1})$, $\boldsymbol{R}(\mathbf{2})$, dan $\boldsymbol{R}(\mathbf{3})$ sebagai berikut:

$$\boldsymbol{R}(\mathbf{1}) = 10^{-3} \times \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}$$

$$\boldsymbol{R}(\mathbf{2}) = 10^{-3} \times \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}$$

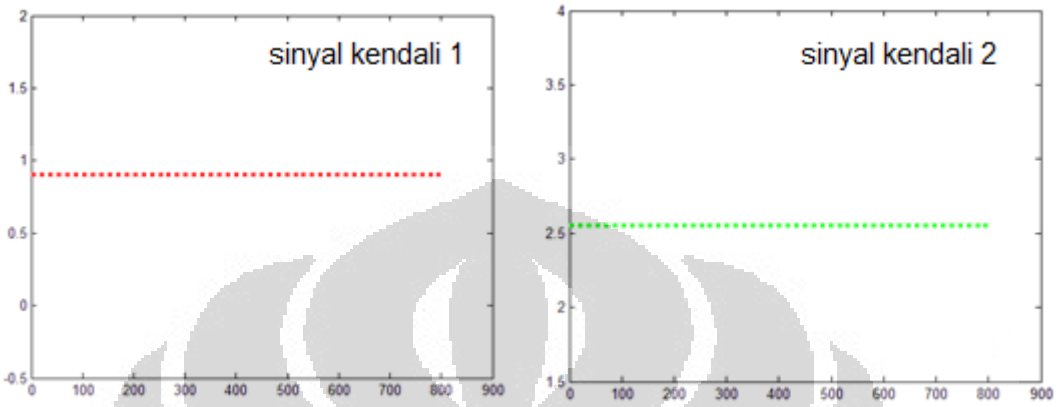
$$\boldsymbol{R}(\mathbf{3}) = 10^{-3} \times \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}$$

Matrik **R** yang dihasilkan sudah definit positif, sehingga memenuhi syarat sebagai matriks kovarian.

3.3.3 Pengujian Pada Sistem Tata Udara Presisi secara *Open Loop* dengan Sinyal Kendali Data Rekam

Untuk melihat apakah algoritma Filter Kalman yang dibuat dalam C-Mex sudah benar dan menghasilkan nilai yang sama dengan algoritma Filter Kalman yang dibuat dalam M-File, maka akan dibandingkan nilai keluaran *state* yang ada pada C-Mex dengan nilai keluaran *state* yang ada pada M-File. Untuk mendapatkan keluaran yang sama maka digunakan inputan yang sama pula antara algoritma Filter Kalman pada C-Mex dengan algoritma Filter Kalman pada M-

File. Nilai inputan yang digunakan berupa sejumlah data yang direkam dalam workspace yang nantinya dapat digunakan sebagai input dalam C-Mex S-Function maupun sebagi input dalam M-File.

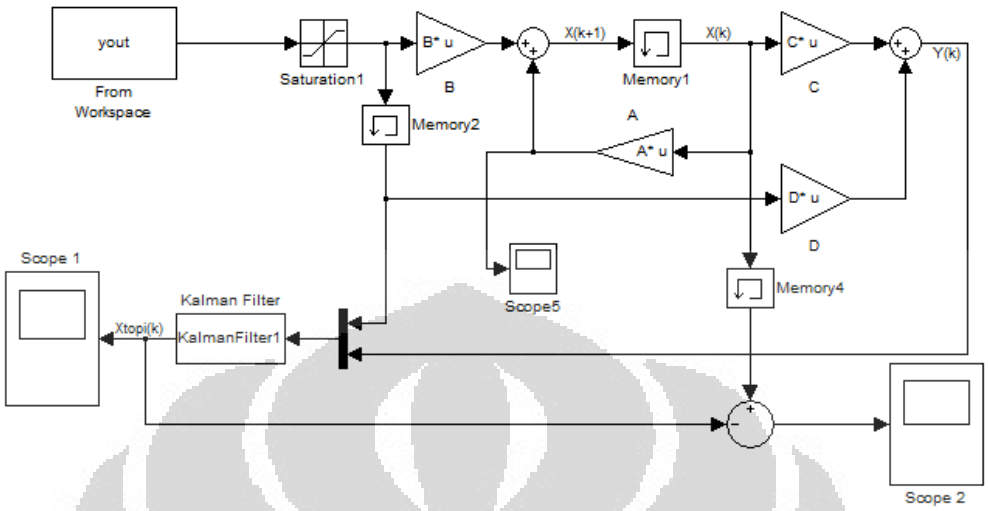


Gambar 3.2 Sinyal Input yang Diberikan untuk Pengujian Algoritma Filter Kalman

Sinyal input memiliki nilai konstan untuk u_1 sebesar 0.9 dan u_2 sebesar 2.55. Sinyal input yang digunakan mempunyai dimensi sebesar 800X2. Hal ini dikarenakan sinyal input yang dibutuhkan oleh algoritma Filter Kalman untuk memprediksi *state* pada sistem tata udara presisi memilii jumlah input dua. Dalam prosesnya, setiap satu baris input (dua kolom nilai) diambil sebagai masukan dalam algortima Filter Kalman baik dalam C-Mex maupun dalam M-File. Begitu seterusnya sampai baris input ke 800.

Selain pengujian secara *open loop* menggunakan sinyal kendali data rekam yang berupa sinyal konstan, dilakukan juga pengujian secara *open loop* menggunakan data rekam sinyal random, yang memiliki dimensi 2 X 800 dengan prosedur yang sama dengan sebelumnya.

Berikut gambar Simulink S-Function untuk C-Mex algoritma Filter Kalman.



Gambar 3.3 Blok Simulink S-Function Filter Kalman dengan Inputan Data Rekam

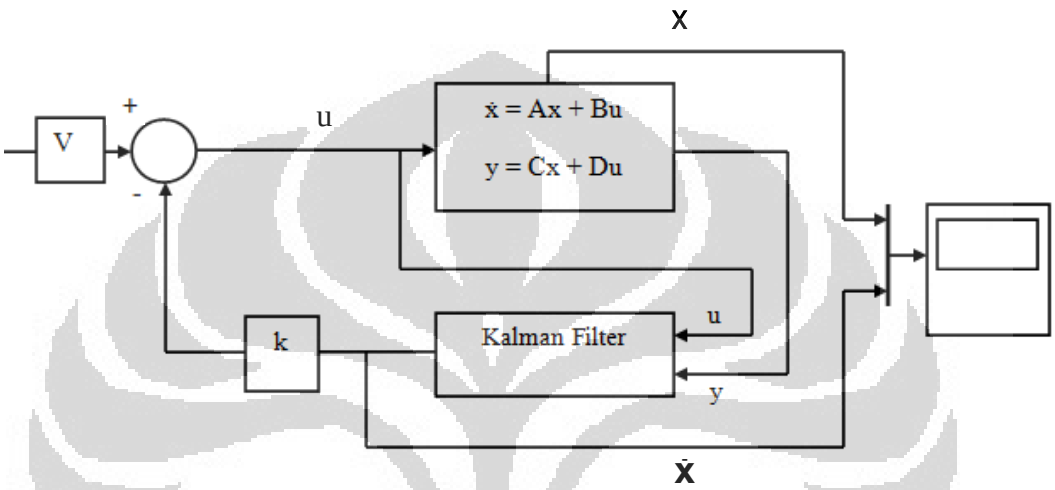
Sedangkan untuk input pada M-File, terlebih dahulu disimpan dalam yout.mat yang nantinya dengan fungsi load yout.mat, nilai dari inputan ini dapat digunakan dalam program M-File. Kedelapan *state* hasil keluaran dari M-File dan C-Mex kemudian dibandingkan untuk mengetahui kesamaan dari *state* hasil C-Mex dengan *state* hasil dari M-File.

Pemberian *gaussian noise* dalam M-File dilakukan dengan menggunakan persamaan $\sqrt{Sw}\ X\ sinyal\ random$ untuk noise proses dan $\sqrt{Sv}\ X\ sinyal\ random$ untuk noise pengukuran atau dapat diartikan sebagai akar kuadrat dari (Sw) atau (Sv) dikaliakan dengan sinyal random. Dimana Sw dan Sv diumpamakan sebagai besarnya nilai *spectral density* dari *gaussian noise* tersebut yang nilainya akan divariasikan untuk melihat kinerja dari algoritma filter kalman. Dalam penelitian ini nilai Sw dan Sv yang digunakan selalu sama dalam setiap proses estimasi. Hal ini dilakukan supaya mempermudah dalam proses variasi data *spectral density gaussian noise* dan supaya terlihat perbedaan setiap variasi yang dilakukan.

3.3.4 Pengujian Pada Sistem Tata Udara Presisi secara *Closed Loop* dengan Sinyal Kendali LQR

Pengujian selanjutnya dilakukan secara *cloosed loop* dengan menggunakan sinyal kendali LQR.

Diagram sistem pengendalian dan estimasi *state* ditunjukkan dalam gambar 3.4.



Gambar 3.4 Diagram Kendali Sistem Tata Udara Presisi Menggunakan LQR

Dari pengujian ini akan dibandingkan hasil estimasi *state* filter kalman dengan *state* aktual secara *cloosed loop* dengan berbagai variasi besarnya nilai *spectral density gaussian noise.* Selanjutnya dibandikan pula hasil keluaran estimasi dengan set point yang diberikan pada sistem tata udara presisi untuk mencapai suhu dan kelembaban pada nilai yang diinginkan.

BAB 4
HASIL SIMULASI DAN ANALISIS

Bab ini menjelaskan hasil simulasi yang didapat dari percobaan seperti yang telah dijelaskan pada Bab 3. Bab ini dibagi menjadi dua subbab. Subbab yang pertama adalah pengujian algoritma Filter Kalman pada sistem CSTR. Tujuan dari subbab ini adalah untuk mengetahui keakuratan Filter Kalman untuk mengestimasi *state* yang lebih sederhana, karena *state* pada CSTR hanya ada dua sedangkan pada sistem tata udara presisi ada delapan *state*. Dalam subbab ini juga akan dianalisa pengaruh penentuan matriks kovarian *error* proses dan pengukuran pada hasil estimasi *state*. Dibahas pula penerapan algoritma untuk mencari nilai kovarians matriks *error* proses dan pengukuran (**Q** dan **R)** yang diterapkan untuk mengestimasi *state* pada CSTR.

Pada subbab berikutnya, dianalisa estimasi *state* menggunakan Filter Kalman pada model sistem tata udara presisi. Penggunaan matriks **A**, **B**, **C**, dan **D** hasil identifikasi yang bervariasi akan semakin membuat estimasi *state* menggunakan Filter Kalman diuji kehandalannya. Dalam subbab ini juga akan dianalisa pengaruh penentuan matriks kovarian *error* proses dan pengukuran pada hasil estimasi *state*. Dibahas pula penerapan algoritma pencari nilai kovarian matriks *error* yang diterapkan untuk mengestimasi *state* pada sistem tata udara presisi dengan penggunaan *spectral density gaussian noise* yang divariasikan di berbagai nilai untuk mengetahui sejauh mana estimasi menggunakan algoritma Filter Kalman pada sistem tata udara presisi ini dapat bekerja dengan baik.

Pengujian dilakukan dengan memberikan nilai *spectral density gaussian noise* yang sama baik untuk *noise* proses maupun *noise* pengukuran untuk setiap kali pengujian yang dilakukan.

4.1.1 Pengujian Menggunakan Model CSTR *(Continuous Stirred Tank Reactor)* dengan Penentuan Nilai Matrik Q dan R Secara Manual

Pengujian menggunakan model CSTR dilakukan dengan berbagai variasi nilai matriks kovarians **Q** dan **R**, sebagai contoh untuk mengetahui nilai keluaran

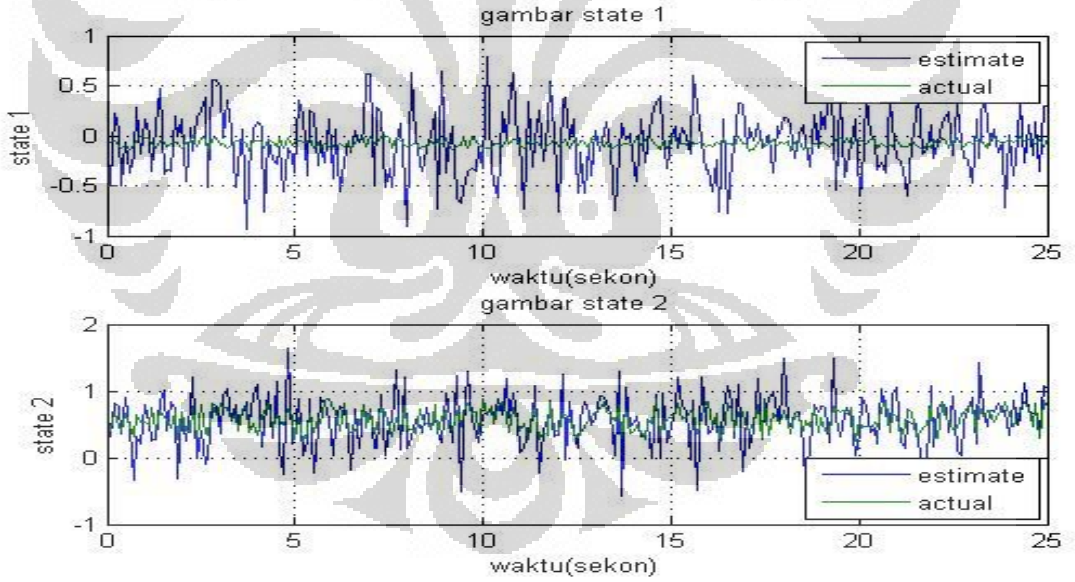
state dari hasil estimasi Filter Kalman dibandingkan dengan nilai *state* yang sebenarnya dengan menggunakan nilai kovarians *error* matriks

$$\boldsymbol{Q} = \begin{bmatrix} 0.002 & 0 \\ 0 & 0.002 \end{bmatrix} \text{ dan } \boldsymbol{R} = \begin{bmatrix} 0.002 & 0 \\ 0 & 0.002 \end{bmatrix}$$

Didapat nilai estimasi *state* dari algoritma Filter Kalman yang dibandingkan dengan *state* asli dari sistem CSTR. Nilai dari *spectral density gaussian noise*nya diset pada nilai 0.1 kemudian diperkecil sampai pada nilai 0.001, dan dengan sinyal kendali yang berupa sinyal random yang nilainya diantara 2 sampai 2.55.

4.1.1.1 Nilai *Spectral Density Gaussian Noise* 0.1

Dengan menggunakan nilai *spectral density gaussian noise* baik untuk *spectral density gaussian noise* proses maupun *spectral density gaussian noise* pengukuran diberi nilai yang sama yaitu masing-masing sebesar 0.1 didapat:



Gambar 4.1 *State* Prediksi dengan Penentuan Matriks Kovarian Secara Manual dan *State* Aktual dari Sistem CSTR dengan *Spectral Density Gaussian Noise* Sebesar 0.1

Untuk menganalisa keakuratan prediksi yang telah dilakukan, maka dapat digunakan komputasi nilai kuadrat kesalahan antara nilai prediksi dengan nilai aktual dari *state*nya.

Dengan nilai kuadrat kesalahan yang dicari dengan menggunakan rumus,

$$Error = \sum_{i=1}^{n} (y_2 - y_1)^2$$

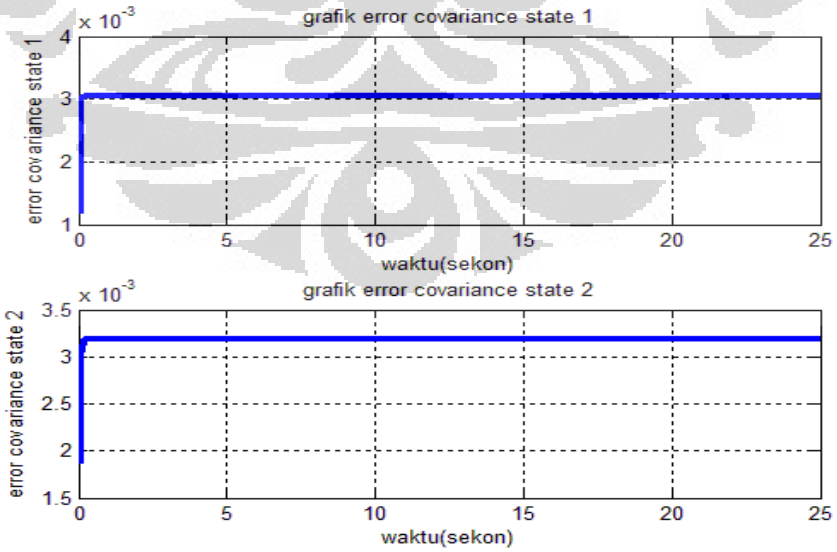
Dengan nilai dari y_2 merupakan nilai dari *state* aktual dan nilai dari y_1 merupakan nilai dari *state* estimasi Filter Kalman. Nilai kuadrat kesalahan masing-masing *state*nya sebesar :

Error kuadrat *state* 1	0.0071
Error kuadrat *state* 2	0.0059

Dari nilai kuadrat kesalahan (*error*) yang terjadi pada tiap *state* baik *state* pertama maupun *state* kedua menunjukkan nilai yang relatif lebih besar daripada nilai-nilai sebelumnya. Hal ini dikarenakan *spectral density gaussian noise* yang diberikan pada penerapan algoritma Filter Kalman ini adalah relatif besar, yaitu sebesar 0.1. Sehingga nilai *noise* yang begitu besar akan mengganggu proses prediksi pada algortima Filter Kalman.

Analisa kedua dapat dilihat dari grafik matriks **P** *(prediksi error covarians)* dari hasil estimasi Filter Kalman yang dikenai *spectral density gaussian noise* sebesar 0.1.

Grafik matriks **P** untuk *spectral density gaussian noise* sebesar 0.1 didapat sebagai berikut:



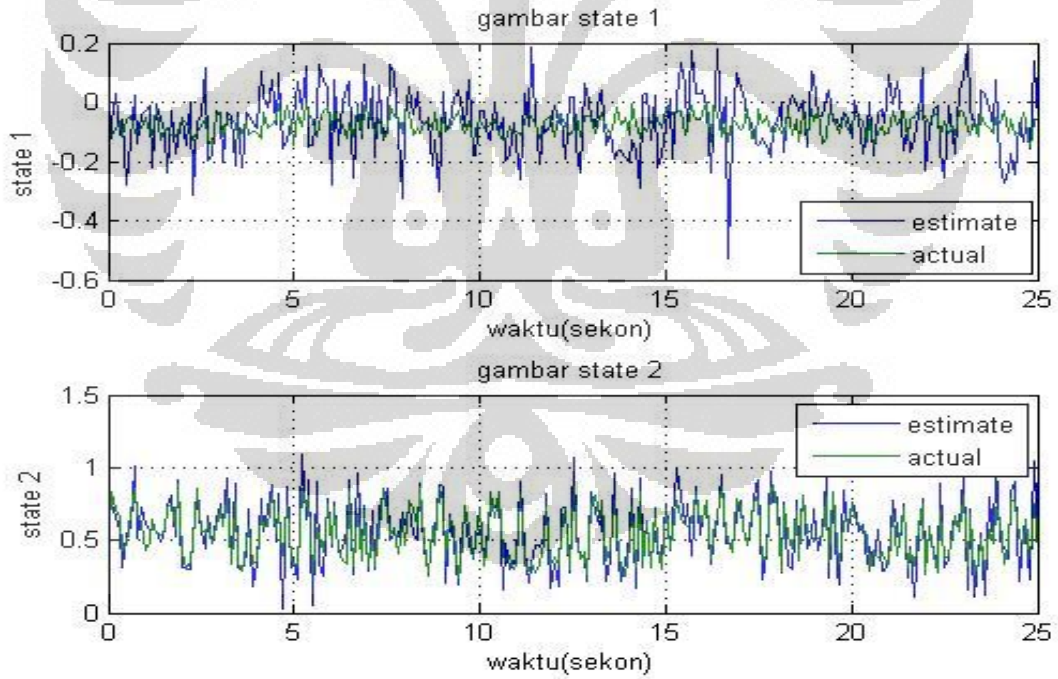
Gambar 4.2 Grafik dari Matriks P Sistem CSTR
dengan *Spectral Density Gaussian Noise* Sebesar 0.1

Dari grafik matriks **P** menunjukkan bahwa nilainya konvergen pada titik yang mendekati nilai nol untuk masing-masing prediksi *state* pada sistem CSTR. Sehingga penerapan algoritma Filter Kalman pada model sistem CSTR ini dapat dikatakan berhasil.

4.1.1.2 Nilai *Spectral Density Gaussian Noise* 0.01

Selanjutnya akan dilihat nilai estimasi Filter Kalman dengan mengubah *spectral density gaussian noise*nya menjadi 0.01 atau diperkecil. Nilai kovarian matriks *error* proses dan kovarian matriks *error* pengukuran yang digunakan masih sama seperti pada penelitian sebelumnya. Dari sini akan dibandingkan hasil estimasi untuk masing-masing *state* apabila *spectral density gaussian noise* nya diperkecil dan dilihat pengaruh dari pemberian variasi *noise*.

Grafik perbandingan antara *state* hasil estimasi dengan *state* sebenarnya untuk nilai *spectral density gaussian noise* 0.01 adalah sebagai berikut:



Gambar 4.3 *State* Prediksi dengan Penentuan Matriks Kovarian Secara Manual dan *State* Aktual dari Sistem CSTR dengan *Spectral Density Gaussian Noise* Sebesar 0.01

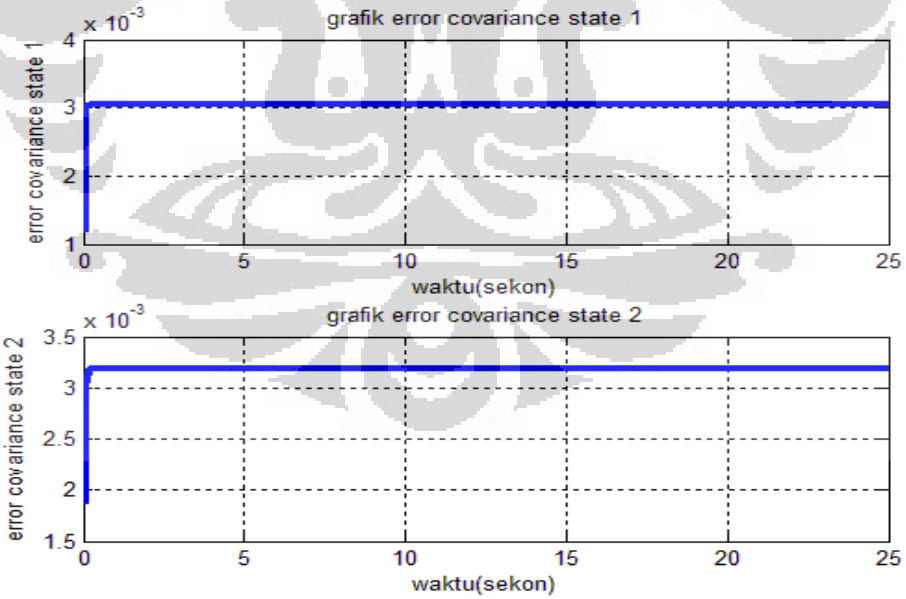
Dari grafik perbandingan tersebut, dihitung nilai kesalahan yang terjadi untuk masing-masing *state*. Dan didapat nilai kuadrat kesalahan masing-masing *state*nya sebesar :

Error kuadrat *state* 1	$1.6499\text{e} \times 10^{-5}$
Error kuadrat *state* 2	$3.6921\text{e} \times 10^{-4}$

Kuadrat kesalahan yang terjadi pada masing-masing *state* hasil estimasi menunjukkan nilai yang semakin mengecil dari percobaan sebelumnya yang menggunakan *spectral density gaussian noise* sebesar 0.1. Hal ini menunjukkan pula bahwa algoritma Filter Kalman mampu mengestimasi nilai *state* sistem CSTR meskipun dengan penentuan matriks *error* kovarian **Q** dan **R** secara manual/coba-coba. Besarnya **Q** dan **R** untuk penerapan di penelitian ini adalah sama dengan yang sebelumnya yaitu menggunakan:

$$\boldsymbol{Q} = \begin{bmatrix} 0.002 & 0 \\ 0 & 0.002 \end{bmatrix} \text{ dan } \boldsymbol{R} = \begin{bmatrix} 0.002 & 0 \\ 0 & 0.002 \end{bmatrix}.$$

Dengan besarnya matriks **P** (prediksi kovarians *error*) sebagai berikut:



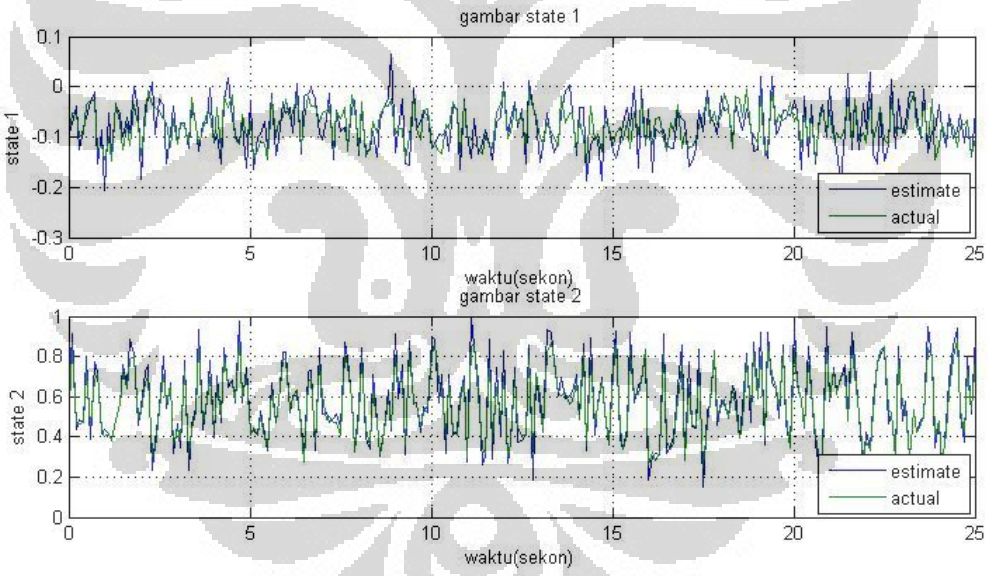
Gambar 4.4 Grafik dari Matriks P Sistem CSTR
dengan *Spectral Density Gaussian Noise* Sebesar 0.01

Grafik dari matriks **P** *(estimate error covariance)* menunjukkan nilai yang mengecil dan konvergen menuju nol, sehingga dapat diartikan bahwa estimasi *error* yang dilakukan oleh Filter Kalman ini benar dan memiliki tingkat kepercayaan yang tinggi. Karena *error* dari estimasi *state*nya semakin mengecil untuk setiap estimasi yang dilakukan.

4.1.1.3 Nilai *Spectral Density Gaussian Noise* 0.001

Untuk mengetahui nilai estimasi yang dihasilkan dapat lebih baik, maka nilai *spectral density gaussian noise*nya dikecilkan lagi menjadi 0.001.

Dengan menambahkan *spectral density gaussian noise* sebesar 0.001, diperoleh nilai hasil keluaran tersebut untuk setiap *state* yang dibandingkan dengan nilai aktual dari *state* CSTR seperti ditunjukkan pada gambar 4.5.



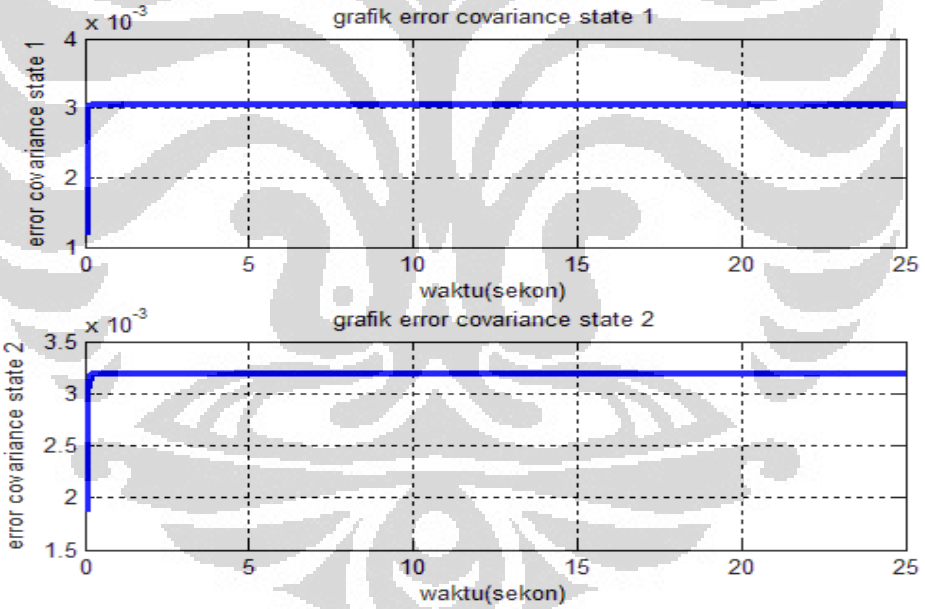
Gambar 4. 5 *State* Prediksi dengan Penentuan Matriks Kovarian Secara Manual dan *State* Aktual dari Sistem CSTR dengan *Spectral Density Gaussian Noise* Sebesar 0.001

Dan didapat nilai kuadrat kesalahan untuk masing-masing *state*nya sebesar:

Error kuadrat *state* 1	2.1449×10^{-4}
Error kuadrat *state* 2	9.2156×10^{-4}

Berdasarkan nilai *error* yang diperoleh pada estimasi *state* sistem CSTR dengan *spectral density gaussian noise* 0.001, menunjukkan nilai yang kecil. Hal ini disebabkan karena *noise* yang diberikan relatif kecil (lebih kecil dari percobaan sebelumnya) sehingga efek yang terjadi juga tidak begitu berpengaruh pada estimasi nilai *state*. Dalam estimasi sistem CSTR ini dapat dikatakan telah berhasil karena semakin diperkecil nilai *spectral density gaussian noise* nya, maka hasil estimasinya semakin baik.

Selanjutnya untuk menganalisa baik atau tidaknya Filter Kalman yang telah digunakan, dapat juga dilihat dari grafik estimasi *error* kovarians (matriks **P**). Filter Kalman yang baik adalah apabila pada grafik *error* kovarian yang didapat nilainya semakin konvergen dan akan semakin baik apabila nilainya mendekati nilai nol, atau konvergen pada nilai di dekat nol. Grafik dari matriks **P** *(prediksi error covarians)* dari sistem didapat sebagai berikut.



Gambar 4.6 Grafik dari Matriks P Sistem CSTR
dengan Spectral Density Gaussian Noise Sebesar 0.001

Dari grafik matriks P sistem CSTR dengan *spectral density gaussian noise* sebesar 0.001, menunjukkan bahwa grafik konvergen pada titik yang mendekati nol, yang menunjukkan bahwa *error* dari estimasi *state*nya semakin mengecil

untuk setiap estimasi yang dilakukan. Hal ini berarti hasil estimasi yang diperoleh memiliki tingkat kepercayaan yang tinggi.

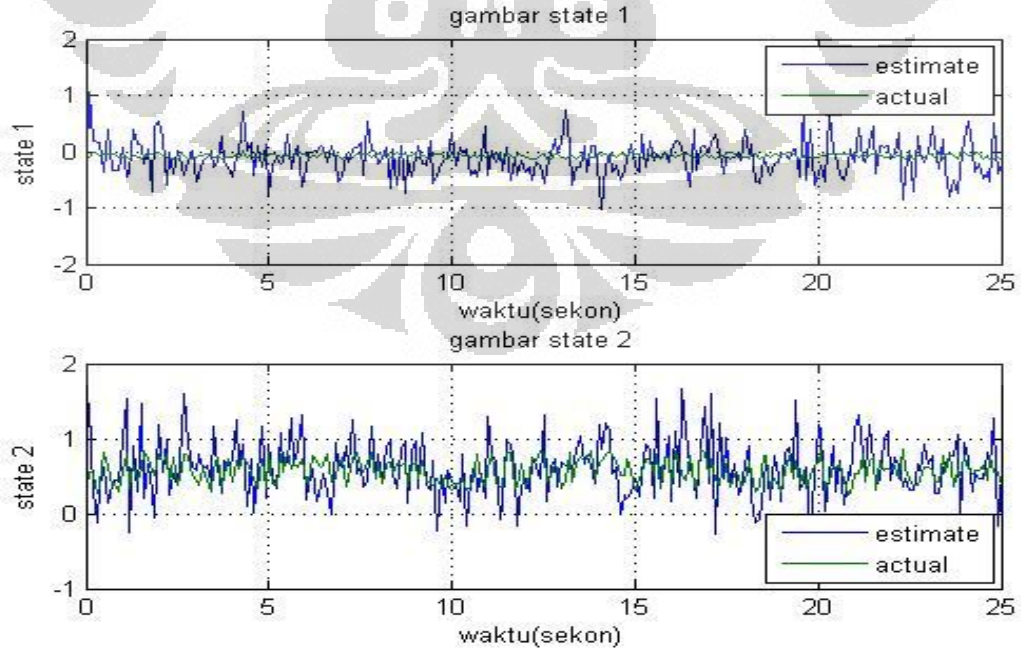
4.1.2 Penerapan Algoritma Pencari Nilai Optimasi Q dan R pada Sistem CSTR

Untuk penelitian selanjutnya digunakan matriks **Q**(3) dan **R**(3) hasil dari estimasi menggunakan algoritma estimasi kovarian *noise* sebagai kovarians matriks **Q** dan **R** untuk mengestimasi nilai *state* CSTR.

$$\boldsymbol{Q} = 10^{-3} \times \begin{bmatrix} 0.6 & -0.1 \\ -0.1 & 0.6 \end{bmatrix} \text{ dan } \boldsymbol{R} = 10^{-3} \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

4.1.2.1 Untuk Nilai *Spectral density gaussian noise* 0.1

Selanjutnya nilai dari optimasi matriks kovarian *error* proses ($\boldsymbol{Q}$) dan matriks kovarians *error* pengukuran ($\boldsymbol{R}$) tersebut diterapkan dalam algortima Filter Kalman dengan menggunakan *nilai spectral density gaussian noise* sebesar 0.1, dan didapat perbandingan *state* estimasi dengan *state* sebenernya sebagai berikut:



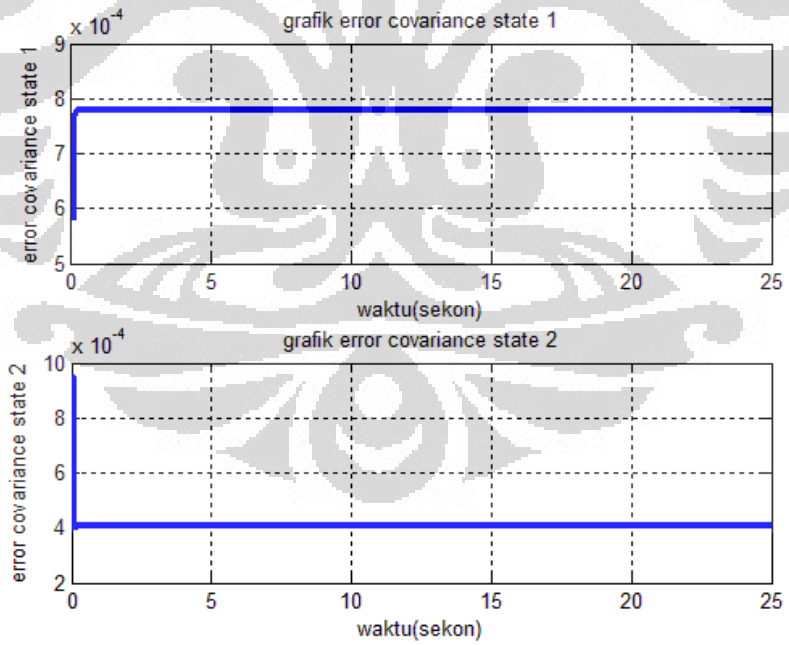
Gambar 4.7 *State* Prediksi dengan Optimasi Matriks Kovarian dan *State* Aktual dari Sistem CSTR dengan *Spectral Density Gaussian Noise* Sebesar 0.1

Analisis yang sama dilakukan dalam penelitian ini yaitu dengan mencari nilai kuadrat kesalahan yang terjadi antara nilai *state* hasil prediksi Filter Kalman dibandingkan dengan nilai *state* aktual dari model sistem CSTR.

Sehingga didapat nilai kuadrat kesalahan untuk masing-masing *state*nya adalah sebesar :

Error kuadrat *state* 1	3.4798×10^{-4}
Error kuadrat *state* 2	0.0039

Dari hasil perhitungan kuadrat kesalahan yang didapat, nilai kuadrat kesalahannya lebih kecil daripada penelitian sebelumnya yang menggunakan metode manual untuk menentukan nilai kovarian matriks *error* proses maupun nilai kovarian matriks *error* pengukuran. Sehingga algoritma optimasi nilai kovarian ini dapat diterapkan. Untuk mengetahui lebih jauh, maka dapat dianalisa mengenai matriks **P** *(estimate prediction error)*nya. Didapat grafik matriks **P** nya adalah sebagai berikut:



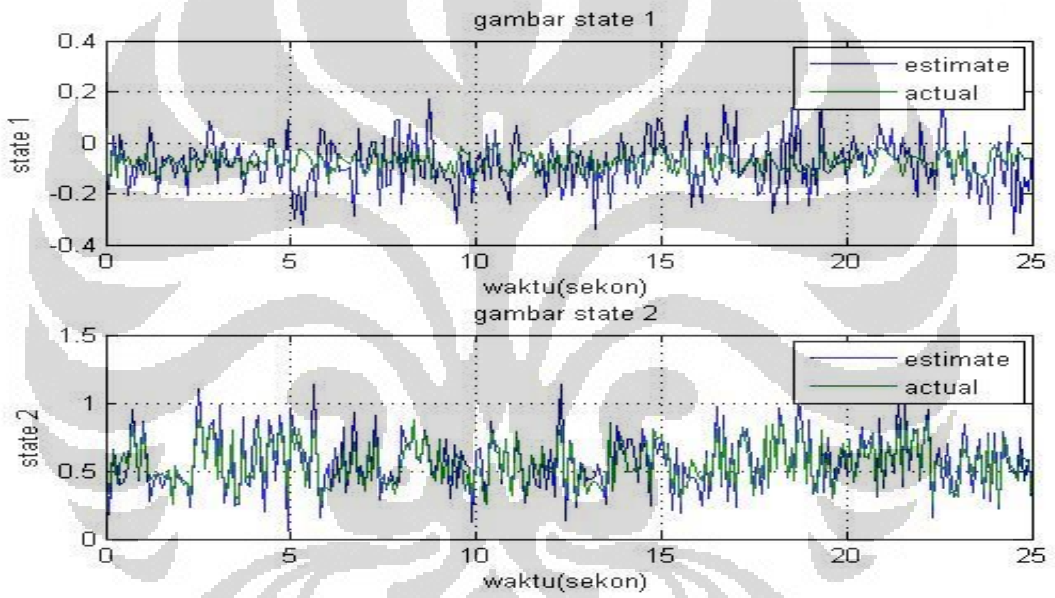
Gambar 4.8 Grafik dari Matriks P Sistem CSTR dengan *Spectral Density Gaussian Noise* Sebesar 0.1 dengan Optimasi Matriks Kovarian

Dari grafik matriks **P** yang dihasilkan, menunjukkan bahwa nilainya konvergen di titik yang sangat kecil atau mendekati nol. Sehingga dapat dikatakan

bahwa estimasi yang dilakukan benar karena *error* dari estimasi *state*nya semakin mengecil untuk setiap estimasi. Nilai saat mencapai konvergennya lebih cepat dibandingkan dengan matriks **P** yang dihasilkan dengan menggunakan penentuan nilai **Q** dan **R** secara manual.

4.1.2.2 Untuk Nilai *Spectral Density Gaussian Noise* 0.01

Selanjutnya nilai *spectral density gaussian noise*nya dikecilkan menjadi 0.01. Dari hasil estimasi didapat *state* estimasi yang dibandingkan dengan *state* aktualnya adalah sebagai berikut:



Gambar 4.9 *State* Prediksi dengan Optimasi Matriks Kovarian dan *State* Aktual dari Sistem CSTR dengan *Spectral Density Gaussian Noise* Sebesar 0.01

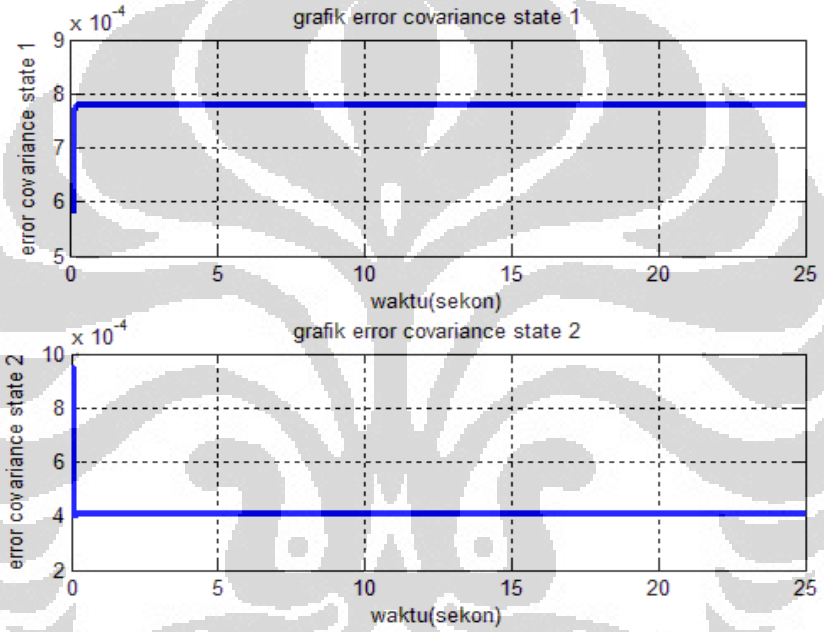
Dari hasil perhitungan nilai kuadrat kesalahan, didapatkan nilai kuadrat kesalahan untuk masing-masing *state*nya sebesar :

Error kuadrat *state* 1	1.3921×10^{-4}
Error kuadrat *state* 2	9.5457×10^{-5}

Kuadrat kesalahan yang terjadi menunjukkan nilai yang kecil untuk pemberian *spectral density gaussian noise* sebesar 0.01. Nilai kuadrat kesalahan

pada percobaan dengan *spectral density gaussian noise* sebesar 0.01 ini nilainya juga lebih kecil daripada percobaan sebelumnya yang penentuan nilai matriks kovarian *error* proses dan kovarian *error* pengukurannya secara manual yang sama-sama menggunakan *spectral density gaussian noise* sebesar 0.01. Untuk nilai *spectral density gaussian noise* yang lebih kecil, algoritma optimasi nilai kovarian ini menunjukkan kinerja yang lebih baik dalam pengestimasian *state*. Selanjutnya dilihat lagi grafik matriks **P** yang dihasilkan untuk mengetahui nilai dari estimasi kovarian *error*.

Dengan matriks **P** yang dihasilkan adalah sebagai berikut:



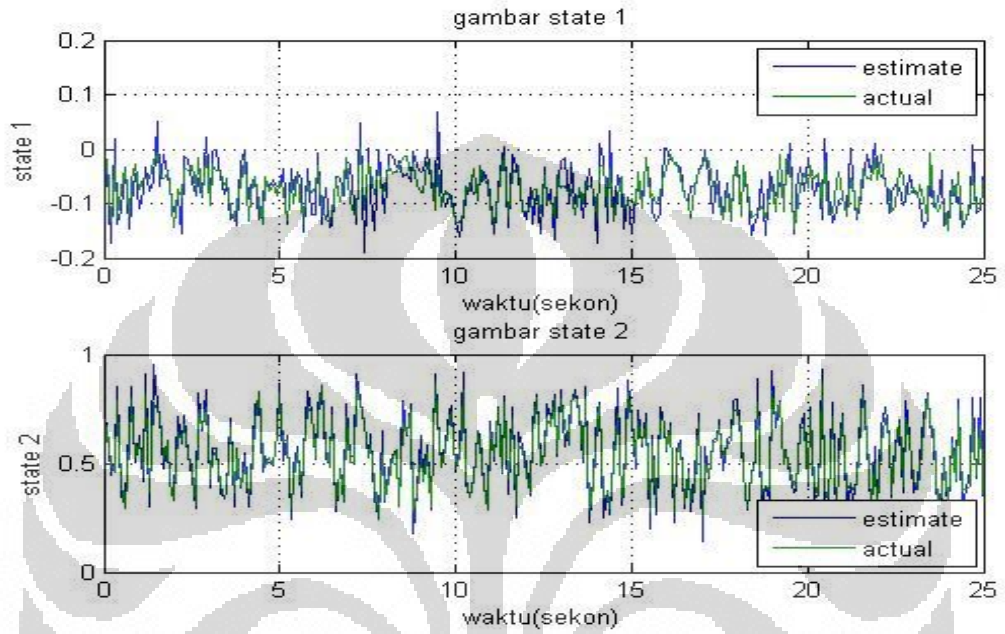
Gambar 4.10 Grafik dari Matriks P Sistem CSTR dengan *Spectral density Gaussian Noise Sebesar* 0.01 dengan Optimasi Matriks Kovarian

Dari grafik matriks **P** sistem CSTR dengan *spectral density gaussian noise* sebesar 0.01, menunjukkan bahwa grafik konvergen pada titik yang mendekati nol, hal ini berarti hasil estimasi yang diperoleh memiliki tingkat kepercayaan yang tinggi dan proses estimasi Filter Kalman yang sudah benar.

4.1.2.3 Untuk Nilai *Spectral Density Gaussian Noise* 0.001

Pengujian dilakukan lagi dengan memperkecil nilai *spectral density gaussian noise* menjadi 0.001. Hal ini juga dilakukan selain untuk mengetahui pengaruh pengurangan nilai *spectral density gaussian noise,* hal ini juga dilakukan

untuk mengetahui kinerja algoritma Filter Kalman dalam memprediksi *state* dengan menggunakan algoritma penentuan nilai kovarian untuk menghasilkan optimasi nilai matriks **Q** dan **R**, dibandingkan dengan penentuan matriks **Q** dan **R** secara manual/coba-coba. *State* yang dihasilkan dengan menggunakan *spectral density gaussian noise* sebesar 0.001 ditunjukkan pada gambar 4.11.



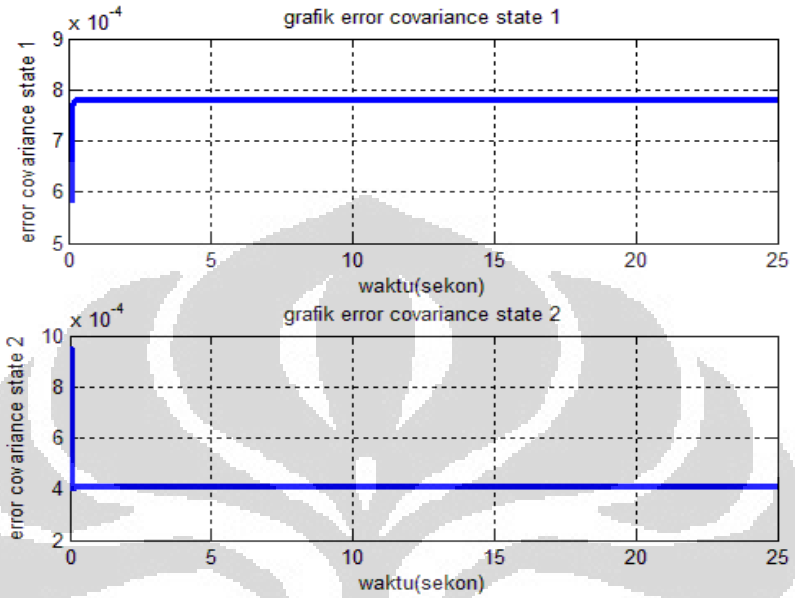
Gambar 4.11 *State* Prediksi dengan Optimasi Matriks Kovarian dan *State* Aktual dari Sistem CSTR dengan *Spectral Density Gaussian Noise* Sebesar 0.001

Didapat nilai kuadrat kesalahan masing-masing *state*nya sebesar :

Error kuadrat *state* 1	1.9602×10^{-5}
Error kuadrat *state* 2	2.1844×10^{-5}

Dari hasil perhitungan kesalahan yang terjadi terhadap hasil estimasi tiap *state* menunjukkan bahwa kesalahan yang terjadi relatif kecil. Hal ini disebabkan karena besarnya *spectral density gaussian noise* yang diberikan pada penerapan algoritma Filter Kalman ini juga relatif kecil. Sehingga *state* prediksinya cenderung mirip dengan *state* sebenarnya dari sistem CSTR. Semakin kecil nilai *spectral density gaussian noise* yang diberikan maka nilai *state* estimasinya akan semakin baik atau mendekati nilai *state* sebenarnya. Hasil estimasi Filter Kalman dengan *spectral density gaussian noise* 0.001 dengan nilai kovarian matriks **Q** dan

R yang dicari menggunakan algoritma optimasi menunjukkan kinerja yang lebih baik daripada menggunakan nilai kovarian **Q** dan **R** secara manual dengan *spectral density gaussian noise* yang sama. Untuk analisa yang lebih lanjut dapat dilihat dari matrik **P** pada gambar 4.12.



Gambar 4.12 Grafik dari Matriks P Sistem CSTR dengan *Spectral Density Gaussian Noise* Sebesar 0.001 dengan Optimasi Matriks Kovarian

Berdasarkan grafik matriks **P** untuk model sistem CSTR dengan *spectral density gaussian noise* 0.001, untuk setiap *state*nya menunjukkan konvergen pada nilai yang sangat kecil yaitu $7.78\,X\,10^{-4}$ dan $3.98\,X\,10^{-4}$. Dari hasil tersebut dapat dianalisa bahwa proses estimasi memiliki tingkat kepercayaan yang tinggi karena nilai matriks **P** nya yang konvergen pada nilai yang mendekati nol yang berarti *error* estimasi *state*nya semakin mengecil untuk setiap estimasi.

4.1.3 Variasi Q dan R untuk Model Sistem CSTR dengan Berbagai Nilai *Spectral Density Gaussian Noise*

Pengujian juga dilakukan dengan menggunakan nilai kovarian matriks *error* proses (**Q**) dan kovarian matriks *error* pengukuran (**R**) dengan variasi nilai yang berbeda-beda. Penggunaan algoritma optimasi nilai kovarian **Q** dan **R** juga akan dibandingkan kinerjanya dengan penentuan nilai matriks **Q** dan **R** secara manual/coba-coba. Secara umum, pengaruh penentuan nilai matriks kovarian **Q**

dan **R** dengan berbagai variasi nilai yang dibagi menjadi tiga bagian berdasarkan pada besarnya *spectral density gaussian noise* yang diberikan pada sistem, hal tersebut disajikan dalam tabel 4.1, 4.2, dan 4.3.

Tabel 4.1 Variasi Matriks **Q** dan **R** untuk Sistem CSTR dengan *Spectral Density Gaussian Noise* 0.1

No.	**Kovarians Q**	**Kovarians R**	**Kuadrat** *Error* ***State* 1**	**Kuadrat** *Error* ***State* 2**
1.	$10^{-3} \times \begin{bmatrix} 0.6 & -0.1 \\ -0.1 & 0.6 \end{bmatrix}$	$10^{-3} \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	3.4798×10^{-4}	0.0039 *
2.	$\begin{bmatrix} 0.0002 & 0 \\ 0 & 0.0002 \end{bmatrix}$	$\begin{bmatrix} 0.0002 & 0 \\ 0 & 0.0002 \end{bmatrix}$	0.0126	0.0241
3.	$\begin{bmatrix} 0.002 & 0 \\ 0 & 0.002 \end{bmatrix}$	$\begin{bmatrix} 0.002 & 0 \\ 0 & 0.002 \end{bmatrix}$	0.0071	0.0059
4.	$\begin{bmatrix} 0.02 & 0 \\ 0 & 0.02 \end{bmatrix}$	$\begin{bmatrix} 0.02 & 0 \\ 0 & 0.02 \end{bmatrix}$	0.0268	0.0078
5.	$\begin{bmatrix} 0.2 & 0 \\ 0 & 0.2 \end{bmatrix}$	$\begin{bmatrix} 0.2 & 0 \\ 0 & 0.2 \end{bmatrix}$	0.0202	0.0186

* dengan penentuan menggunakan algoritma optimasi matriks kovarian

Tabel 4.2 Variasi Matriks **Q** dan **R** untuk Sistem CSTR dengan *Spectral Density Gaussian Noise* 0.01

No	**Kovarians Q**	**Kovarians R**	**Kuadrat** *Error* ***State* 1**	**Kuadrat** *Error* ***State* 2**
1.	$10^{-3} \times \begin{bmatrix} 0.6 & -0.1 \\ -0.1 & 0.6 \end{bmatrix}$	$10^{-3} \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	1.3921×10^{-4}	9.5457×10^{-5}*
2.	$\begin{bmatrix} 0.0002 & 0 \\ 0 & 0.0002 \end{bmatrix}$	$\begin{bmatrix} 0.0002 & 0 \\ 0 & 0.0002 \end{bmatrix}$	7.345×10^{-4}	5.247×10^{-4}
3.	$\begin{bmatrix} 0.002 & 0 \\ 0 & 0.002 \end{bmatrix}$	$\begin{bmatrix} 0.002 & 0 \\ 0 & 0.002 \end{bmatrix}$	1.6499×10^{-5}	3.6921×10^{-4}
4.	$\begin{bmatrix} 0.02 & 0 \\ 0 & 0.02 \end{bmatrix}$	$\begin{bmatrix} 0.02 & 0 \\ 0 & 0.02 \end{bmatrix}$	7.6636×10^{-4}	7.1637×10^{-4}
5.	$\begin{bmatrix} 0.2 & 0 \\ 0 & 0.2 \end{bmatrix}$	$\begin{bmatrix} 0.2 & 0 \\ 0 & 0.2 \end{bmatrix}$	1.3153×10^{-4}	0.0022

* dengan penentuan menggunakan algoritma optimasi matriks kovarian

Tabel 4.3 Variasi Matriks **Q** dan **R** untuk Sistem CSTR dengan *Spectral Density Gaussian Noise* 0.001

No	**Kovarians Q**	**Kovarians R**	**Kuadrat** *Error* ***State* 1**	**Kuadrat** *Error* ***State* 2**
1.	$10^{-3} \times \begin{bmatrix} 0.6 & -0.1 \\ -0.1 & 0.6 \end{bmatrix}$	$10^{-3} \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	1.9602×10^{-5}	2.184×10^{-5}*
2.	$\begin{bmatrix} 0.0002 & 0 \\ 0 & 0.0002 \end{bmatrix}$	$\begin{bmatrix} 0.0002 & 0 \\ 0 & 0.0002 \end{bmatrix}$	7.9039×10^{-5}	5.2328×10^{-5}
3.	$\begin{bmatrix} 0.002 & 0 \\ 0 & 0.002 \end{bmatrix}$	$\begin{bmatrix} 0.002 & 0 \\ 0 & 0.002 \end{bmatrix}$	2.1449×10^{-4}	9.2156×10^{-4}
4.	$\begin{bmatrix} 0.02 & 0 \\ 0 & 0.02 \end{bmatrix}$	$\begin{bmatrix} 0.02 & 0 \\ 0 & 0.02 \end{bmatrix}$	2.3816×10^{-4}	3.0887×10^{-5}
5.	$\begin{bmatrix} 0.2 & 0 \\ 0 & 0.2 \end{bmatrix}$	$\begin{bmatrix} 0.2 & 0 \\ 0 & 0.2 \end{bmatrix}$	8.1666×10^{-5}	3.3439×10^{-5}

* dengan penentuan menggunakan algoritma optimasi matriks kovarian

Secara umum, besarnya *noise* mempengaruhi kinerja dari estimator Filter Kalman untuk mengestimasi nilai *state* pada sistem. Semakin besar *spectral density gaussian noise* yang diberikan, maka akan semakin besar kesalahan yang terjadi. Pada penelitian pada model sistem CSTR ini, penerapan algoritma penentuan nilai kovarian untuk menentukan nilai matriks **Q** dan **R** yang diaplikasikan dalam penelitian ini berhasil, dapat dilihat dari nilai kesalahan yang cenderung paling kecil dibandingkan dengan penentuan nilai matriks kovarian **Q** dan **R** secara manual/coba-coba.

4.2 Pengujian Menggunakan Model Sistem Tata Udara Presisi

Estimasi menggunakan algoritma Filter Kalman selanjutnya diujikan pada sistem tata udara presisi. Dalam pengujian ini sinyal input yang diberikan tidak lagi sinyal random melainkan menggunakan sinyal konstan seperti yang sudah dibahas pada bab sebelumnya. Nilai kovarian matriks **Q** dan **R** ditentukan menggunakan algoritma optimasi matriks kovarian. Pengujian ini dilakukan menggunakan C-Mex dan M-File pada Matlab.

4.2.1 Membandingkan Keluaran *State* pada M-File dengan Keluaran *State* pada C-Mex

Kedelapan *state* hasil keluaran dari M-File dan C-Mex kemudian dibandingkan untuk mengetahui kesamaan dari *state* hasil C-Mex dengan *state* hasil dari M-File.

Untuk mengetahui besarnya perbedaan yang terjadi dapat dicari dengan menggunakan rumus *error* dengan menganggap salah satu keluaran *state* (baik dari C-Mex maupun dari M-File) merupakan *state* yang ideal. Besarnya *error* untuk setiap *state*nya adalah dihitung dengan menggunakan persamaan kuadrat kesalahan dengan persamaan $Error = \frac{1}{N} X \sum_{i=1}^{n} (y_{C-Mex} - y_{M-File})^2$ dan didapat besarnya *error* tiap *state*nya adalah sebagai berikut:

Tabel 4.4 Nilai Kuadrat Kesalahan Estimasi *State* Filter Kalman dengan M-File dibandingkan dengan *State* Estimasi *State* Filter Kalman dengan C-Mex Sistem Tata Udara Presisi

No. *State*	Besarnya Kuadrat Kesalahan
State 1	$1.8646 X 10^{-23}$
State 2	$3.6466 X 10^{-23}$
State 3	$1.1208 X 10^{-24}$
State 4	$2.7958 X 10^{-24}$
State 5	$9.2414 X 10^{-23}$
State 6	$4.8292 X 10^{-23}$
State 7	$3.9010 X 10^{-23}$
State 8	$9.7497 X 10^{-25}$

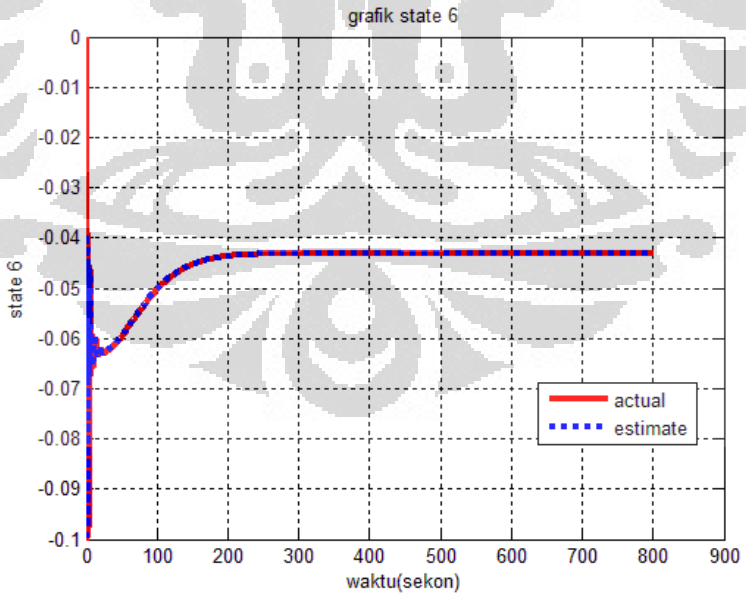
Dari hasil perhitungan perbedaan antara *state* keluaran dari C-Mex dibandingkan dengan *state* keluaran dari M-File menunjukkan sedikit sekali perbedaan yang terjadi antara dua metode ini yang mungkin dikarenakan karena perbedaan proses komputasi antara C-Mex dengan M-File. Dengan kata lain dapat dikatan bahwa nilai keluaran dari C-Mex sama dengan nilai keluaran dari M-File. Sehingga algoritma yang ditulis di dalam C-Mex sudah sama dengan algoritma yang ditulis di dalam M-File.

4.2.2 Hasil Estimasi Filter Kalman dengan Variasi Nilai *Spectral Density Gaussian Noise* secara *Open Loop* dengan Sinyal Kendali Data Rekam Sinyal Konstan

4.2.2.1 Tanpa Menggunakan *Noise*

Langkah selanjutnya adalah membandingkan nilai keluaran *state* dari M-File dengan nilai *state* sebenarnya dengan berbagai variasi *spectral density gaussian noise* dengan menggunakan sinyal kendali berupa data rekam sinyal konstan. Untuk mengidentifikasi nilai *state* sebenarnya dilakukan dengan menggunakan persamaan umum ruang keadaan $x(k+1) = Ax(k) + Bu(k)$ dan $y(k) = Cx(k) + Du(k)$, kemudian mencari nilai dari *state* x dari persamaan ruang keadaan tersebut. Untuk membandingkan nilai estimasi hasil Filter Kalman dengan nilai *state* sebenarnya, digunakan nilai keluaran *state* dari C-Mex Filter.Kalman tanpa menggunakan variasi *gaussian noise*.

State hasil estimasi dibandingkan dengan *state* sebenarnya yang didapat dari sistem tata udara presisi. Berikut ini merupakan *state* keenam dari sistem tata udara presisi yang ditampilkan untuk analisa. Grafik perbandingan *state* yang lain secara keseluruhan ada dalam halaman Lampiran.



Gambar 4.13 Grafik Perbandingan *State* Keenam Estimasi Filter Kalman dengan *State* Sebenarnya pada Sistem Tata Udara Presisi

Berdasarkan data perhitungan kuadrat kesalahan, dapat dianalisa bahwa hasil estimasi *state* menggunakan Filter Kalman tanpa *noise* memiliki nilai yang sama dengan *state* sebenarnya sistem tata udara presisi, hal ini dikarenakan kuadrat kesalahan yang terjadi sangat kecil bahkan sangat mendekati nol. Sehingga dapat dikatakan bahwa estimasi filter kalman berjalan dengan baik.

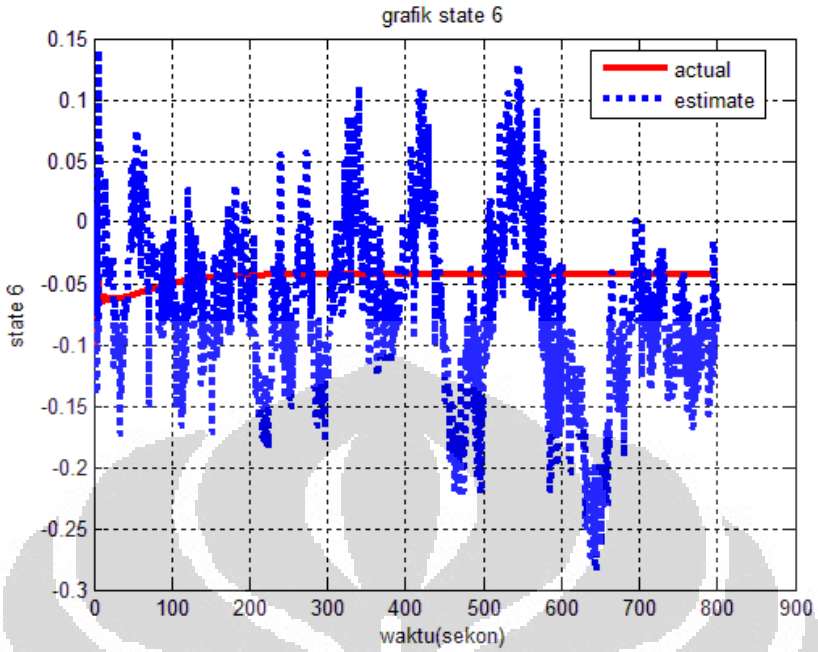
Untuk mengetahui besarnya perbedaan yang terjadi dapat dicari dengan menggunakan rumus *error* dengan menganggap salah satu keluaran *state* (baik dari C-Mex maupun dari M-File) merupakan *state* yang ideal. Besarnya *error* untuk setiap *state*nya adalah dihitung dengan menggunakan persamaan kuadrat kesalaha $Error = \frac{1}{N} \text{X} \sum_{i=1}^{n} (y_{C-Mex} - y_{M-File})^2$ dan didapat besarnya *error* tiap *state*nya adalah sebagai berikut:

Tabel 4.5 Nilai Kuadrat Kesalahan Estimasi *State* Filter Kalman Menggunakan *Spectral Density Gaussian Noise* Sebesar Nol dengan Nilai *State* Sebenarnya dari Sistem Tata Udara Presisi

No. *State*	Besarnya Kuadrat Kesalahan
State 1	$3.3862 \text{ X } 10^{-34}$
State 2	$2.6201 \text{ X } 10^{-33}$
State 3	$5.1166 \text{ X } 10^{-34}$
State 4	$3.7904 \text{ X } 10^{-34}$
State 5	$3.2410 \text{ X } 10^{-34}$
State 6	$1.2194 \text{ X } 10^{-34}$
State 7	$7.4148 \text{ X } 10^{-35}$
State 8	$4.6463 \text{ X } 10^{-35}$

Analisa selanjutnya dilakukan dengan memvariasikan nilai *spectral density gaussian noise* dengan berbagai variasi nilai, mulai dari 0.1, 0.001, dan seterusnya. Untuk hasil estimasi masing-masing *state* sistem tata udara presisi, dapat dilihat dalam lampiran.

4.2.2.2 Menggunakan *Spectral Density Gaussian Noise* Sebesar 10^{-1}



Gambar 4.14 Grafik Perbandingan *State* Keenam Estimasi Filter Kalman dengan *State* Sebenarnya pada Sistem Tata Udara Presisi dengan *Spectral Density Gaussian Noise* Sebesar 10^{-1}

Berdasarkan hasil perhitungan, didapat nilai dari kuadrat kesalahan urntuk masing-masing *state* dengan penambahan spectral density gaussian noise sebesar 0.1 adalah sebagai berikut:

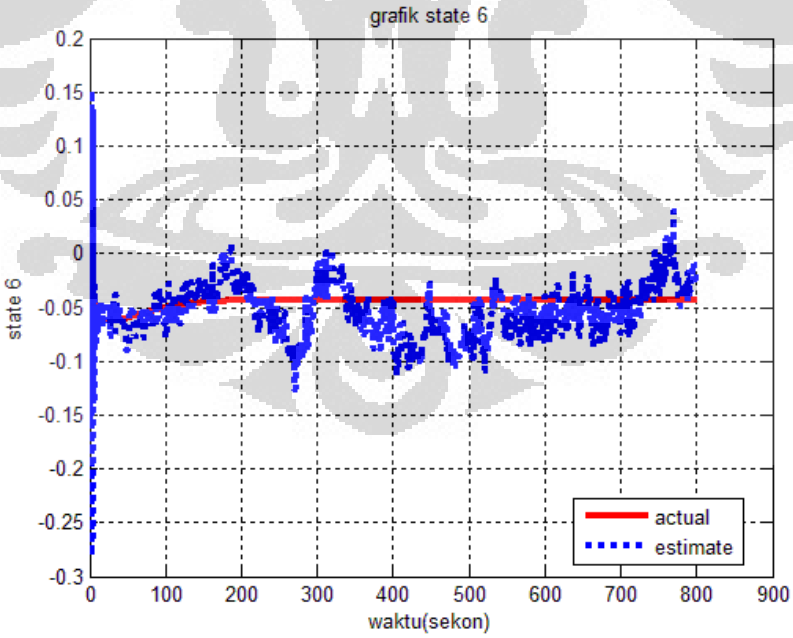
Tabel 4.6 Nilai Kuadrat Kesalahan Estimasi *State* Filter Kalman Menggunakan *Spectral Density Gaussian Noise* Sebesar 10^{-1} dengan Nilai *State* Sebenarnya dari Sistem Tata Udara Presisi

No. *State*	Besarnya Kuadrat Kesalahan
State 1	1.8595
State 2	0.2600
State 3	3.1944
State 4	0.0162
State 5	0.0092
State 6	0.0059
State 7	0.0100
State 8	0.0024

Dari hasil analisa perhitungan kuadrat kesalahan, didapatkan nilai kuadrat kesalahan yang cukup besar terutama untuk *state* pertama dan ketiga yang menunjukkan nilai kuadrat kesalahan di atas satu. Hal ini disebabkan karena *noise* yang diberikan pada proses prediksi *state* cukup besar pula, sehingga akan mempengaruhi kinerja algoritma Filter Kalman dalam memprediksi nilai *state*. Meskipun nilai kudrat kesalahan yang ditimbulkan memiliki nilai yang besar, dari grafik perbandingan antara nilai *state* prediksi dengan *state* aktual terlihat bahwa *state* prediksi Filter Kalman arahnya masih mengikuti *state* aktual dari sistem tata udara presisi meskipun memiliki amplitudo nilai yang cukup besar.

Untuk mengetahui kinerja algoritma Filter Kalman dalam memprediksi *state* sistem tata udara presisi, selanjutkan nilai *spectral density gaussian noise* yang diberikan akan diperkecil untuk dapat mengetahui baik atau tidaknya kinerja Filter Kalman dalam memprediksi *state* sistem tata udara presisi ini untuk pemberian *noise* yang diperkecil. Grafik hasil prediksi *state-state* sistem tata udara presisi yang lain, lebih lengkapnya disajikan dalam halaman Lampiran.

4.2.2.3 Menggunakan *Spectral Density Gaussian Noise* Sebesar 10^{-2}



Gambar 4.15 Grafik Perbandingan *State* Keenam Estimasi Filter Kalman dengan *State* Sebenarnya pada Sistem Tata Udara Presisi dengan *Spectral Density Gaussian Noise* Sebesar 10^{-2}

Berdasarkan hasil perhitungan, didapat nilai dari kuadrat kesalahan urntuk masing-masing *state* dengan penambahan *spectral density gaussian noise* sebesar 0.01 adalah sebagai berikut:

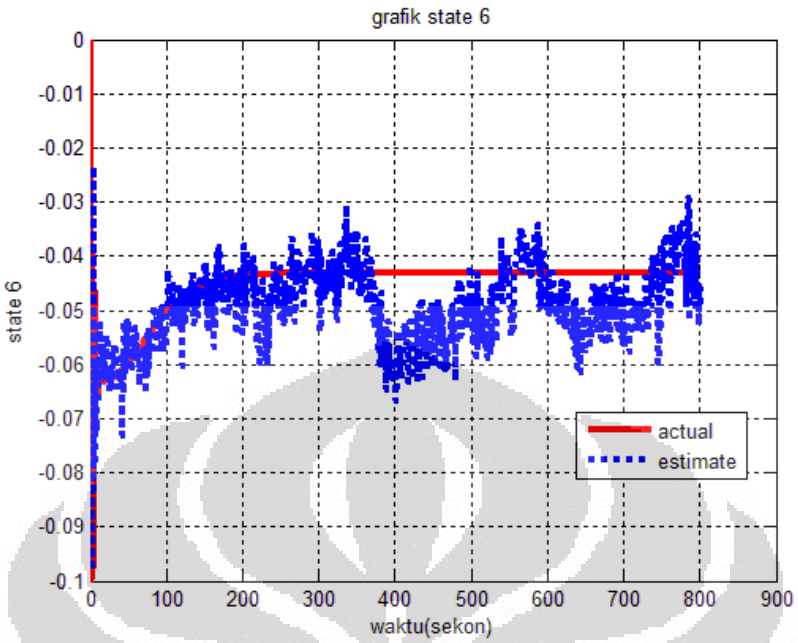
Tabel 4.7 Nilai Kuadrat Kesalahan Estimasi *State* Filter Kalman Menggunakan *Spectral Density Gaussian Noise* Sebesar 10^{-2} dengan Nilai *State* Sebenarnya dari Sistem Tata Udara Presisi

No. *State*	Besarnya Kuadrat Kesalahan
State 1	0.1819
State 2	0.0289
State 3	0.1992
State 4	0.0020
State 5	0.0013
State 6	$7.1470 \text{ X } 10^{-4}$
State 7	$8.7834 \text{ X } 10^{-4}$
State 8	$1.6578 \text{ X } 10^{-4}$

Setelah nilai *spectral density gaussian noise*nya diperkecil menjadi 10^{-2} terlihat nilai kuadrat kesalahan yang dihasilkan menjadi lebih kecil daripada nilai kesalahan untuk percobaan sebelumnya. Bahkan untuk nilai kesalahan untuk *state* keenam, ketujuh, dan kedelapan menunjukkan nilai kuadrat kesalahan yang relatif sangat kecil sehingga dapat dikatakan nilai prediksi *state* keenam, *state* ketujuh, dan *state* kedelapan, nilai *state* prediksinya sudah mendekati nilai *state* aktual dari sistem tata udara presisi untuk variasi nilai *spectral density gaussian noise* ini. Dengan kata lain, Filter Kalman bekerja dengan baik dalam mengestimasi nilai *state* sistem tata udara presisi ini, nilai kuadrat kesalahan yang terjadi dikarenakan pemberian *spectral density gaussian noise* yang nilainya masih cukup tinggi.

Variasi pemberian *spectral density gaussian noise* yang yang lebih kecil diberikan pada pengujian yang grafik dan analisanya akan ditampilkan pada subbab selanjutnya untuk mengetahui kinerja algortima Filter Kalman apabila diberikan nilai *spectral density gaussian noise* yang lebih kecil lagi.

4.2.2.4 Menggunakan *Spectral Density Gaussian Noise* Sebesar 10^{-3}



Gambar 4.16 Grafik Perbandingan *State* Keenam Estimasi Filter Kalman dengan *State* Sebenarnya pada Sistem Tata Udara Presisi dengan *Spectral Density Gaussian Noise* Sebesar 10^{-3}

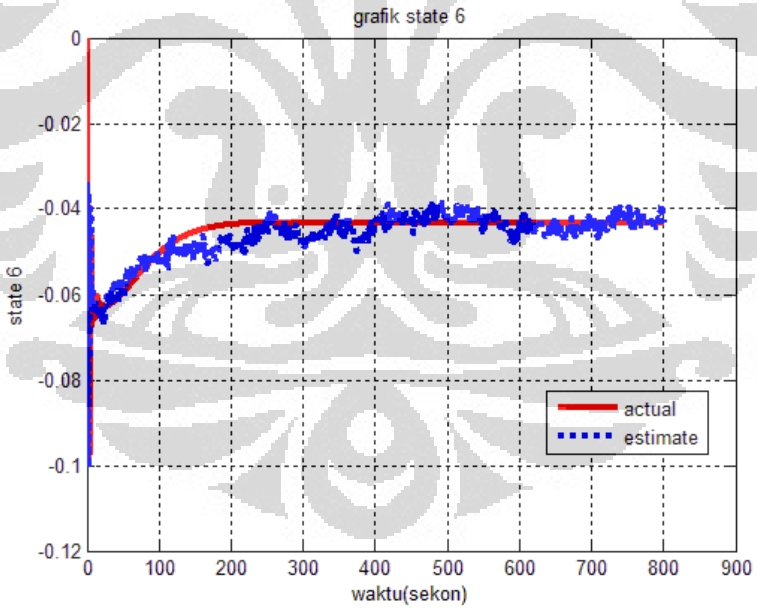
Berdasarkan hasil perhitungan, didapat nilai dari kuadrat kesalahan urntuk masing-masing *state* dengan penambahan spectral density gaussian noise sebesar 0.001 adalah sebagai berikut:

Tabel 4.8 Nilai Kuadrat Kesalahan Estimasi *State* Filter Kalman Menggunakan *Spectral Density Gaussian Noise* Sebesar 10^{-3} dengan Nilai *State* Sebenarnya dari Sistem Tata Udara Presisi

No. *State*	Besarnya Kuadrat Kesalahan
State 1	0.0176
State 2	0.0025
State 3	0.0203
State 4	$1.2714 \text{ X } 10^{-4}$
State 5	$8.3750 \text{ X } 10^{-5}$
State 6	$6.3879 \text{ X } 10^{-5}$
State 7	$9.2718 \text{ X } 10^{-5}$
State 8	$2.1204 \text{ X } 10^{-5}$

Untuk nilai *spectral density gaussian noise* yang lebih kecil lagi, terlihat nilai kuadrat kesalahan yang dihasilkan semakin mengecil. Yang berarti bahwa kinerja Filter Kalman dalam memprediksi nilai *state*nya semakin baik. Nilai-nilai kuadrat kesalahan yang sangat kecil terutama untuk *state* keempat, *state* kelima, *state* keenam, *state* ketujuh, dan *state* kedelapan menunjukkan bahwa *state* hasil prediksi sudah mendekati nilai *state* aktual sistem tata udara presisi. Nilai kuadrat kesalahan untuk *state* pertama, kedua, dan ketiga yang tidak sebagus nilai kuadrat kesalahan yang *state* lainnya, disebabkan karena pengaruh letak pole dari sistem tata udara presisi ini. Karena dilihat dari nilai *eigen value*nya, nilai *eigen value* untuk *state* pertama dan kedua berada pada nilai negatif, sedangkan yang *state* yang lain berada pada nilai positif. Namun kedelapan *eigen value* sistem tata udara presisi hasil identifikasi ini masih berada pada region stabil. Nilainya masih di dalam *unit circle* (kurang dari 1 dan lebih dari -1).

4.2.2.5 Menggunakan Spectral Density Gaussian Noise Sebesar 10^{-4}



Gambar 4.17 Grafik Perbandingan *State* Keenam Estimasi Filter Kalman dengan *State* Sebenarnya pada Sistem Tata Udara Presisi dengan *Spectral Density Gaussian Noise* Sebesar 10^{-4}

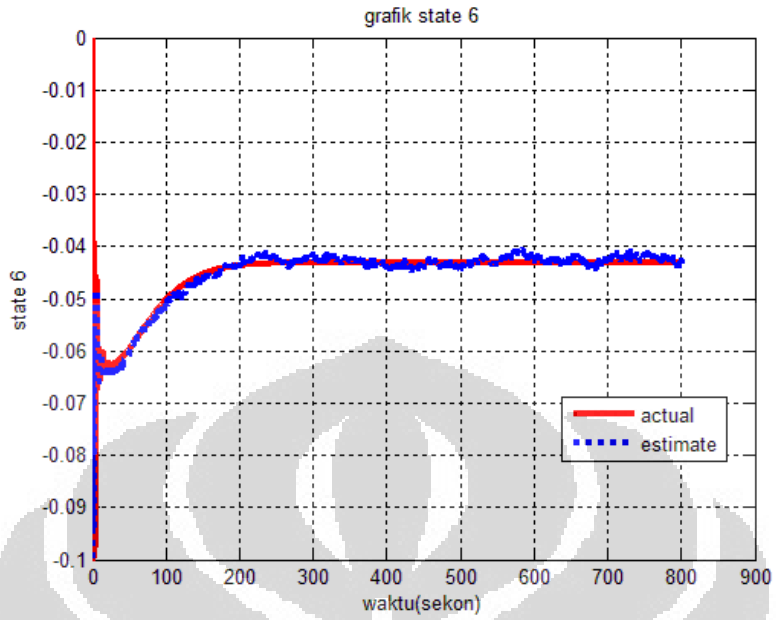
Berdasarkan hasil perhitungan, didapat nilai dari kuadrat kesalahan urntuk masing-masing *state* dengan penambahan *spectral density gaussian noise* sebesar 0.0001 adalah sebagai berikut:

Tabel 4.9 Nilai Kuadrat Kesalahan Estimasi *State* Filter Kalman Menggunakan *Spectral density gaussian noise* Sebesar 10^{-4} dengan Nilai *State* Sebenarnya dari Sistem Tata Udara Presisi

No. *State*	Besarnya Kuadrat Kesalahan
State 1	0.0027
State 2	$4.0644 \text{ X } 10^{-4}$
State 3	0.0031
State 4	$1.6152 \text{ X } 10^{-5}$
State 5	$1.0000 \text{ X } 10^{-5}$
State 6	$7.4402 \text{ X } 10^{-6}$
State 7	$1.3330 \text{ X } 10^{-5}$
State 8	$2.8902 \text{ X } 10^{-6}$

Hasil pengujian algortima Filter Kalman untuk memprediksi *state* sistem tata udara presisi ini menunjukkan kinerja yang memuaskan terlihat pada hasil estimasi selanjutnya yang menunjukkan nilai estimasi yang mendekati nilai aktual *state* sistem tata udara presisi yang ditunjukkan pada gambar grafik 4.21 sampai gambar grafik 4.28 yang menunjukkan perbandingan *state* estimasi Filter Kalman dengan *state* aktual sistem tata udara presisi dengan pemberian *spectral density gaussian noise* yang semakin diperkecil serta tabel 4.9 sampai 4.12 yang menunjukkan nilai kuadrat kesalahan *state* hasil estimasi Filter Kalman dibandingkan dengan *state* aktual sistem tata udara presisi yang menunjukkan nilai kuadrat kesalahan yang semakin mengecil seiring dengan pemberian *spectral density gaussian noise* yang diperkecil.

4.2.2.6 Menggunakan Spectral density gaussian noise sebesar 10^{-5}



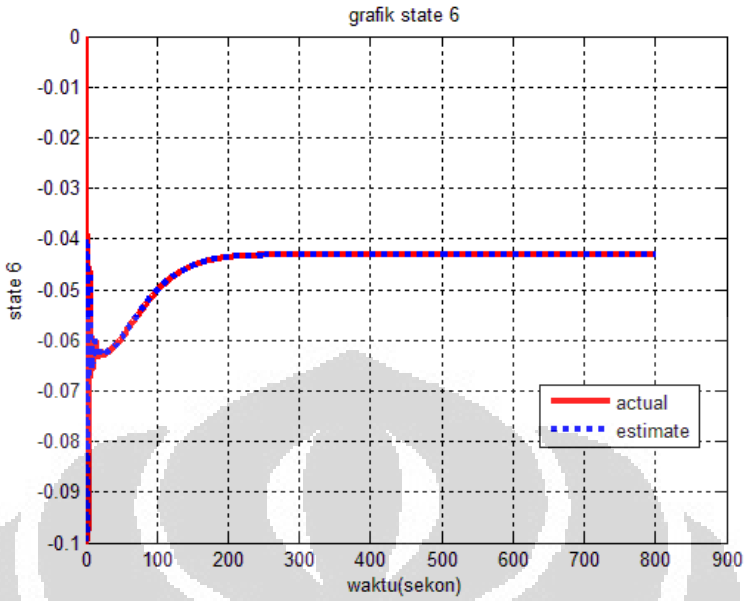
Gambar 4.18 Grafik Perbandingan *State* Keenam Estimasi Filter Kalman dengan *State* Sebenarnya pada Sistem Tata Udara Presisi dengan *Spectral Density Gaussian Noise* Sebesar 10^{-5}

Berdasarkan hasil perhitungan, didapat nilai dari kuadrat kesalahan urntuk masing-masing *state* sebagai berikut:

Tabel 4.10 Nilai Kuadrat Kesalahan Estimasi *State* Filter Kalman Menggunakan *Spectral Density Gaussian Noise* Sebesar 10^{-5} dengan Nilai *State* Sebenarnya dari Sistem Tata Udara Presisi

No. *State*	Besarnya Kuadrat Kesalahan
State 1	$4.5430 \text{ X } 10^{-4}$
State 2	$7.4572 \text{ X } 10^{-5}$
State 3	$3.1337 \text{ X } 10^{-4}$
State 4	$1.3612 \text{ X } 10^{-6}$
State 5	$1.1095 \text{ X } 10^{-6}$
State 6	$1.3470 \text{ X } 10^{-6}$
State 7	$2.0803 \text{ X } 10^{-6}$
State 8	$3.9799 \text{ X } 10^{-7}$

4.2.2.7 Menggunakan *Spectral Density Gaussian Noise* Sebesar 10^{-8}



Gambar 4.19 Grafik Perbandingan *State* Keenam Estimasi Filter Kalman dengan *State* Sebenarnya pada Sistem Tata Udara Presisi dengan *Spectral Density Gaussian Noise* Sebesar 10^{-8}

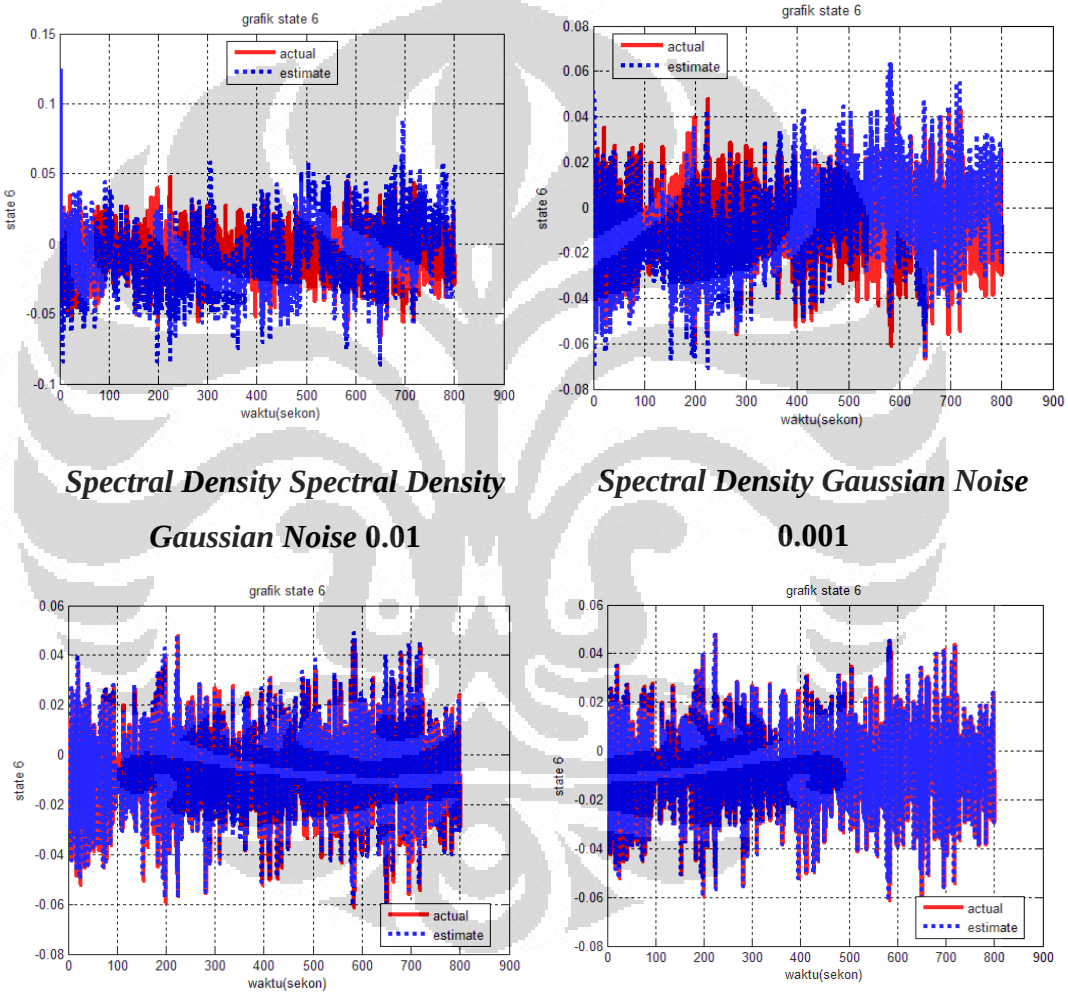
Berdasarkan hasil perhitungan didapat nilai dari kuadrat kesalahan sebagai berikut:

Tabel 4.11 Nilai Kuadrat Kesalahan Estimasi *State* Filter Kalman Menggunakan *Spectral Density Gaussian Noise* Sebesar 10^{-8} dengan Nilai *State* Sebenarnya dari Sistem Tata Udara Presisi

No. *State*	Besarnya Kuadrat Kesalahan
State 1	$2.1672 \text{ X } 10^{-7}$
State 2	$3.3297 \text{ X } 10^{-8}$
State 3	$3.6237 \text{ X } 10^{-7}$
State 4	$2.0221 \text{ X } 10^{-9}$
State 5	$1.1973 \text{ X } 10^{-9}$
State 6	$1.1054 \text{ X } 10^{-9}$
State 7	$1.2508 \text{ X } 10^{-9}$
State 8	$3.1938 \text{ X } 10^{-10}$

4.2.3 Pengujian Pada Sistem Tata Udara Presisi Secara *Open Loop* dengan Sinyal Kendali Data Rekam Sinyal Random

Diambil contoh grafik nilai *state* estimasi dan *state* sebenarnya dari *state* keenam untuk masing-masing hasil estimasi. Hasil pengujian dengan menggunakan sinyal kendali data rekam berupa sinyal random didapatkan sebagai berikut: Dengan menggunakan *spectral density gaussian noise* sebesar 0.01; 0.001; 0.0001; dan menggunakan *spectral density gaussian noise* sebesar nol.



***Spectral Density Spectral Density Gaussian Noise* 0.01**

***Spectral Density Gaussian Noise* 0.001**

***Spectral Density Gaussian Noise* 0.0001**

***Spectral Density Gaussian Noise* Nol**

Gambar 4.20 Grafik Perbandingan *State* Keenam Sistem Tata Udara Presisi dengan Sinyal Kendali Data Rekam Sinyal Random dengan Berbagai Variasi Nilai *Spectral Density Gaussian Noise*

Dari hasil estimasi filter kalman, menunjukkan bahwa semakin kecil *spectral density noise* yang diberikan pada sistem, hasil estimasinya semakin baik dan sudah mendekati nilai *state* sebenarnya. Kesalahan hasil estimasi terlihat dalam kesalahan kuadrat hasil estimasi dengan nilai aktual *state*.

Tabel 4.12 Nilai Kuadrat Kesalahan Estimasi *State* Filter Kalman Menggunakan *Spectral Density Gaussian Noise* Sebesar 10^{-2} dengan Nilai *State* Sebenarnya dari Sistem Tata Udara Presisi Secara *Open loop* Menggunakan Sinyal Kendali Data Rekam Random

No. *State*	Besarnya Kuadrat Kesalahan
State 1	0.1413
State 2	0.0254
State 3	0.2577
State 4	0.0018
State 5	0.0010
State 6	$4.7466 \text{ X } 10^{-4}$
State 7	$7.3702 \text{ X } 10^{-4}$
State 8	$1.9790 \text{ X } 10^{-4}$

Nilai kuadrat kesalahan menunjukkan nilai yang relatif besar, hal ini dikarenakan besarnya *spectral density gaussian noise* yang diberikan juga relatif besar. Untuk selanjutnya, variasi nilai *spectral density gaussian noise* yang semakin diperkecil menunjukkan hasil estimasi yang semakin membaik. Untuk nilai *spectral density gaussian noise* 0.001, besarnya nilai kuadrat kesalahan untuk masing-masing statenya lebih kecil daripada untuk variasi nilai *spectral density gaussian noise* 0.01. Besarnya kuadrat kesalahan dapat dilihat dari tabel 2.

Nilai kuadrat kesalahan untuk estimasi state pertama, kedua, dan ketiga, cenderung lebih besar daripada nilai kuadrat kesalahan untuk estimasi state yang lain. Namun, nilai kuadrat ini masih tergolong kecil karena variasi nilai *spectral density gaussian noise* yang digunakan masih relatif besar.

Tabel 4.13 Nilai Kuadrat Kesalahan Estimasi *State* Filter Kalman Menggunakan *Spectral Density Gaussian Noise* Sebesar 10^{-3} dengan Nilai *State* Sebenarnya dari Sistem Tata Udara Presisi Secara *Open loop* Menggunakan Sinyal Kendali Data Rekam Random

No. *State*	Besarnya Kuadrat Kesalahan
State 1	0.0227
State 2	0.0036
State 3	0.0461
State 4	$2.4399 \text{ X } 10^{-4}$
State 5	$1.3367 \text{ X } 10^{-4}$
State 6	$8.0514 \text{ X } 10^{-5}$
State 7	$1.1683 \text{ X } 10^{-4}$
State 8	$3.0133 \text{ X } 10^{-5}$

Tabel 4.14 Nilai Kuadrat Kesalahan Estimasi *State* Filter Kalman Menggunakan *Spectral Density Gaussian Noise* Sebesar 10^{-4} dengan Nilai *State* Sebenarnya dari Sistem Tata Udara Presisi Secara *Open loop* Menggunakan Sinyal Kendali Data Rekam Random

No. *State*	Besarnya Kuadrat Kesalahan
State 1	0.0017
State 2	$3.6804 \text{ X } 10^{-4}$
State 3	0.0016
State 4	$1.6506 \text{ X } 10^{-5}$
State 5	$1.0576 \text{ X } 10^{-5}$
State 6	$3.9507 \text{ X } 10^{-6}$
State 7	$6.6218 \text{ X } 10^{-6}$
State 8	$1.3430 \text{ X } 10^{-6}$

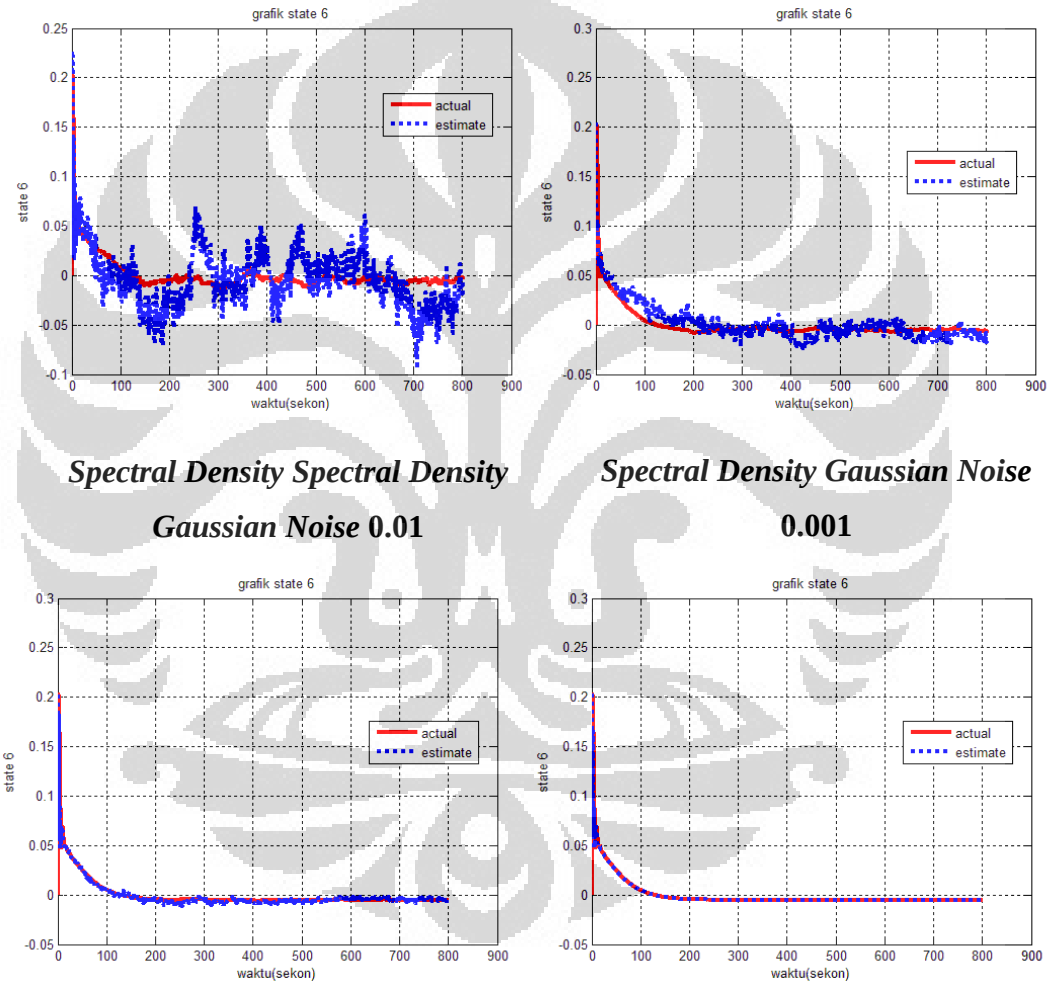
Tabel 4.15 Nilai Kuadrat Kesalahan Estimasi *State* Filter Kalman Menggunakan *Spectral Density Gaussian Noise* Sebesar Nol dengan Nilai *State* Sebenarnya dari Sistem Tata Udara Presisi Secara *Open loop* Menggunakan Sinyal Kendali Data Rekam Random

No. *State*	Besarnya Kuadrat Kesalahan
State 1	$1.5540 \text{ X } 10^{-35}$
State 2	$5.8410 \text{ X } 10^{-35}$
State 3	$1.3834 \text{ X } 10^{-35}$
State 4	$1.4389 \text{ X } 10^{-35}$
State 5	$1.0684 \text{ X } 10^{-35}$
State 6	$5.5381 \text{ X } 10^{-36}$
State 7	$2.0450 \text{ X } 10^{-36}$
State 8	$1.3218 \text{ X } 10^{-36}$

Dari hasil perhitungan kuadrat kesalahan masing-masing *state* terlihat bahwa nilai *state* hasil estimasinya sudah sangat mirip dengan nilai *state* aktualnya, atau bisa dikatakan nilainya sudah sama antara *state* hasil estimasi dengan nilai *state* sebenarnya. Hal ini menunjukkan bahwa kinerja filter kalman dalam mengestimasi nilai *state* sistem tata udara presisi sudah menunjukkan proses estimasi yang baik. Beberapa nilai kuadarat kesalahan untuk dua *state* (*state* pertama dan *state* ketiga) yang masih relatif besar untuk variasi *spectral density gaussian noise* terjadi karena pengaruh dari model linier sistem tata udara presisi yang digunakan yang didapat dari proses sistem identifikasi.

4.2.4 Pengujian Pada Sistem Tata Udara Presisi Secara *Closed Loop* dengan Sinyal Kendali LQR

Diambil contoh grafik nilai *state* estimasi dan *state* sebenarnya dari *state* keenam untuk masing-masing hasil estimasi. Hasil pengujian dengan menggunakan sinyal kendali LQR didapatkan sebagai berikut: Dengan menggunakan *spectral density gaussian noise* sebesar 0.01; 0.001; 0.0001; dan menggunakan *spectral density gaussian noise* sebesar nol.



***Spectral Density Spectral Density Gaussian Noise* 0.01**

***Spectral Density Gaussian Noise* 0.001**

***Spectral Density Gaussian Noise* 0.0001**

***Spectral Density Gaussian Noise* Nol**

Gambar 4.21 Grafik Perbandingan *State* Keenam Sistem Tata Udara Presisi dengan Sinyal Kendali LQR dengan Berbagai Variasi Nilai *Spectral Density Gaussian Noise*

Dari hasil estimasi filter kalman, menunjukkan bahwa semakin kecil *spectral density noise* yang diberikan pada sistem, hasil estimasinya semakin baik dan sudah mendekati nilai *state* sebenarnya. Kesalahan hasil estimasi terlihat dalam kesalahan kuadrat hasil estimasi dengan nilai aktual *state*.

Untuk nilai *spectral density gaussian noise* 0.01, besarnya kuadrat kesalahan yang terjadi untuk *state* keempat sampai kedelapan sudah menunjukkan hasil estimasi yang baik karena nilai kuadrat kesalahannya sangat kecil. Tapi untuk *state* pertama, kedua, dan ketiga nilai kuadrat kesalahannya masih agak besar. Hal ini dikarenakan selain nilai noise yang diberikan masih cukup besar tetapi juga pengaruh letak kutub dari *state* yang nilainya negatif berbeda dengan *state* yang lain yang nilainya positif meskipun semua kutub *state* berada dalam *unit circle*.

Tabel 4.16 Nilai Kuadrat Kesalahan Estimasi *State* Filter Kalman Menggunakan *Spectral Density Gaussian Noise* Sebesar 10^{-2} dengan Nilai *State* Sebenarnya dari Sistem Tata Udara Presisi Secara *Closed loop* Menggunakan Sinyal Kendali LQR

No. *State*	Besarnya Kuadrat Kesalahan
State 1	0.1472
State 2	0.0260
State 3	0.1628
State 4	0.0014
State 5	$9.3863 \text{ X } 10^{-4}$
State 6	$4.6867 \text{ X } 10^{-4}$
State 7	$6.8884 \text{ X } 10^{-4}$
State 8	$1.5870 \text{ X } 10^{-4}$

Untuk nilai s*pectral density gaussian noise* 0.001, nilai kuadrat kesalahannya lebih kecil bila dibandingkan dengan pengaruh *spectral density gaussian noise* 0.01. Nilai kuadarat kesalahan *state* pertama dan ketiga masih lebih besar daripada nilai kuadrat kesalahan dari keenam *state* yang lainnya. Namun nilainya kuadrat kesalahannya lebih kecil dibandingkan dengan nilai estimasi dengan variasi *spectral density gaussian noise* 0.01 sebelumnya. Yang

berarti proses estimasi yang terjadi adalah benar, karena nilai kuadrat kesalahannya turun sebanding dengan kecilnya noise yang diberikan.

Tabel 4.17 Nilai Kuadrat Kesalahan Estimasi *State* Filter Kalman Menggunakan *Spectral Density Gaussian Noise* Sebesar 10^{-3} dengan Nilai *State* Sebenarnya dari Sistem Tata Udara Presisi Secara *Closed loop* Menggunakan Sinyal Kendali LQR

No. *State*	Besarnya Kuadrat Kesalahan
State 1	0.0221
State 2	0.0034
State 3	0.0437
State 4	2.1722×10^{-4}
State 5	1.1754×10^{-4}
State 6	8.9765×10^{-5}
State 7	1.2083×10^{-4}
State 8	3.0473×10^{-5}

Tabel 4.18 Nilai Kuadrat Kesalahan Estimasi *State* Filter Kalman Menggunakan *Spectral Density Gaussian Noise* Sebesar 10^{-4} dengan Nilai *State* Sebenarnya dari Sistem Tata Udara Presisi Secara *Closed loop* Menggunakan Sinyal Kendali LQR

No. *State*	Besarnya Kuadrat Kesalahan
State 1	0.0045
State 2	4.6946×10^{-4}
State 3	0.0089
State 4	2.2246×10^{-5}
State 5	1.0820×10^{-5}
State 6	1.5959×10^{-5}
State 7	2.6983×10^{-5}
State 8	6.9025×10^{-6}

Performa yang baik ditunjukkan untuk variasi *spectral density gaussian noise* sebesar 10^{-4} dimana kuadrat kesalahan yang terjadi untuk seluruh *state*

sangat kecil. Dalam hal ini untuk variasi *spectral density gaussian noise* yang lebih kecil lagi, akan menghasilkan estimasi yang nilainya mendekati nilai *state* sebenarnya. Untuk menganalisa lebih mendalam mengenai kinerja filter kalman dalam mengestimasi kedelapan *state* sistem tata udara presisi, maka *spectral density gaussian noise*nya dijadikan nol.

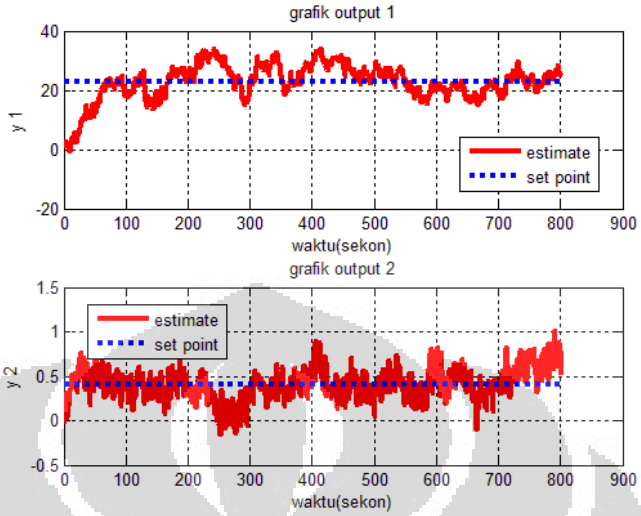
Tabel 4.19 Nilai Kuadrat Kesalahan Estimasi *State* Filter Kalman Menggunakan *Spectral Density Gaussian Noise* Sebesar Nol dengan Nilai *State* Sebenarnya dari Sistem Tata Udara Presisi Secara *Closed loop* Menggunakan Sinyal Kendali LQR

No. *State*	Besarnya Kuadrat Kesalahan
State 1	$1.2892 \text{ X } 10^{-32}$
State 2	$1.4609 \text{ X } 10^{-33}$
State 3	$1.8632 \text{ X } 10^{-33}$
State 4	$2.0436 \text{ X } 10^{-33}$
State 5	$1.2341 \text{ X } 10^{-33}$
State 6	$1.3100 \text{ X } 10^{-33}$
State 7	$5.2863 \text{ X } 10^{-34}$
State 8	$1.7890 \text{ X } 10^{-34}$

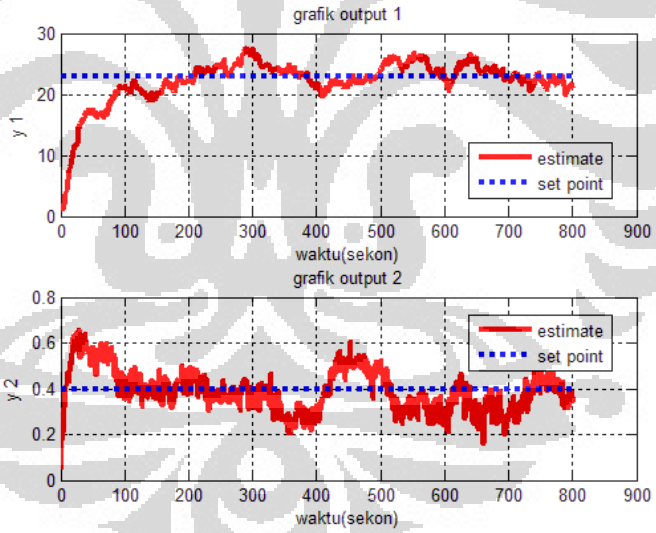
Dari hasil perhitungan kuadrat kesalahan masing-masing *state* terlihat bahwa nilai *state* hasil estimasinya sudah sangat mirip dengan nilai *state* aktualnya, atau bisa dikatakan nilainya sudah sama antara *state* hasil estimasi dengan nilai *state* sebenarnya. Hal ini menunjukkan bahwa kinerja filter kalman dalam mengestimasi nilai *state* sistem tata udara presisi sudah menunjukkan proses estimasi yang baik. Beberapa nilai kuadarat kesalahan untuk dua *state* (*state* pertama dan *state* ketiga) yang masih relatif besar untuk variasi *spectral density gaussian noise* terjadi karena pengaruh dari model linier sistem tata udara presisi yang digunakan yang didapat dari proses sistem identifikasi.

Untuk output yang dihasilkan juga menunjukkan hasil estimasi filter kalman mampu mengikuti *set point* yang diberikan. Output pertama (y_1) berupa suhu yang harus dicapai dan nilai *set point*nya diset pada nilai 23ºC, sedangkan

output kedua (y_2) merupakan kelembaban yang harus dicapai. Nilai *set point*nya diset pada nilai 0,4 atau 40%.



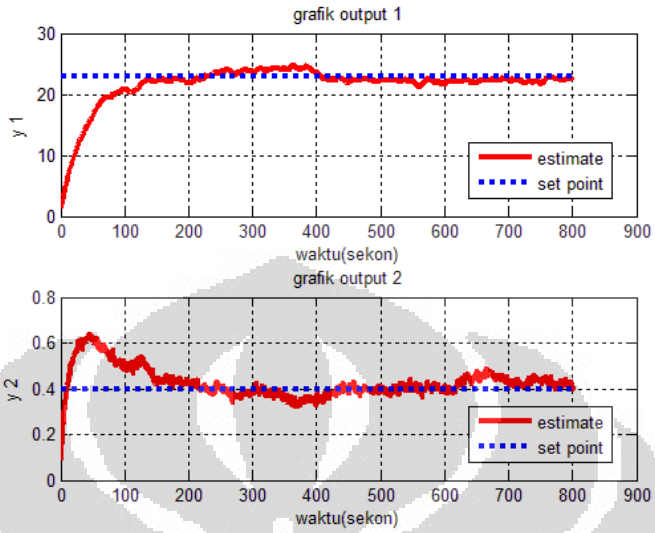
Gambar 4.22 Keluaran Pada Sistem Tata Udara Presisi dengan *Spectral Density Spectral Density Gaussian Noise* sebesar 0.01



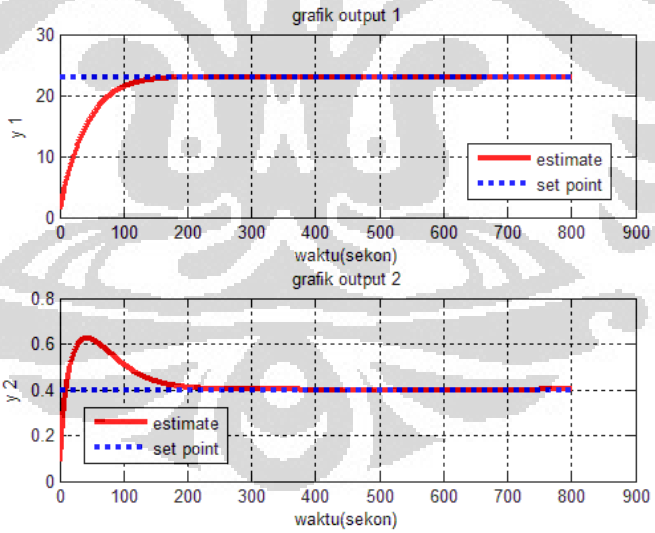
Gambar 4.23 Keluaran Pada Sistem Tata Udara Presisi dengan *Spectral Density Spectral Density Gaussian Noise* Sebesar 0.001

Berdasarkan grafik keluaran menunjukkan bahwa hasil estimasi telah mampu mengikuti *set point* yang diberikan. Dan untuk *spectral density gaussian noise* yang semakin kecil, menunjukkan performa hasil estimasi yang semakin baik dan semakin mengikuti *set point* yang diberikan. Analisa selanjutnya dilakukan dengan memperkecil *noise* sampai dengan nilai nol. Dari hasil

penelitian menunjukkan, filter kalman mampu mengestimasi secara sempurna *state* pada sistem tata udara presisi, sehingga menghasilkan keluaran yang sesuai dengan *set point* yang diberikan.



Gambar 4.24 Keluaran Pada Sistem Tata Udara Presisi dengan *Spectral Density Gaussian Noise* Sebesar 0.0001



Gambar 4.25 Keluaran Pada Sistem Tata Udara Presisi dengan *Spectral Density Gaussian Noise* Sebesar Nol

Dengan proses estimasi tanpa *noise* menunjukkan nilai keluaran yang sama persis dengan nilai *set point* yang diberikan. Sehingga kalman filter dapat digunakan untuk mengestimasi *state* pada sistem tata udara presisi.

BAB 5
KESIMPULAN

Kesimpulan dari hasil penelitian yang telah dilakukan adalah sebagai berikut:

1. Hasil estimasi *state* sistem menggunakan algoritma Filter Kalman tergantung dari model A, B, C, D sistem. Semakin linear model sistem, maka hasil estimasi *state* akan semakin baik.
2. Hasil estimasi state sistem menggunakan algoritma Filter Kalman dipengaruhi juga oleh besarnya *spectral density gaussian noise* yang ada pada proses maupun pada pengukuran. Semakin kecil *spectral density gaussian noise* yang ada pada proses maupun pengukuran, maka hasil estimasi *state* sistem semakin baik.
3. Hasil estimasi *state* menggunakan algoritma Filter Kalman pada model sistem CSTR menghasilkan estimasi *state* yang cukup bagus untuk beberapa variasi *spectral density gaussian noise* yang diberikan.
4. Hasil estimasi state menggunakan algoritma Filter Kalman pada model sistem tata udara presisi menghasilkan estimasi *state* yang cukup bagus untuk beberapa variasi *spectral density gaussian noise* yang diberikan baik secara sistem *open loop* maupun sistem *cloosed loop*.
5. Optimasi nilai matriks kovarian *error* proses (**Q**) dan matriks kovarian *error* pengukuran (**R**) sangat diperlukan dalam setiap proses estimasi menggunakan algoritma Filter Kalman.
6. Hasil optimasi matriks **Q** dan matriks **R** untuk model sistem CSTR yang digunakan dalam penelitian ini adalah

$$\boldsymbol{Q} = 10^{-3} \times \begin{bmatrix} 0.6 & -0.1 \\ -0.1 & 0.6 \end{bmatrix} \text{ dan } \boldsymbol{R} = 10^{-3} \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

7. Hasil optimasi matriks **Q** dan matriks **R** untuk model sistem tata udara presisi yang digunakan dalam penelitian ini adalah

$$Q(3) = 10^{-3} \times \begin{bmatrix} 3.60 & 0.01 & 0.00 & 0.00 & 0.01 & 0.00 & 0.00 & 0.00 \\ 0.00 & 3.60 & 0.01 & 0.00 & 0.00 & 0.01 & 0.01 & 0.00 \\ -0.1 & 0.01 & 3.60 & 0.01 & 0.00 & 0.00 & 0.01 & 0.00 \\ 0.00 & 0.01 & 0.00 & 3.60 & 0.00 & 0.01 & 0.00 & 0.00 \\ 0.00 & 0.01 & 0.00 & 0.01 & 3.60 & 0.01 & 0.00 & 0.00 \\ 0.00 & 0.01 & 0.00 & 0.01 & 0.01 & 3.60 & 0.00 & 0.00 \\ 0.00 & 0.01 & 0.00 & 0.01 & 0.01 & 0.00 & 3.60 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 00.0 & 0.01 & 3.60 \end{bmatrix}$$

dan

$$R(3) = 10^{-3} \times \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}$$

DAFTAR REFERENSI

[1] A. Tianoa, R.Suttonb, A. Lozowicki, W. Naeemb. (2007). Observer Kalman filter identification of an autonomous Underwater vehicle, 2006. *Elsevier. Control Engineering Practice 15, 727–739*

[2] A. Vasebi, S. M. T. Bathaee, M. Partovibakhsh. (2008). Predicting state of charge of lead-acid batteries for hybrid electric vehicles by extended Kalman filter, 2007. *Elsevier. Energy Conversion and Management 49, 75–82.*

[3] Brendan M. Quine. (2006). A derivative-free implementation of the extended Kalman filter, 2006. *Elsevier. Automatica 42, 1927–1934*

[4] D. Loebis, R. Sutton, J. Chudley, W. Naeem. (2003). Adaptive tuning of a Kalman filter via fuzzy logic for an intelligent AUV navigation system. *Elsevier. Control Engineering Practice 12 (2004) 1531–1539*

[5] H. M. Al-Hamadi. (2011). Fuzzy logic voltage flicker estimation using Kalman filter. *Elsevier. Electrical Power and Energy Systems. Int J Electr Power Energ Syst (2011), doi: 10.1016/j.ijepes.2011.10.024*

[6] J. Kim. (2008). Identification of lateral tyre force dynamics using an extended Kalman filter From experimental road test data. *Elsevier. Control Engineering Practice 17 (2009) 357–367*

[7] John Valasek, Wei Chen. (2003). Observer/Kalman Filter Identication for Online System Identcation of Aircraft. *JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS Vol. 26, No. 2, March–April 2003*

[8] Jose de Jesus Rubioa, Wen Yub. (2006). Non linear system identification with recurrent neural networks and dead-zone Kalman filter algorithm. *Elsevier. Neurocomputing 70 (2007) 2460–2466*

[9] L. Boillereaux ,H. Fibrianto, J. M. Flaus. (2005). A switched Kalman filter dedicated to Assisted pressure food thawing. *Elsevier. Computers and Electronicsin Agriculture 49 (2005) 392–406*

[10] Mickael Hilairet, FrancoisAuger, Eric Berthelot. (2007). Speed and rotor flux estimation of induction machines using a two-stage extended Kalman filter. *Elsevier. Automatica 45 (2009) 1819-1827*

[11] Murat Barut. (2009). Bi Input-extended Kalman filter based estimation technique for speed-sensorless control of induction motors. *Elsevier. Energy Conversion and Management 51 (2010) 2032–2040.*

[12] Nicolas Boizot, Eric Busvelle, Jean-Paul Gauthier. (2010). An adaptive high-gain observer for nonlinear systems. *Elsevier. Automatica 46 (2010) 1483-1488*

[13] Pratap R. Patnaik. (2004). The extended Kalman filter as a noise modulator for continuous yeast Cultures under monotonic, oscillating and chaotic conditions. *Elsevier. Chemical Engineering Journal 108 (2005) 91–99*

[14] Rizky P.A.N. (2011). *Strategi lokalisasi mobile robot dengan menggunakan Extended Kalman Filter pada lingkungan terstruktur*. Depok: Departemen Elektro, Fakultas Teknik, Universitas Indonesia

[15] Salvatore A.Velardi, Hassan Hammouri, Antonello A. Barresi. (2009). In line monitoring of the primary drying phase of the Freeze drying process in vial by means of a Kalman filter based observer. *Chemical engineering research and design 87 (2009) 1409–1419*

[16] Tae Yoon Um, Jang Gyu Lee, Seong Taek Park, Chan Gook Park. (2000). Noise Covariances Estimation for Systems with Bias States. IEEE Transactions on Aerospace and Electronic Systems, vol. 36, No. 1 January 2000.

[17] W. J. Sung, S. C. Lee, K. H. You. (2008). Ultra-precision positioning using adaptive fuzzy-Kalman filter observer. *Elsevier. Precision Engineering 34 (2010) 195–199*

[18] WANG Jianlin, ZHAO Liqiang, YU Tao. (2010). On-line Estimation in Fed-batch Fermentation Process Using State Space Model and Unscented Kalman Filter. *PROCESS SYSTEMS ENGINEERING Chinese Journal of Chemical Engineering, 18 (2) 258-264 (2010)*

[19] Victor. (2011). *Identifikasi Model Ruang Keadaan Multivariabel pada Sistem Tata Udara Presisi Menggunakan Algoritma Subspace State-Space System Identification (4SID).* Depok: Departemen Elektro, Fakultas Teknik, Universitas Indonesia

[20] Weihua Li, Sirish L. Shaha, Deyun Xiao. (2007). Kalman filters in non-uniformly sampled multirate systems: For FDI and beyond. *Elsevier. Automatica 44 (2008) 199–208*

[21] X. Luoa, I. M. Moroz. (2008). Ensemble Kalman filter with the unscented transform. *Elsevier. Physica D238 (2009) 549-562*

[22] Yibing Wang, Markos Papageorgiou. (2004). Real-time free way traffic state estimation based on Extended Kalman filter : a general approach. *Elsevier. Transportation Research Part B 39 (2005) 141–167*

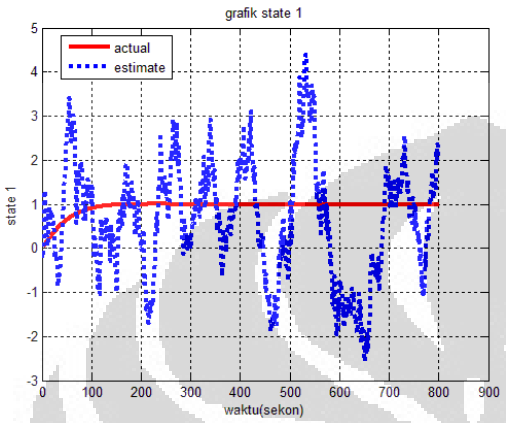
[23] Yongjin (James) Kwon, Yongmin Park. (2011). Improvement of vision guided robotic accuracy using Kalman filter. *Elsevier. Computers & Industrial Engineering xxx (2011), doi:10.1016/j.cie.2011.11.018*

[24] Zongbo Xie, Jiuchao Feng. (2011). Real-time nonlinear structural system identification via iterated unscented Kalman filter. *Elsevier. Mechanical Systems and Signal Processing. doi:10.1016/j.ymssp.2011.02.005*

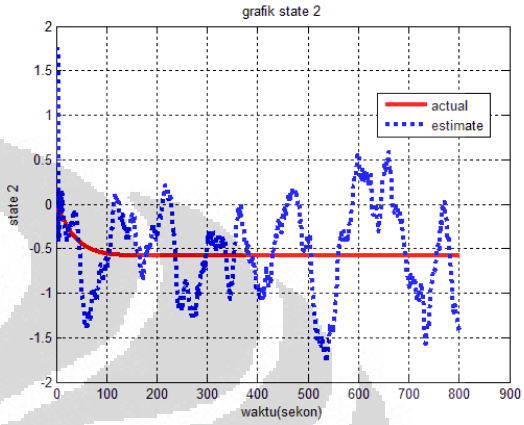
[25] Zhuang Fu, M. F. Rahman. An Extended Kalman Filter Observer for the Direct Torque Controlled Interior Permanent Magnet Synchronous Motor Drive. IEEE. Volume 18, issue 1.

LAMPIRAN A

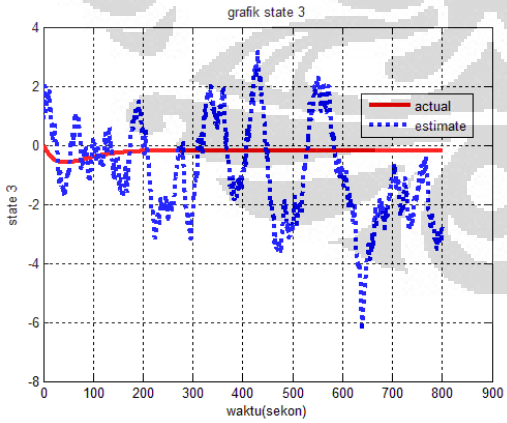
Grafik Perbandingan *State* Estimasi dengan *State* Sebenarnya dari Sistem Tata Uadara Presisi Secara *Open Loop* Menggunakan Sinyal Kendali Data Rekam Sinyal Konstan dengan Berbagai Variasi *Spectral Density Gaussian Noise*



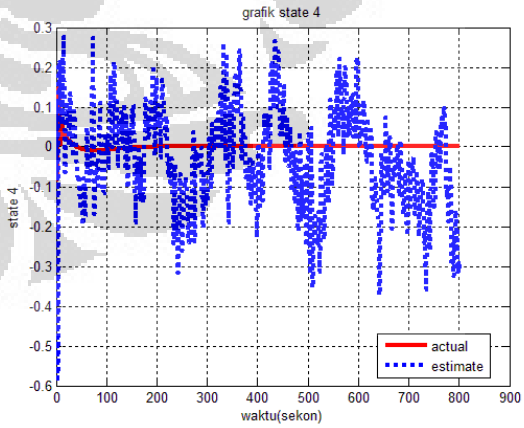
State Pertama
Untuk *Spectral Density Gaussian Noise* 0.1



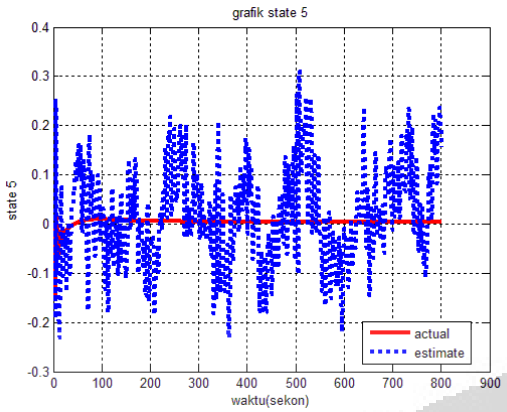
State Kedua
Untuk *Spectral Density Gaussian Noise* 0.1



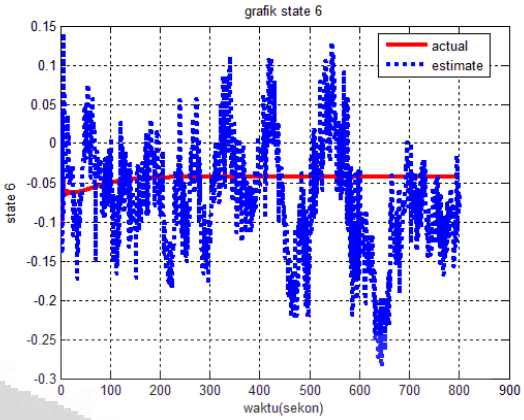
State Ketiga
Untuk *Spectral Density Gaussian Noise* 0.1



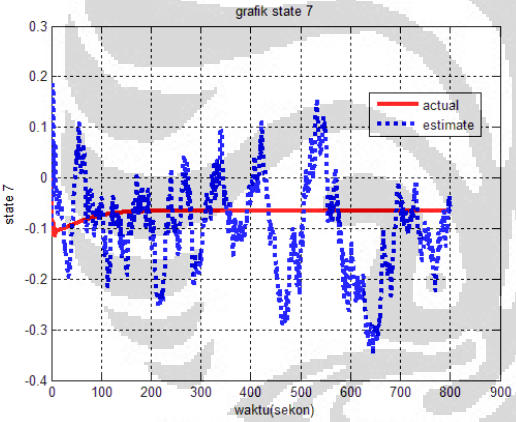
State Keempat
Untuk *Spectral Density Gaussian Noise* 0.1



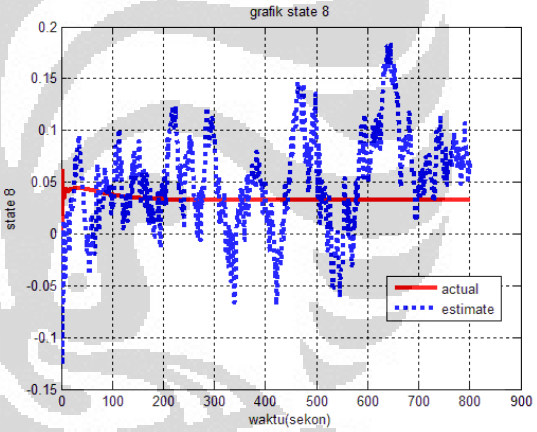
State Kelima
Untuk *Spectral Density Gaussian Noise* 0.1



State Keenam
Untuk *Spectral Density Gaussian Noise* 0.1

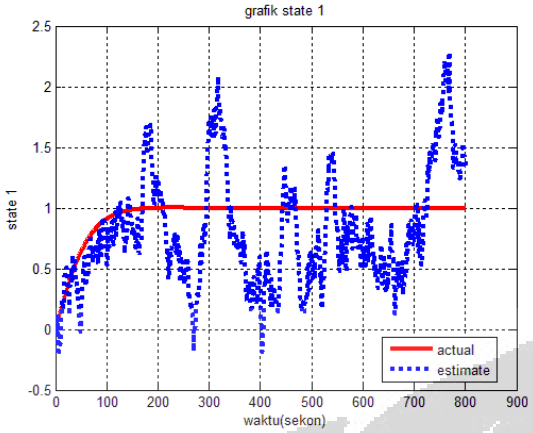


State Ketujuh
Untuk *Spectral Density Gaussian Noise* 0.1

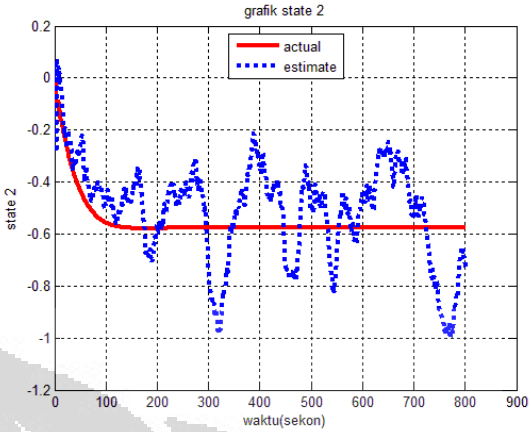


State Pertama
Untuk *Spectral Density Gaussian Noise* 0.1

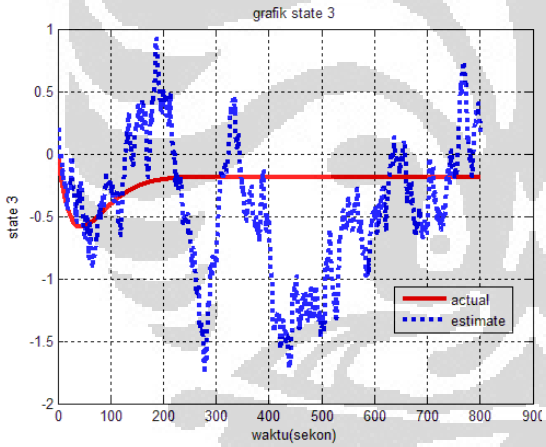
***Spectral Density Gaussian Noise* 0.01**



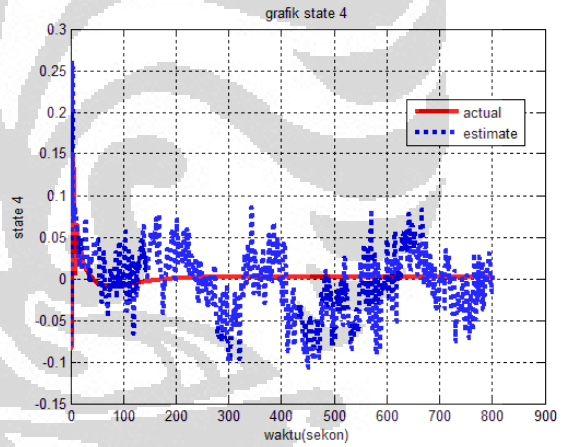
State Pertama Untuk *Spectral Density Gaussian Noise* 0.01



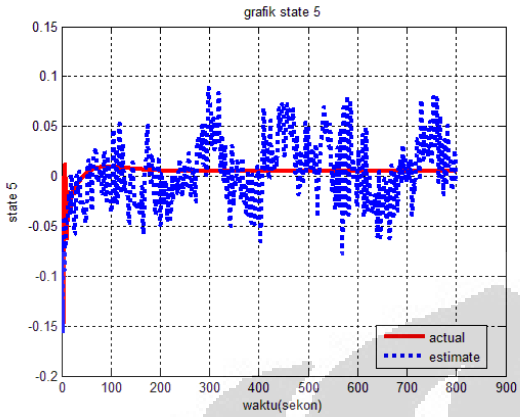
State Kedua Untuk *Spectral Density Gaussian Noise* 0.01



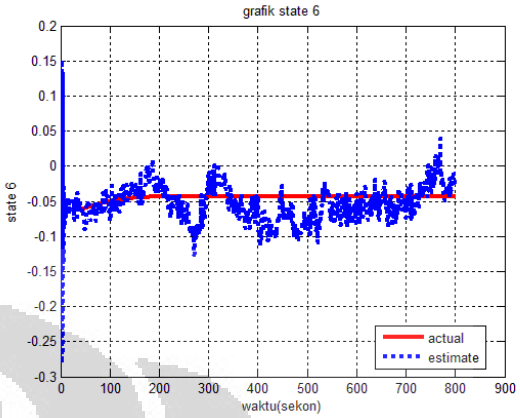
State Ketiga Untuk *Spectral Density Gaussian Noise* 0.01



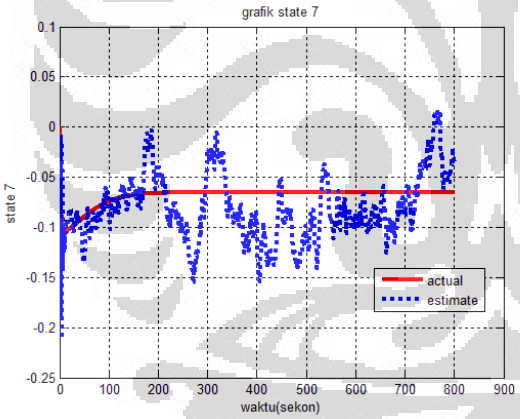
State Keempat Untuk *Spectral Density Gaussian Noise* 0.01



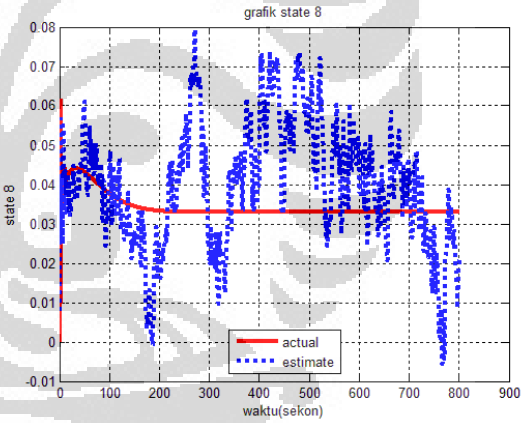
State Kelima
Untuk *Spectral Density*
Gaussian Noise 0.01



State Keenam
Untuk *Spectral Density*
Gaussian Noise 0.01

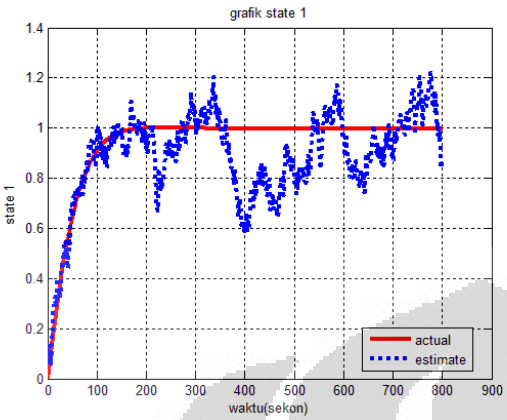


State Ketujuh
Untuk *Spectral Density*
Gaussian Noise 0.01

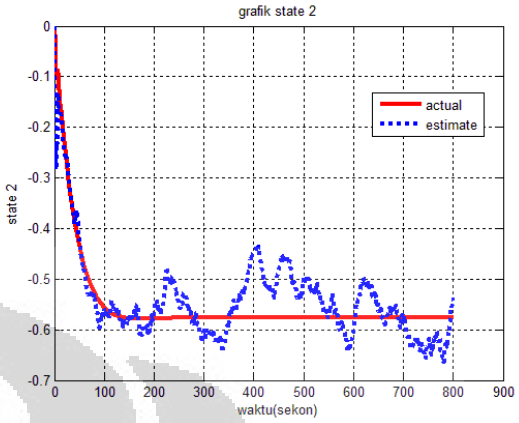


State Kedelapan
Untuk *Spectral Density*
Gaussian Noise 0.01

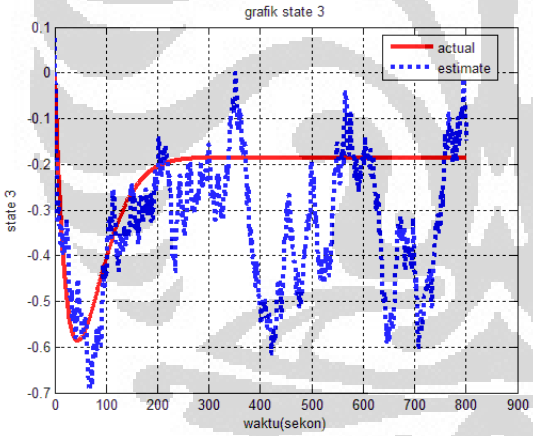
Untuk *Spectral Density Gaussian Noise* 10^{-3}



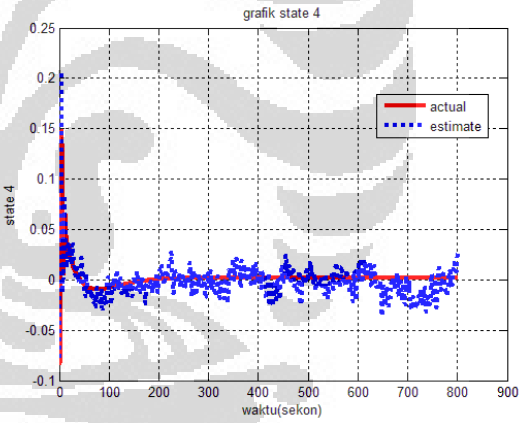
State Pertama Untuk *Spectral Density Gaussian Noise* 0.001



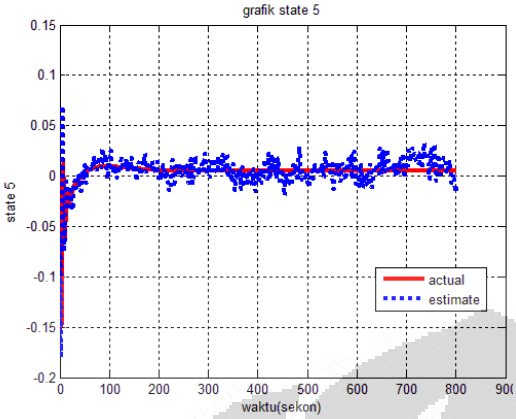
State Kedua Untuk *Spectral Density Gaussian Noise* 0.001



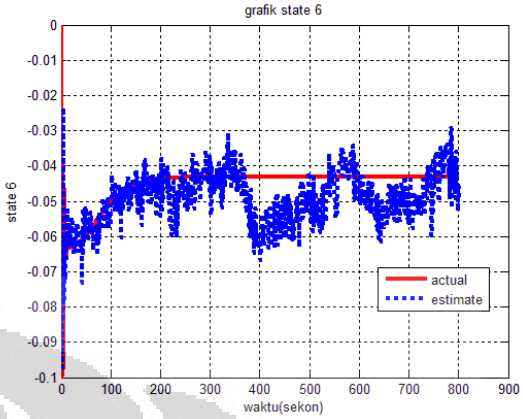
State Ketiga Untuk *Spectral Density Gaussian Noise* 0.001



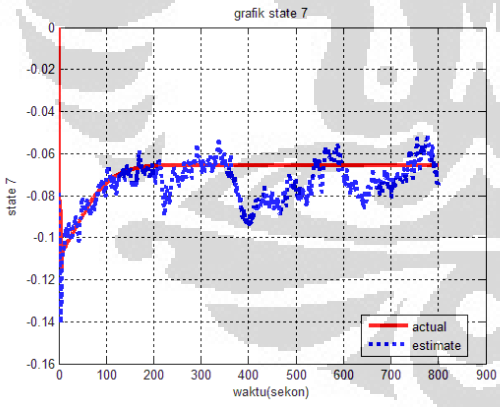
State Keempat Untuk *Spectral Density Gaussian Noise* 0.001



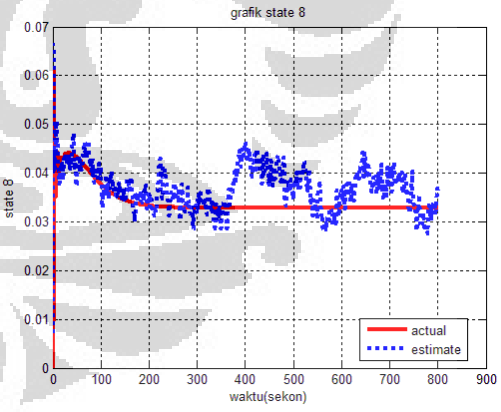
State Kelima
Untuk *Spectral Density*
Gaussian Noise 0.001



State Keenam
Untuk *Spectral Density*
Gaussian Noise 0.001

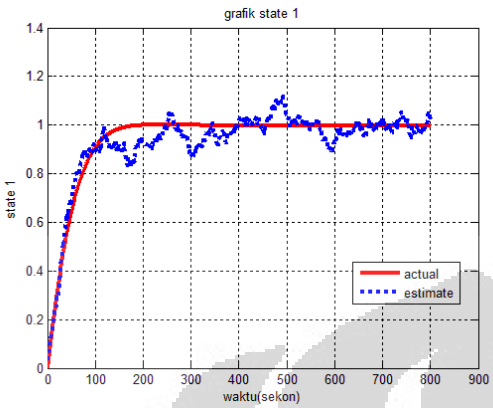


State Ketujuh
Untuk *Spectral Density*
Gaussian Noise 0.001

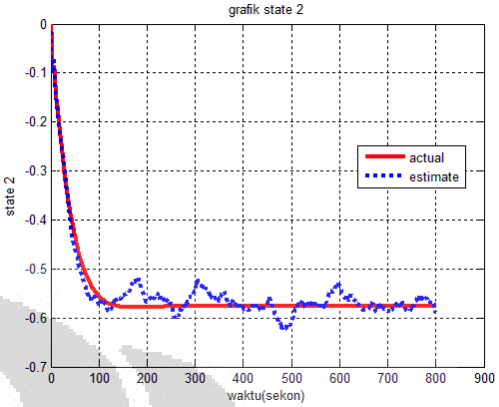


State Kedelapan
Untuk *Spectral Density*
Gaussian Noise 0.001

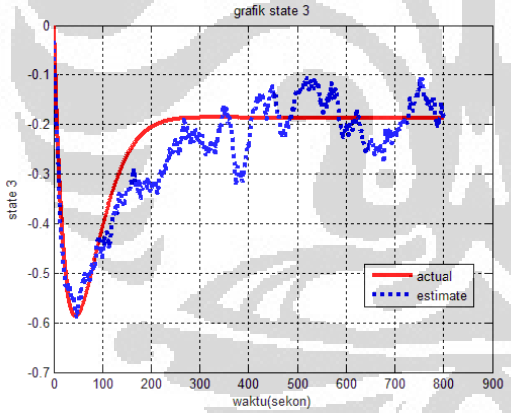
Untuk *Spectral Density* Gaussian Noise 10^{-4}



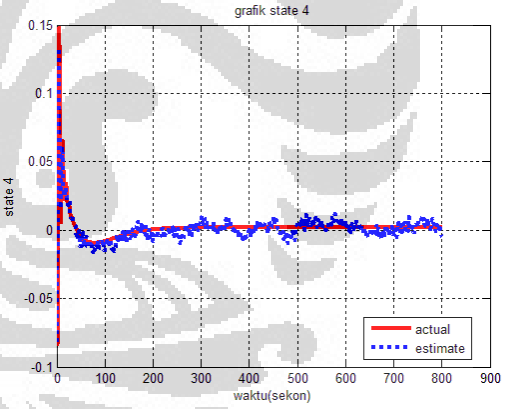
State Pertama Untuk *Spectral Density Gaussian Noise* 0.0001



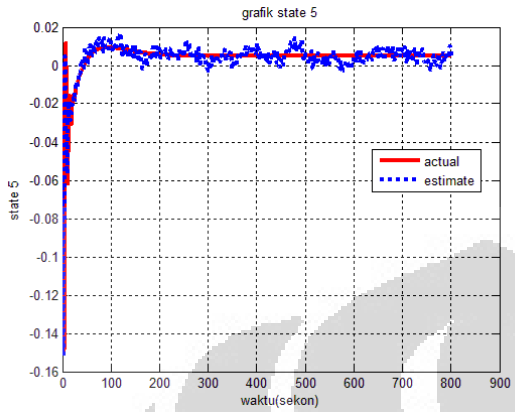
State Kedua Untuk *Spectral Density Gaussian Noise* 0.0001



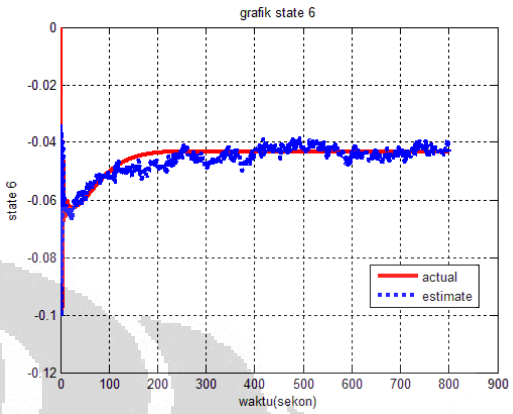
State Pertama Untuk *Spectral Density Gaussian Noise* 0.001



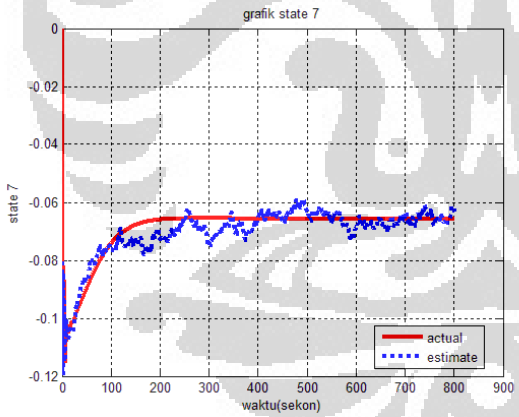
State Pertama Untuk *Spectral Density Gaussian Noise* 0.001



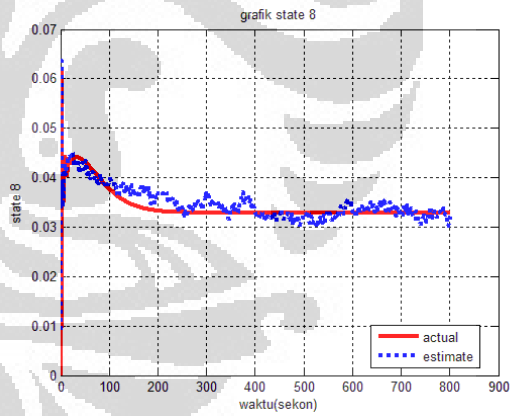
State Kelima
Untuk *Spectral Density*
Gaussian Noise 0.0001



State Keenam
Untuk *Spectral Density*
Gaussian Noise 0.0001

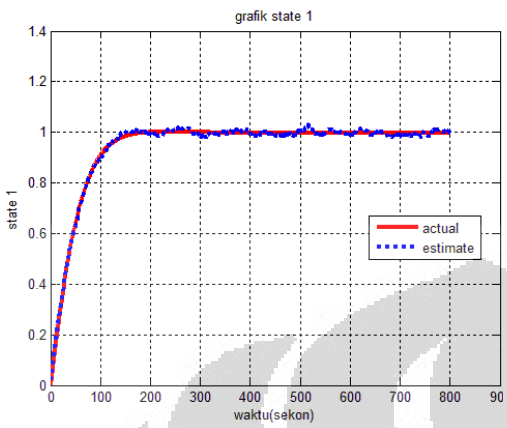


State Ketujuh
Untuk *Spectral Density*
Gaussian Noise 0.0001

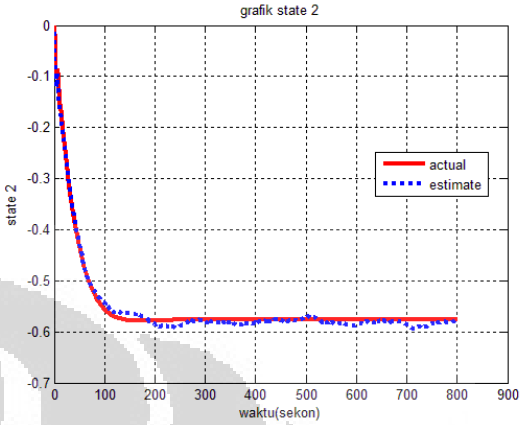


State Kedelapan
Untuk *Spectral Density*
Gaussian Noise 0.0001

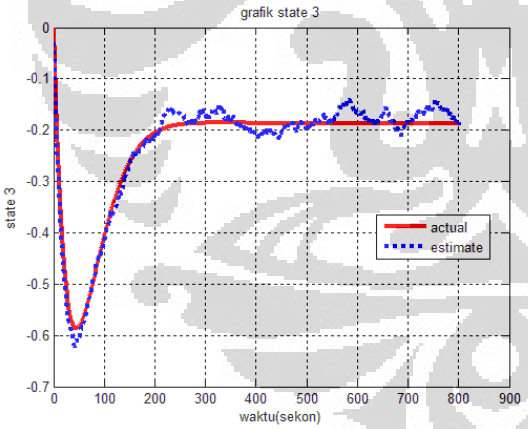
Untuk *Spectral Density Gaussian Noise* 10^{-5}



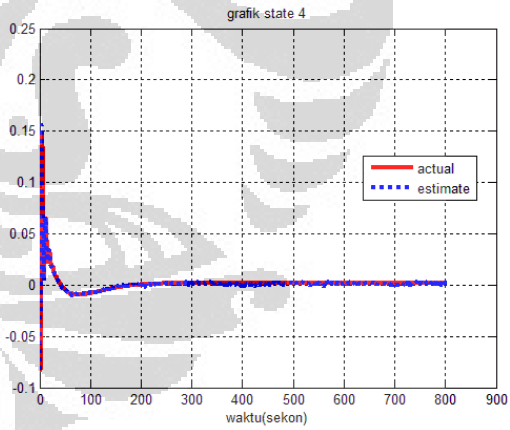
State Pertama
Untuk *Spectral Density Gaussian Noise* 0.00001



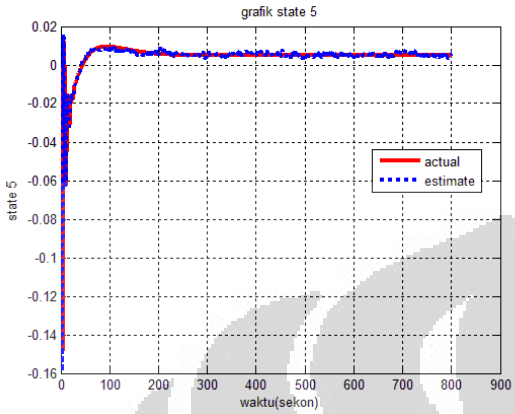
State Kedua
Untuk *Spectral Density Gaussian Noise* 0.00001



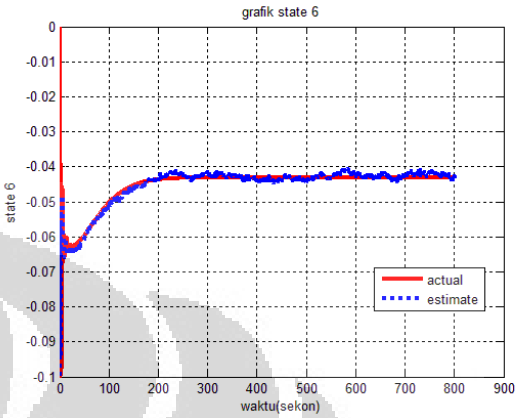
State Ketiga
Untuk *Spectral Density Gaussian Noise* 0.00001



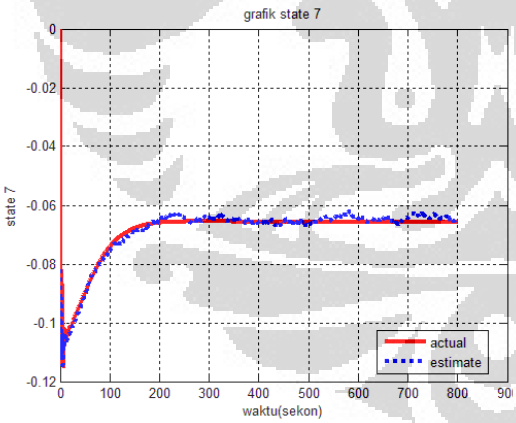
State Keempat
Untuk *Spectral Density Gaussian Noise* 0.00001



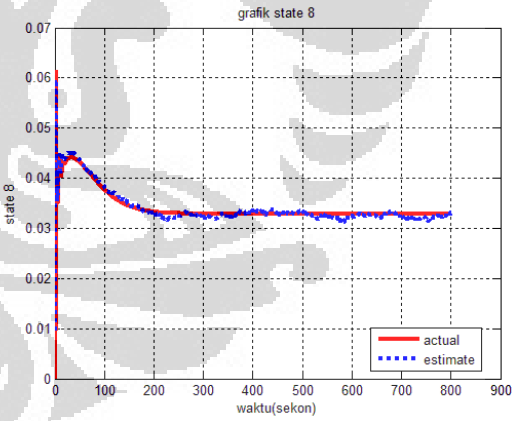
State Kelima
Untuk *Spectral Density Gaussian Noise* 0.00001



State Keenam
Untuk *Spectral Density Gaussian Noise* 0.00001

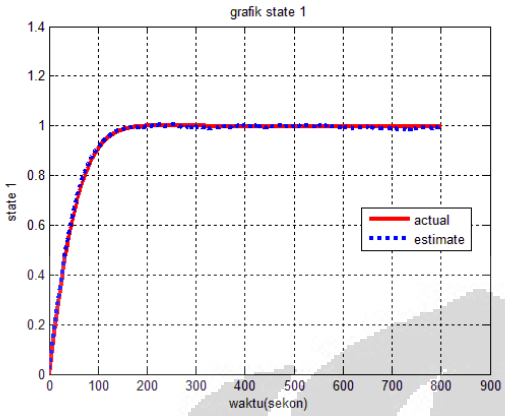


State Ketujuh
Untuk *Spectral Density Gaussian Noise* 0.00001

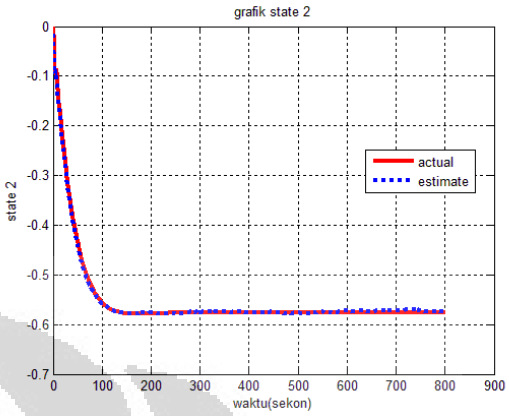


State Kedelapan
Untuk *Spectral Density Gaussian Noise* 0.00001

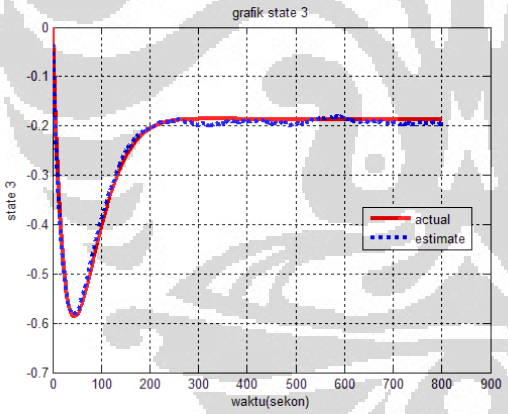
Untuk *Spectral Density Gaussian*
***Noise* 10^{-6}**



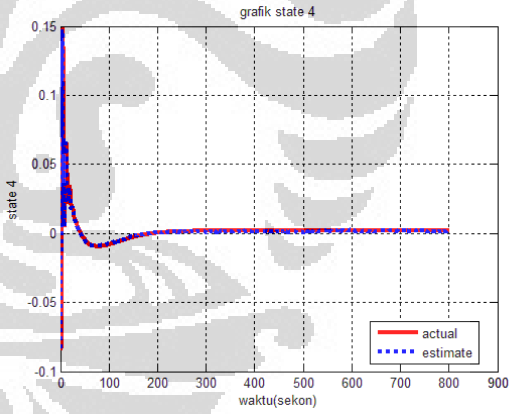
State Pertama
Untuk *Spectral Density Gaussian*
Noise 0.000001



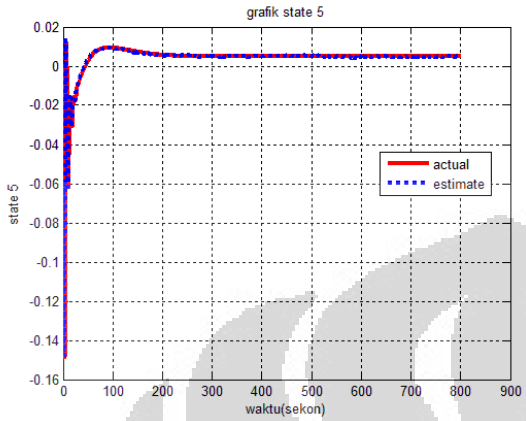
State Kedua
Untuk *Spectral Density Gaussian*
Noise 0.000001



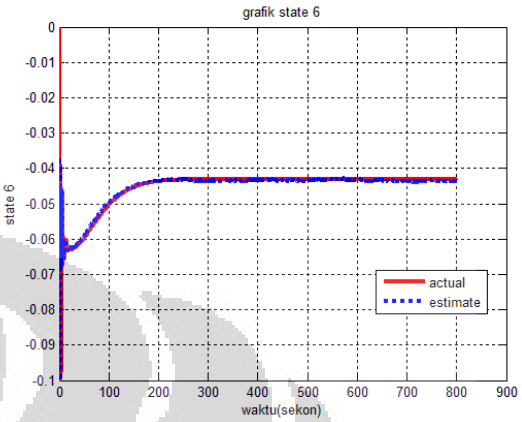
State Ketiga
Untuk *Spectral Density Gaussian*
Noise 0.000001



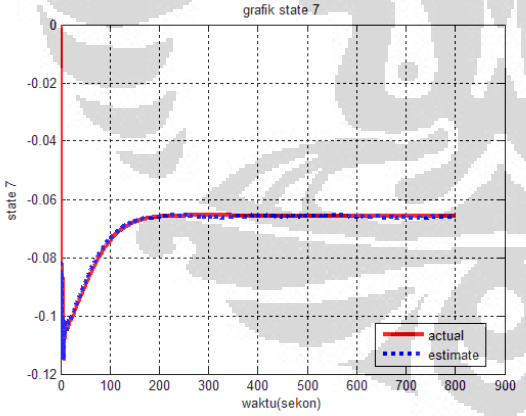
State Keempat
Untuk *Spectral Density Gaussian*
Noise 0.000001



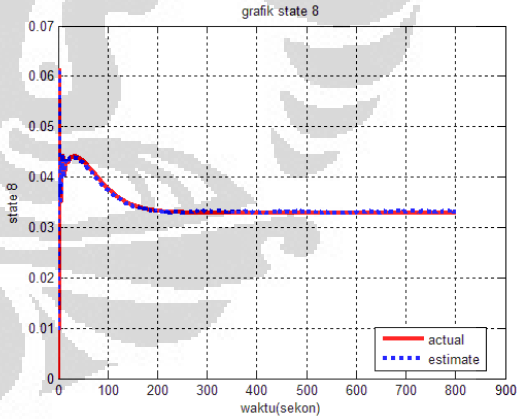
State Kelima
Untuk *Spectral Density Gaussian Noise* 0.000001



State Keenam
Untuk *Spectral Density Gaussian Noise* 0.000001

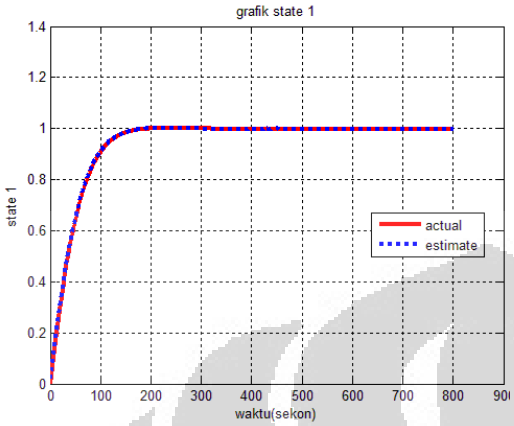


State Ketujuh
Untuk *Spectral Density Gaussian Noise* 0.000001

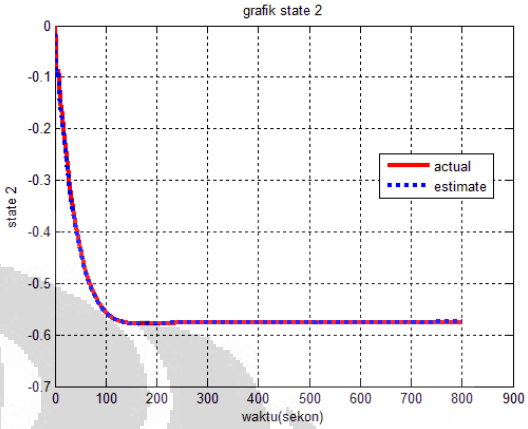


State Kedelapan
Untuk *Spectral Density Gaussian Noise* 0.000001

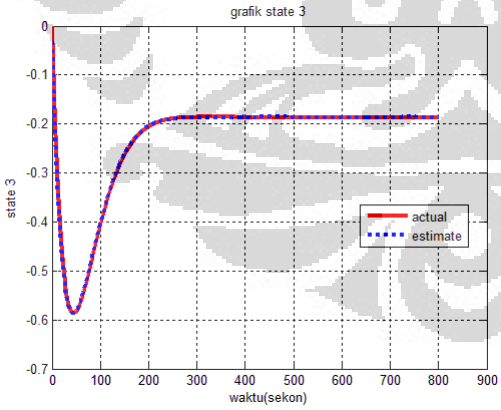
Untuk *Spectral Density Gaussian*
***Noise* 10^{-8}**



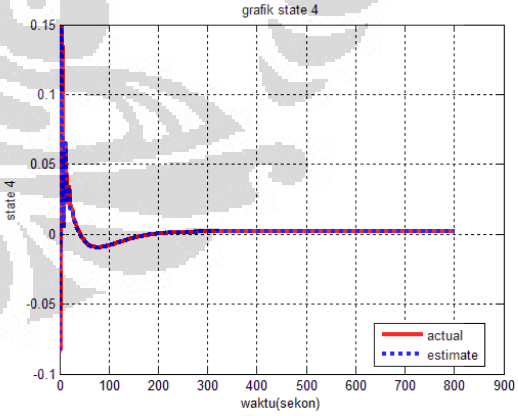
State Pertama
Untuk *Spectral Density Gaussian*
Noise 0.00000001



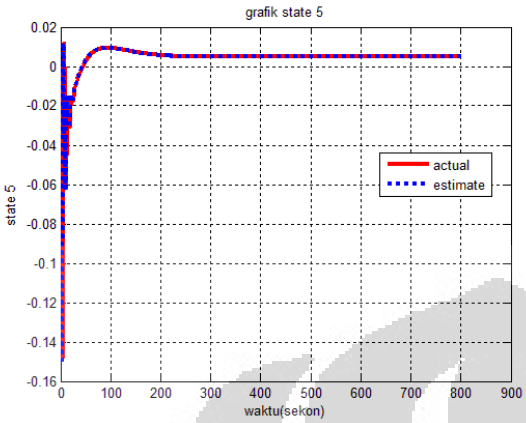
State Kedua
Untuk *Spectral Density Gaussian*
Noise 0.00000001



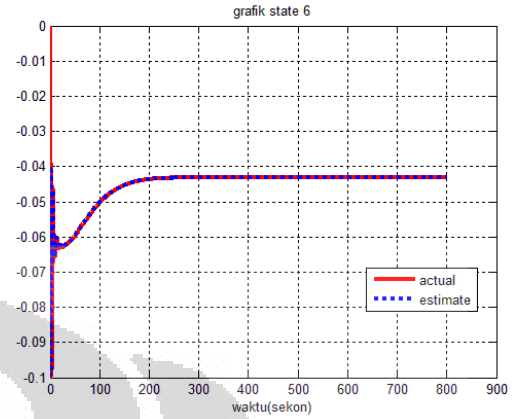
State Ketiga
Untuk *Spectral Density Gaussian*
Noise 0.00000001



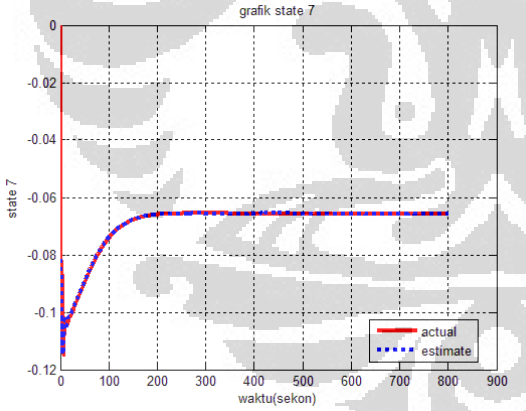
State Keempat
Untuk *Spectral Density Gaussian*
Noise 0.00000001



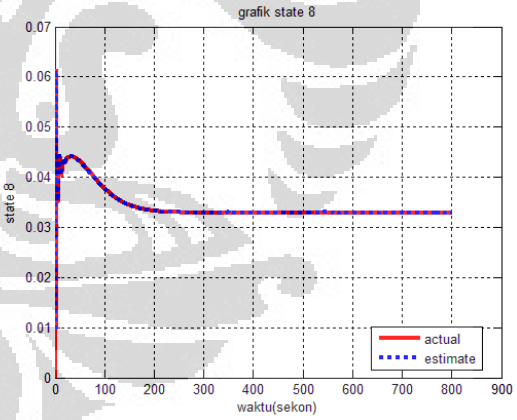
State Kelima
Untuk *Spectral Density Gaussian Noise* 0.00000001



State Keenam
Untuk *Spectral Density Gaussian Noise* 0.00000001

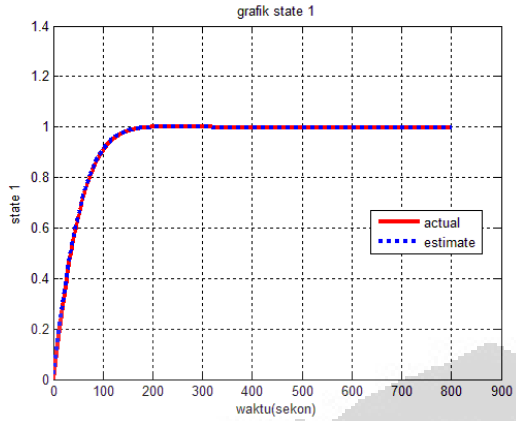


State Ketujuh
Untuk *Spectral Density Gaussian Noise* 0.00000001

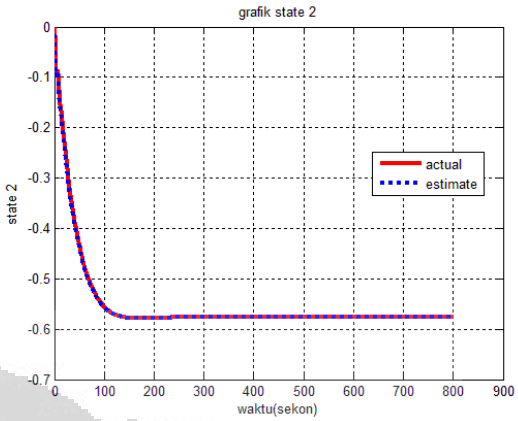


State Kedelapan
Untuk *Spectral Density Gaussian Noise* 0.00000001

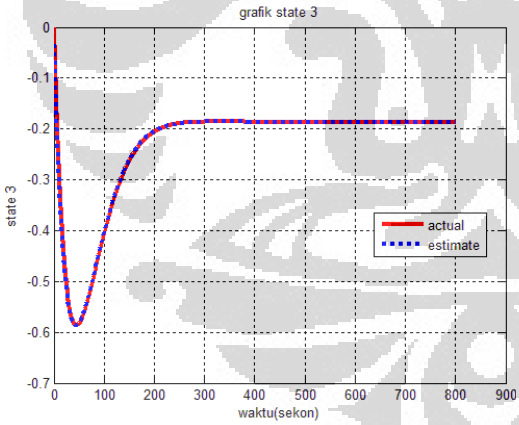
Untuk *Spectral Density Gaussian Noise* Nol



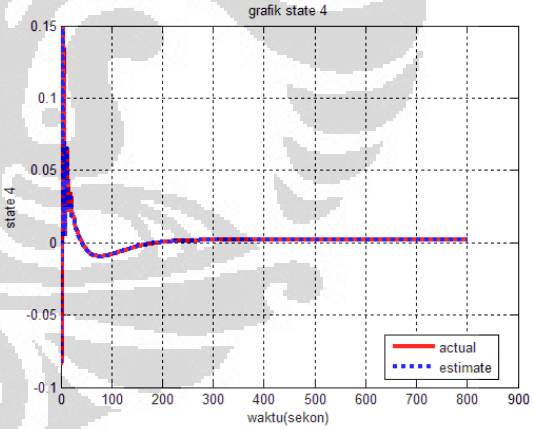
State Pertama
Untuk *Spectral Density*
Gaussian Noise nol



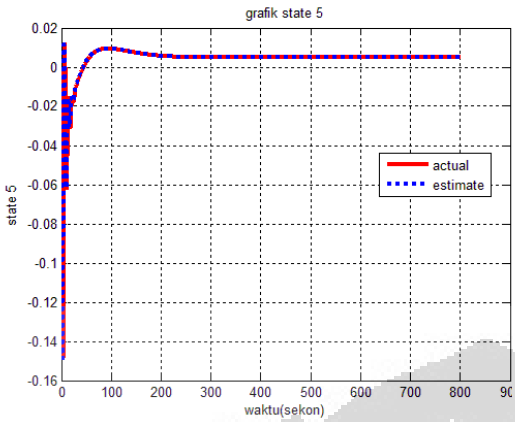
State Kedua
Untuk *Spectral Density*
Gaussian Noise nol



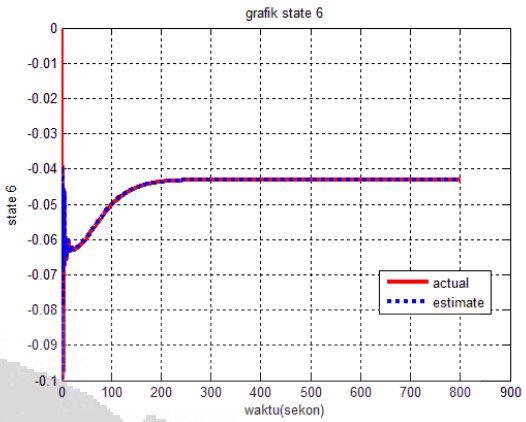
State Ketiga
Untuk *Spectral Density*
Gaussian Noise nol



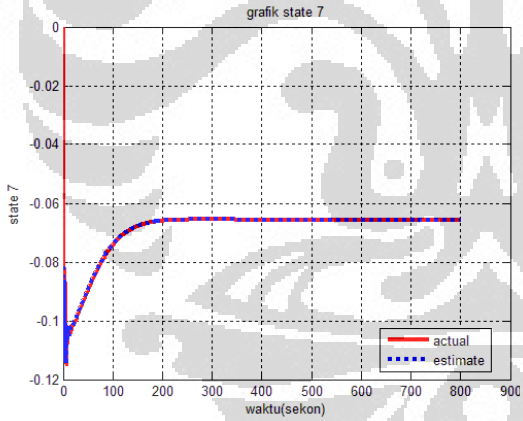
State Kempat
Untuk *Spectral Density*
Gaussian Noise nol



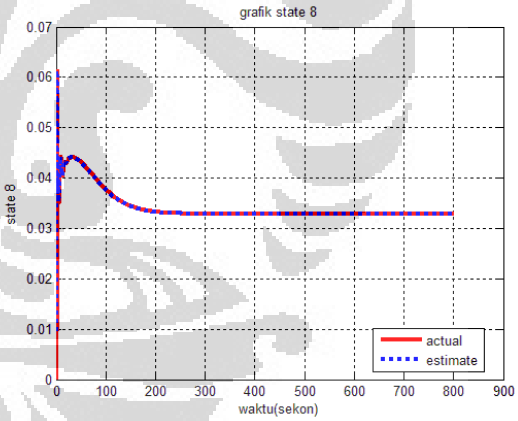
State Kelima
Untuk *Spectral Density*
Gaussian Noise nol



State Keenam
Untuk *Spectral Density*
Gaussian Noise nol



State Kelima
Untuk *Spectral Density*
Gaussian Noise nol

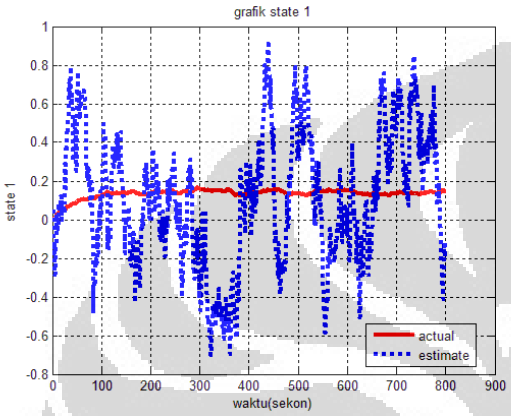


State Keenam
Untuk *Spectral Density*
Gaussian Noise nol

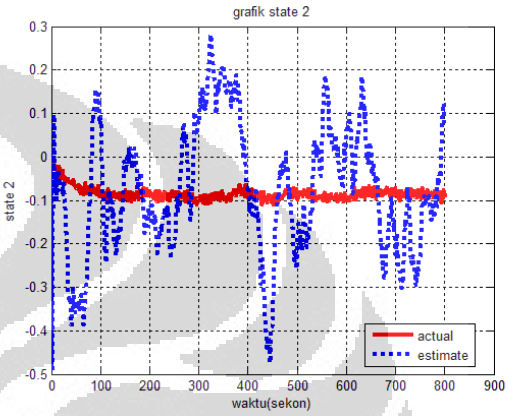
LAMPIRAN B

Grafik Perbandingan *State* Estimasi dengan *State* Sebenarnya dari Sistem Tata Udara Presisi Secara *Open Loop* Menggunakan Sinyal Kendali Data Rekam Sinyal Random dengan Berbagai Variasi *Spectral Density Gaussian Noise*

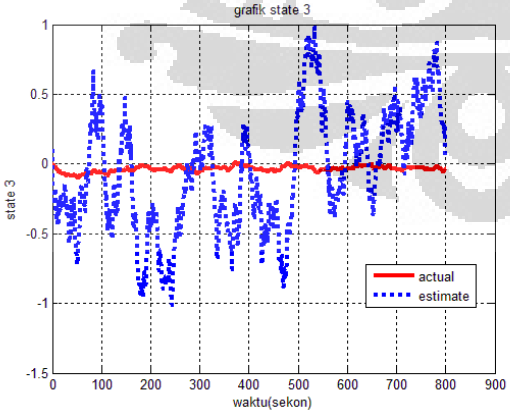
***Spectral Density Gaussian Noise* 0.01**



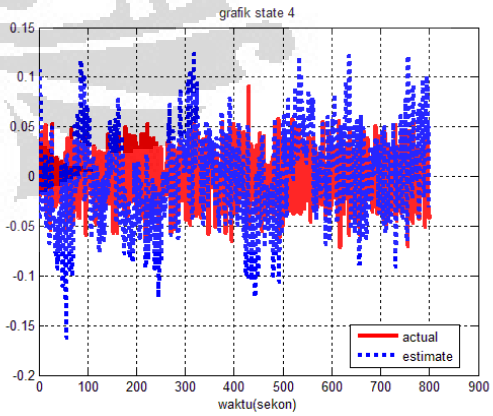
State Pertama
Untuk *Spectral Density Gaussian Noise* 0.01



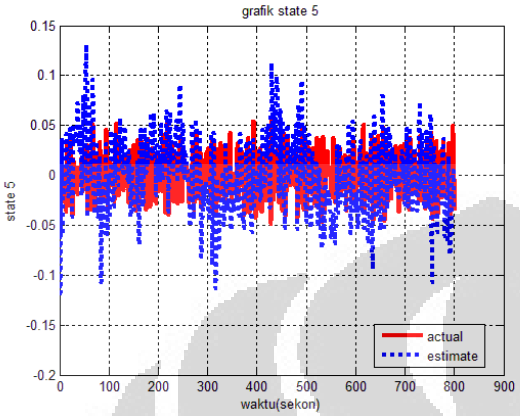
State Kedua
Untuk *Spectral Density Gaussian Noise* 0.01



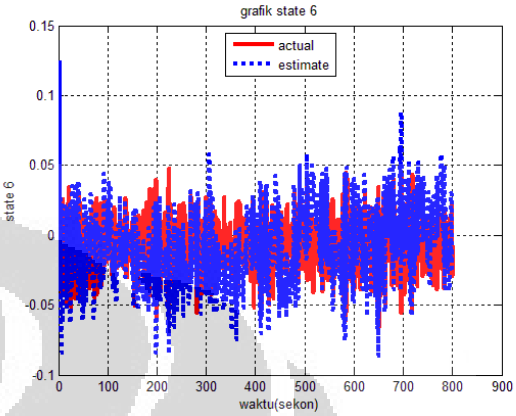
State Ketiga
Untuk *Spectral Density Gaussian Noise* 0.01



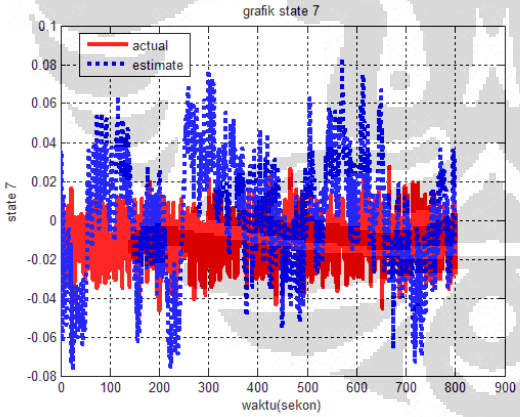
State Keempat
Untuk *Spectral Density Gaussian Noise* 0.01



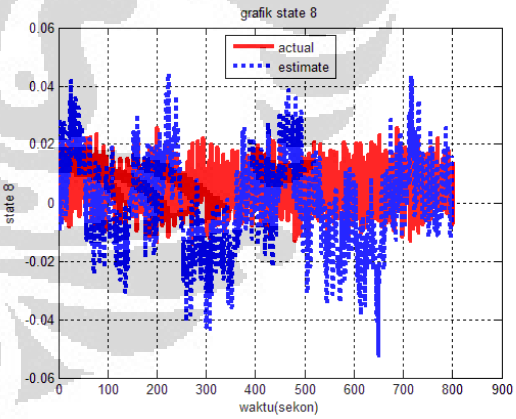
State Kelima
Untuk *Spectral Density*
Gaussian Noise 0.01



State Keenam
Untuk *Spectral Density*
Gaussian Noise 0.01

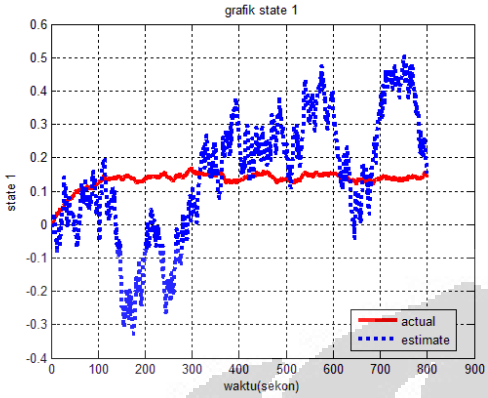


State Ketujuh
Untuk *Spectral Density*
Gaussian Noise 0.01

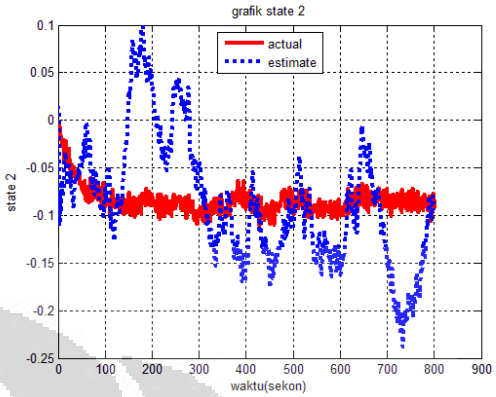


State Kedelapan
Untuk *Spectral Density*
Gaussian Noise 0.01

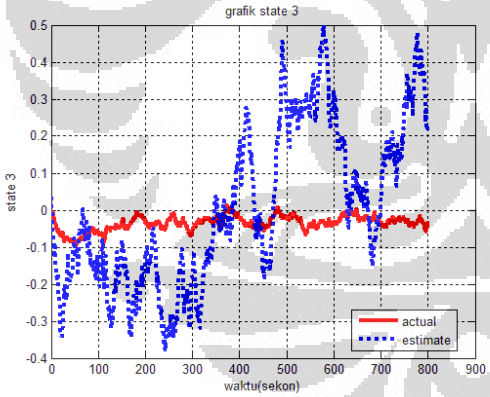
***Spectral Density Gaussian Noise* 0.001**



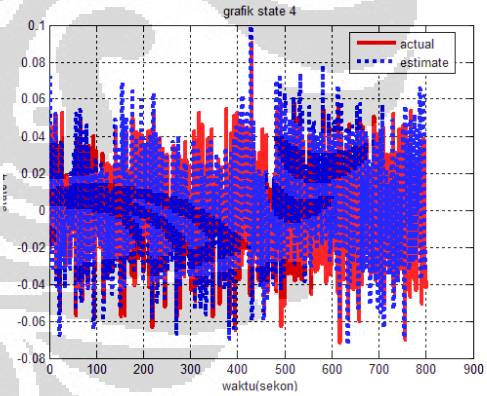
State Pertama
Untuk *Spectral Density Gaussian Noise* 0.001



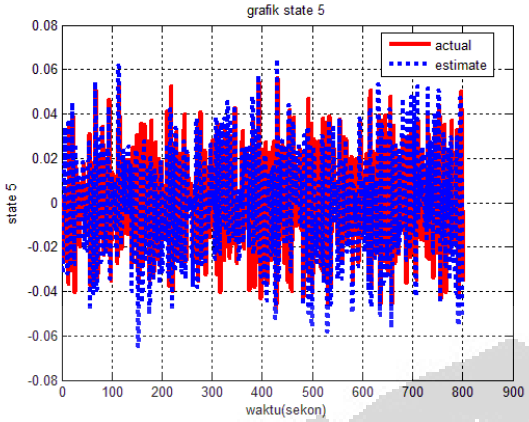
State Kedua
Untuk *Spectral Density Gaussian Noise* 0.001



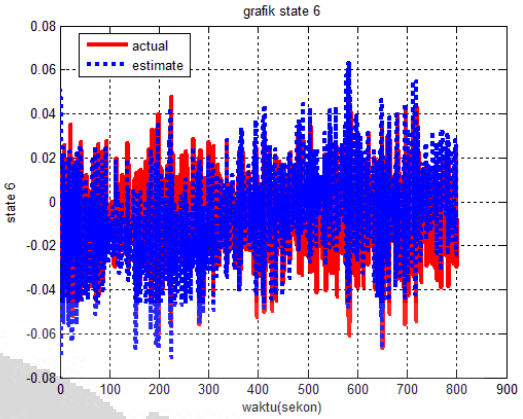
State Ketiga
Untuk *Spectral Density Gaussian Noise* 0.001



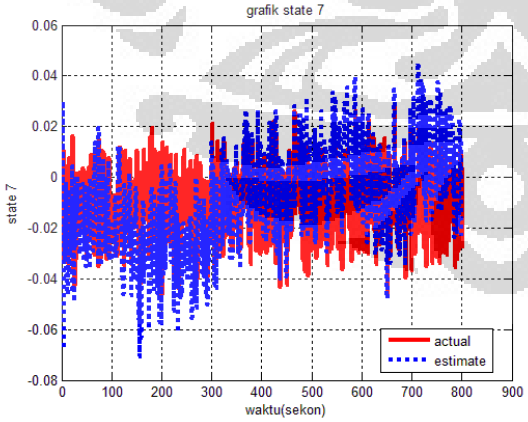
State Keempat
Untuk *Spectral Density Gaussian Noise* 0.001



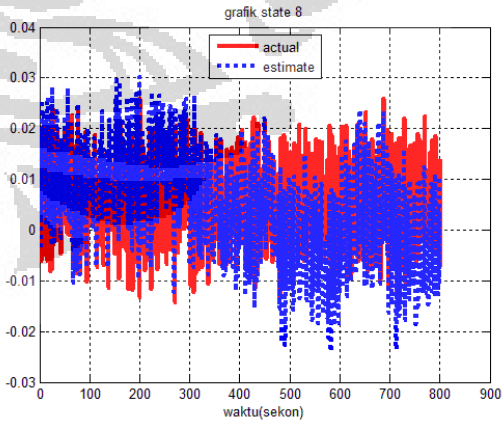
State Kelima
Untuk *Spectral Density*
Gaussian Noise 0.001



State Keenam
Untuk *Spectral Density*
Gaussian Noise 0.001

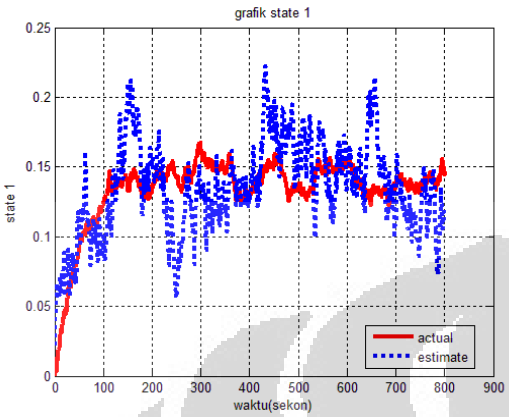


State Ketujuh
Untuk *Spectral Density*
Gaussian Noise 0.001

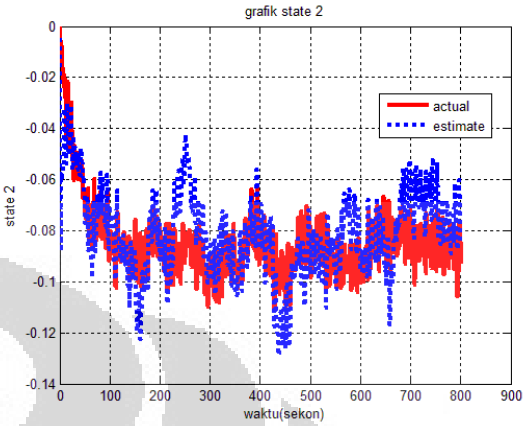


State Kedelapan
Untuk *Spectral Density*
Gaussian Noise 0.001

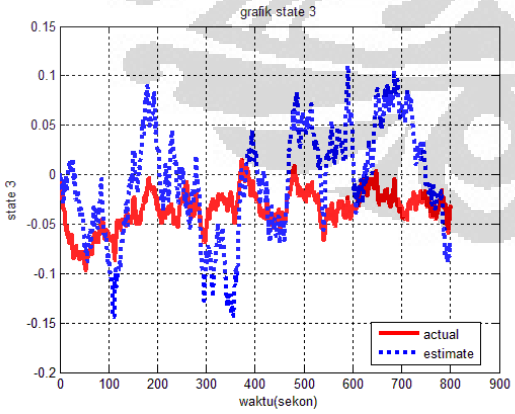
***Spectral Density Gaussian Noise* 0.0001**



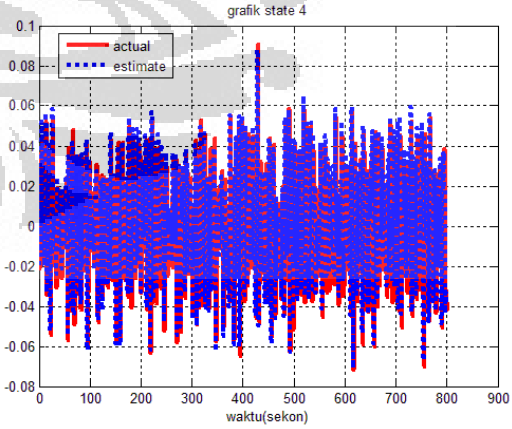
State Pertama
Untuk *Spectral Density*
Gaussian Noise 0.0001



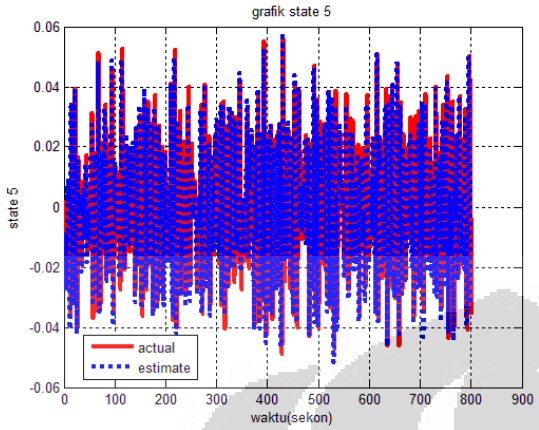
State Kedua
Untuk *Spectral Density*
Gaussian Noise 0.0001



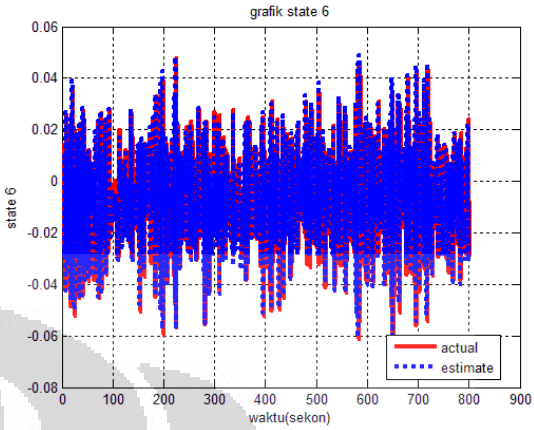
State Ketiga
Untuk *Spectral Density*
Gaussian Noise 0.0001



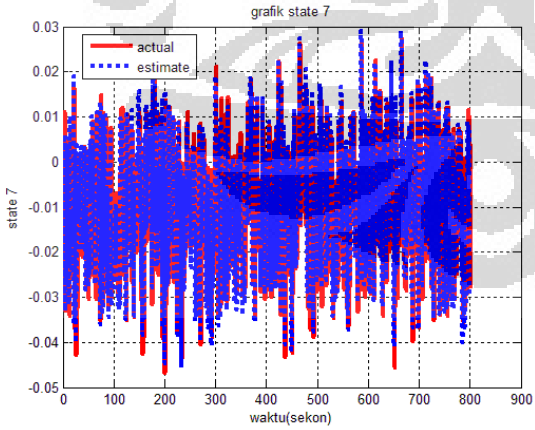
State Keempat
Untuk *Spectral Density*
Gaussian Noise 0.0001



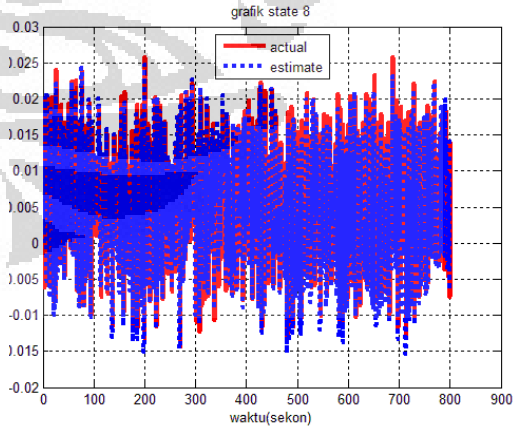
State Kelima
Untuk *Spectral Density*
Gaussian Noise 0.0001



State Keenam
Untuk *Spectral Density*
Gaussian Noise 0.0001

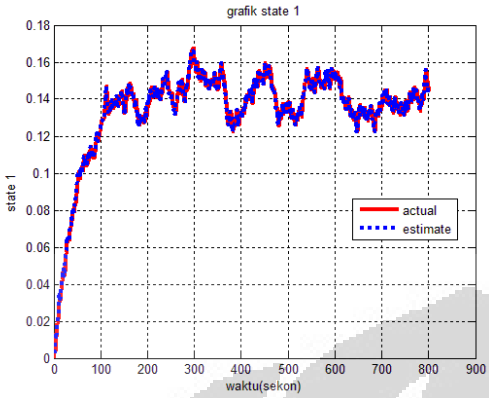


State Ketujuh
Untuk *Spectral Density*
Gaussian Noise 0.0001

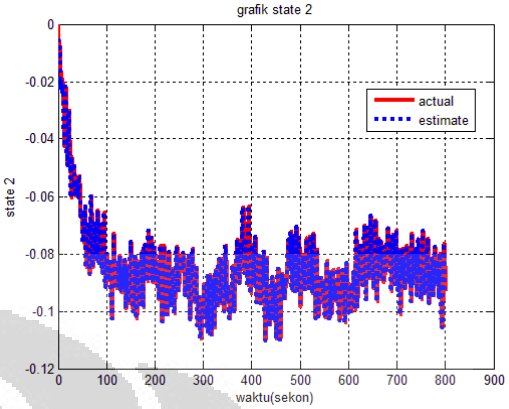


State Kedelapan
Untuk *Spectral Density*
Gaussian Noise 0.0001

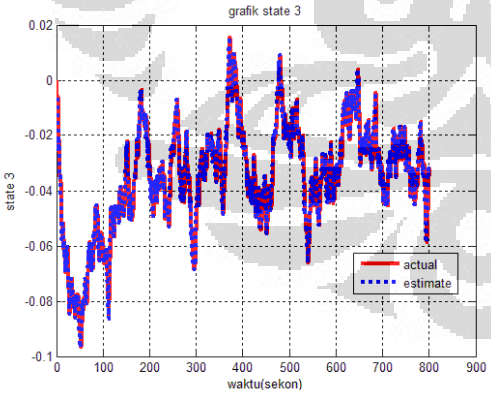
***Spectral Density Gaussian Noise* Nol**



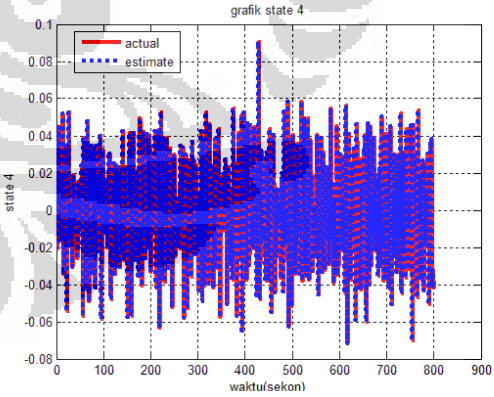
State Pertama
Untuk *Spectral Density*
Gaussian Noise Nol



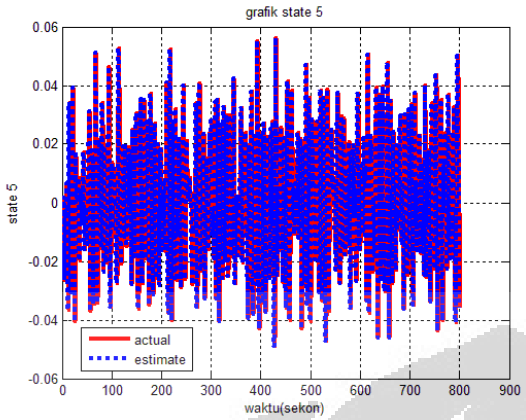
State Kedua
Untuk *Spectral Density*
Gaussian Noise Nol



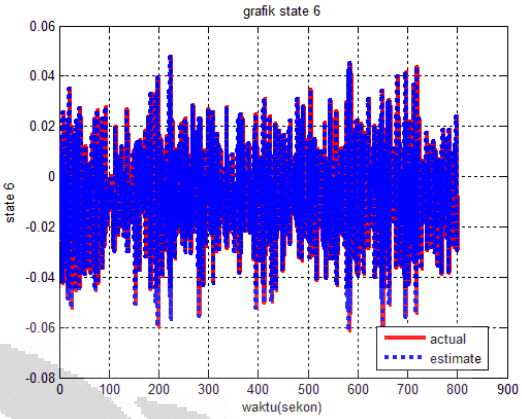
State Ketiga
Untuk *Spectral Density*
Gaussian Noise Nol



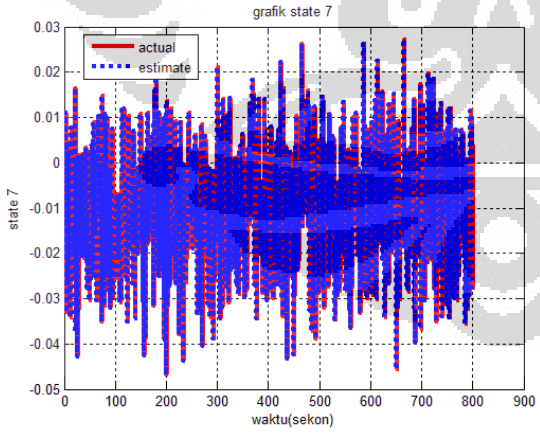
State Keempat
Untuk *Spectral Density*
Gaussian Noise Nol



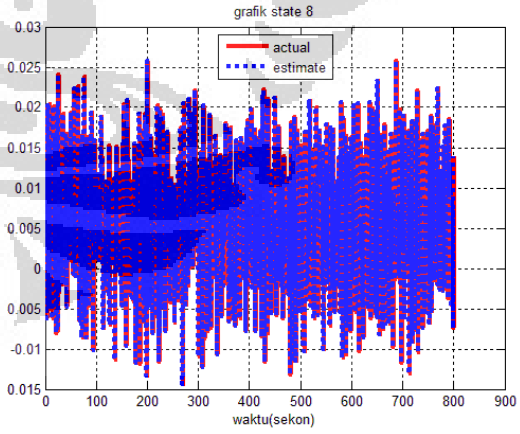
State Kelima
Untuk *Spectral Density*
Gaussian Noise Nol



State Keenam
Untuk *Spectral Density*
Gaussian Noise Nol



State Ketujuh
Untuk *Spectral Density*
Gaussian Noise Nol

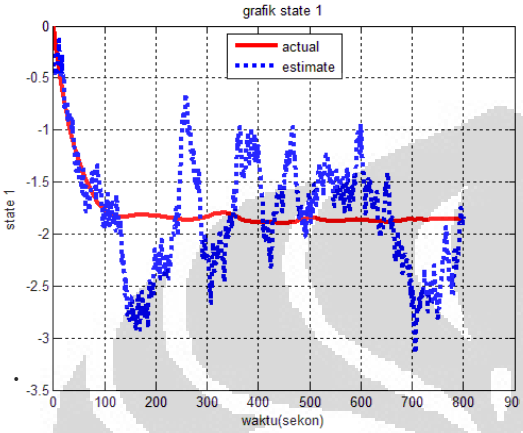


State Kedelapan
Untuk *Spectral Density*
Gaussian Noise Nol

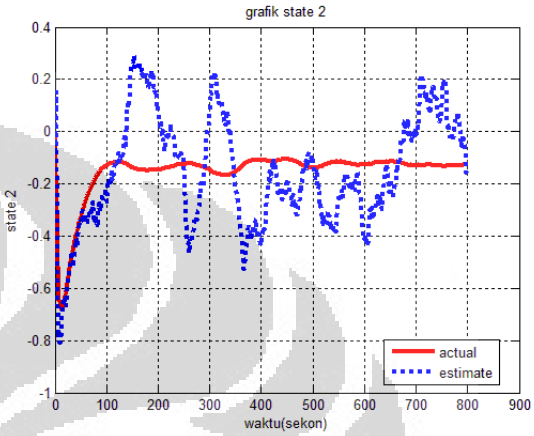
LAMPIRAN C

Grafik Perbandingan *State* Estimasi dengan *State* Sebenarnya dari Sistem Tata Uadara Presisi Secara *Closed Loop* Menggunakan Sinyal Kendali LQR dengan Berbagai Variasi *Spectral Density Gaussian Noise*

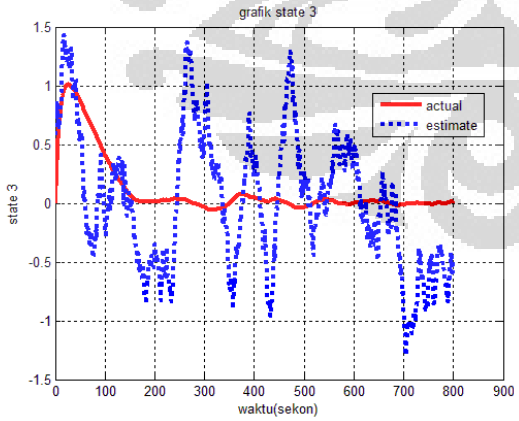
Spectral Density Gaussian Noise 0.01



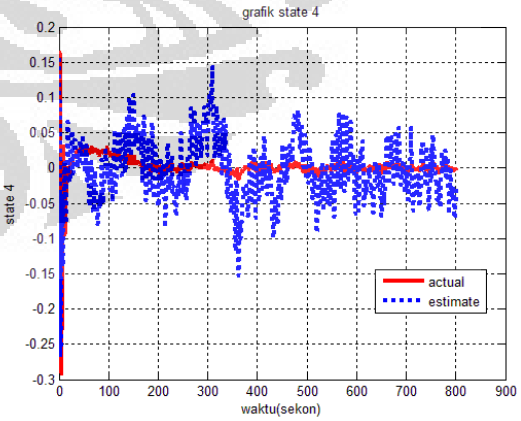
State Pertama
Untuk *Spectral Density Gaussian Noise* 0.01



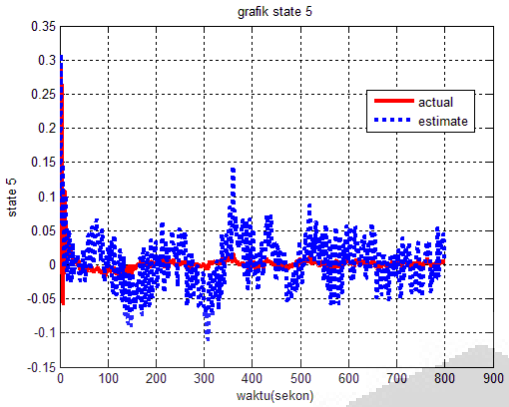
State Kedua
Untuk *Spectral Density Gaussian Noise* 0.01



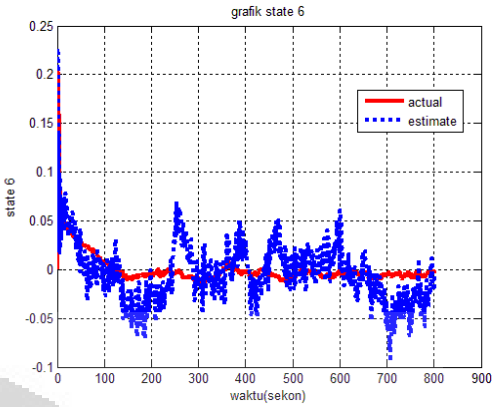
State Ketiga
Untuk *Spectral Density Gaussian Noise* 0.01



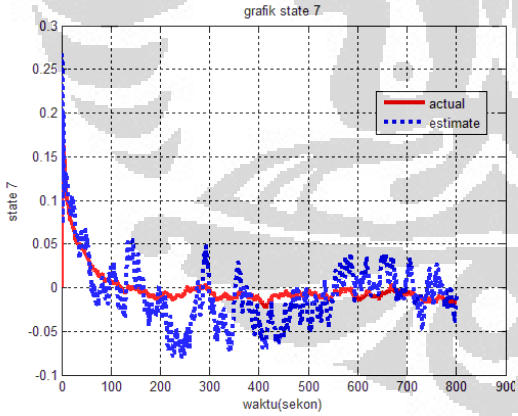
State Keempat
Untuk *Spectral Density Gaussian Noise* 0.01



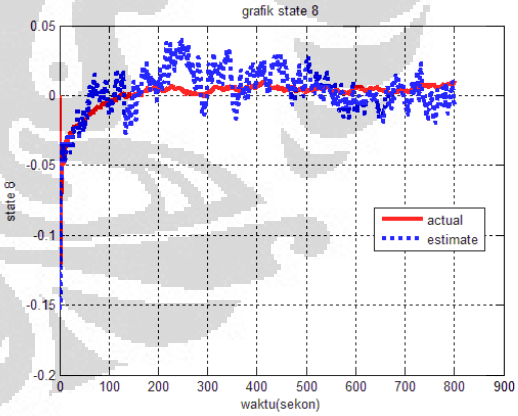
State Kelima
Untuk *Spectral Density*
Gaussian Noise 0.01



State Keenam
Untuk *Spectral Density*
Gaussian Noise 0.01

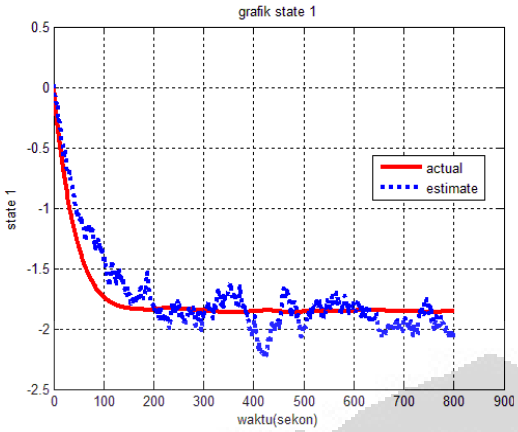


State Ketujuh
Untuk *Spectral Density*
Gaussian Noise 0.01

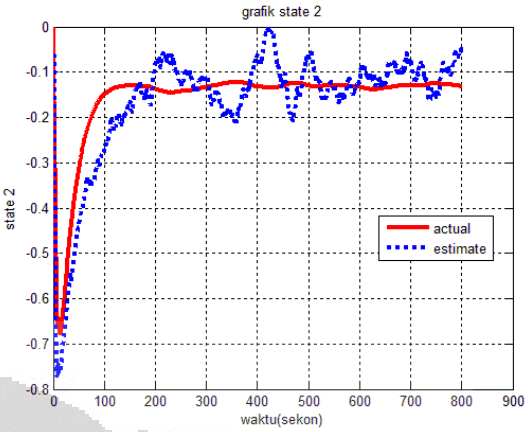


State Kedelapan
Untuk *Spectral Density*
Gaussian Noise 0.01

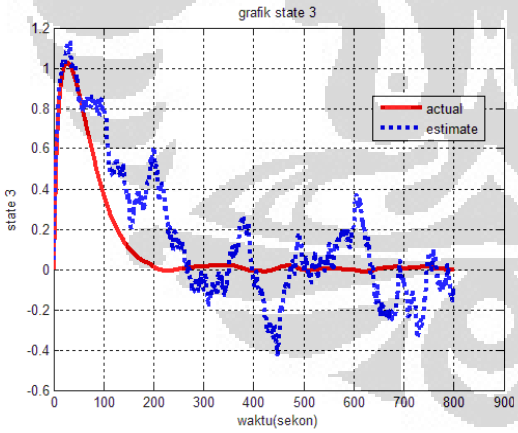
***Spectral Density Gaussian Noise* 0.001**



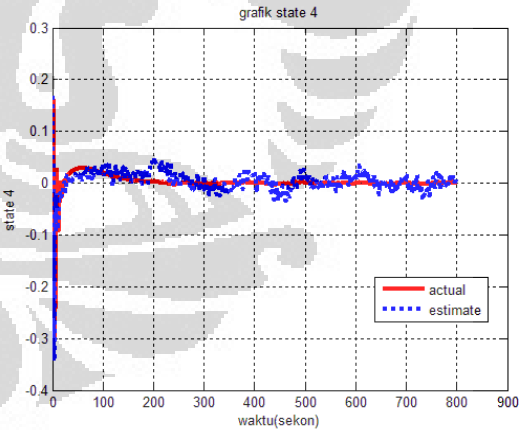
State Pertama
Untuk *Spectral Density*
Gaussian Noise 0.001



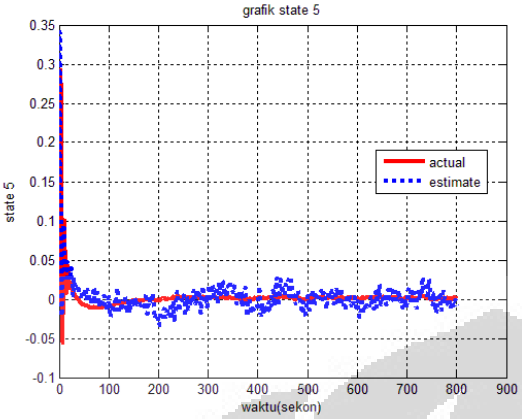
State Kedua
Untuk *Spectral Density*
Gaussian Noise 0.001



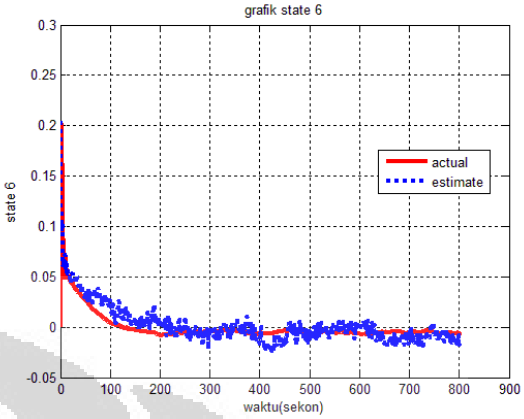
State Ketiga
Untuk *Spectral Density*
Gaussian Noise 0.001



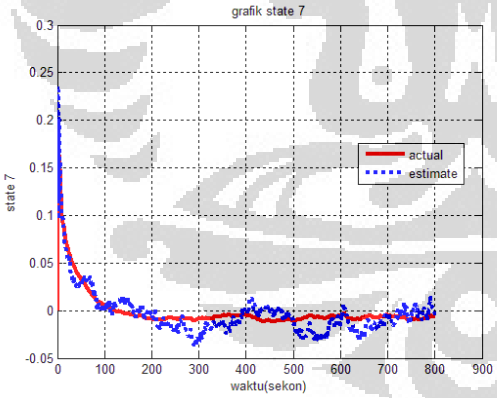
State Keempat
Untuk *Spectral Density*
Gaussian Noise 0.001



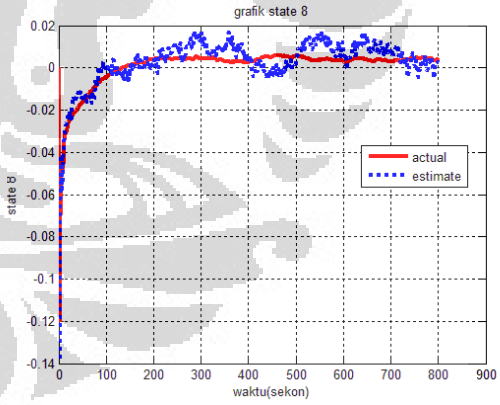
State Kelima
Untuk *Spectral Density*
Gaussian Noise 0.001



State Keenam
Untuk *Spectral Density*
Gaussian Noise 0.001

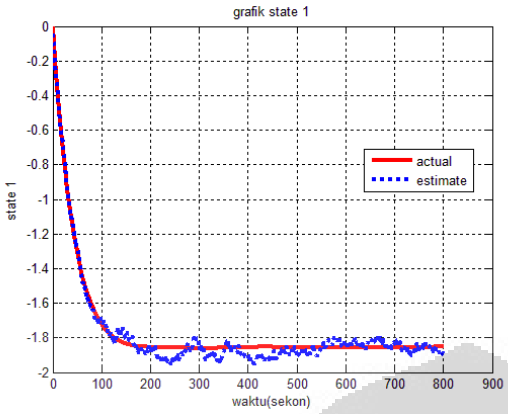


State Ketujuh
Untuk *Spectral Density*
Gaussian Noise 0.001

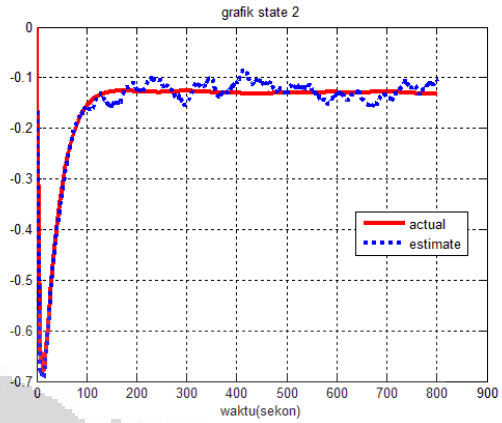


State Kedelapan
Untuk *Spectral Density*
Gaussian Noise 0.001

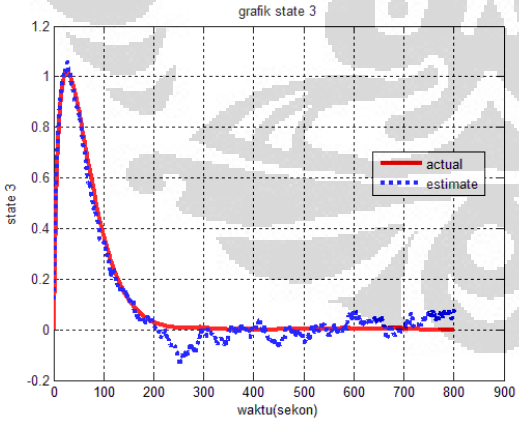
***Spectral Density Gaussian Noise* 0.0001**



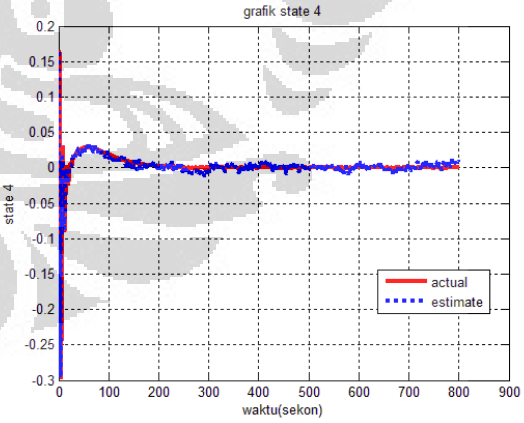
State Pertama
Untuk *Spectral Density*
Gaussian Noise 0.0001



State Kedua
Untuk *Spectral Density*
Gaussian Noise 0.0001

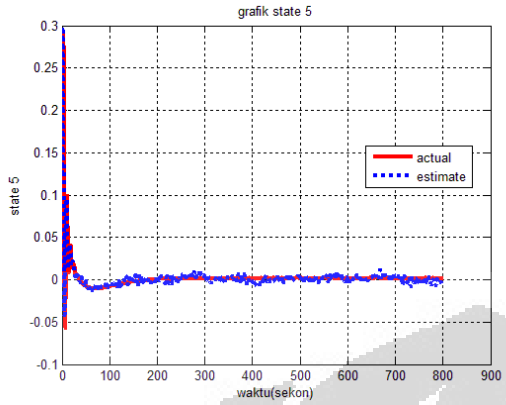


State Ketiga
Untuk *Spectral Density*
Gaussian Noise 0.0001

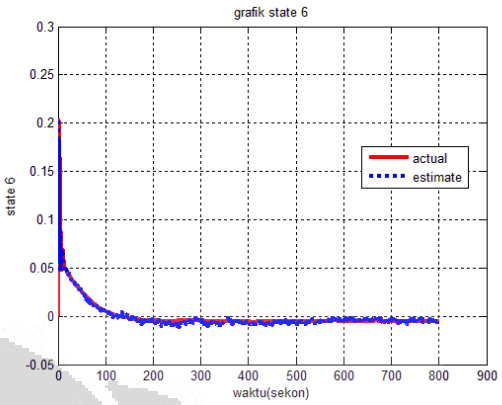


State Keempat
Untuk *Spectral Density*
Gaussian Noise 0.0001

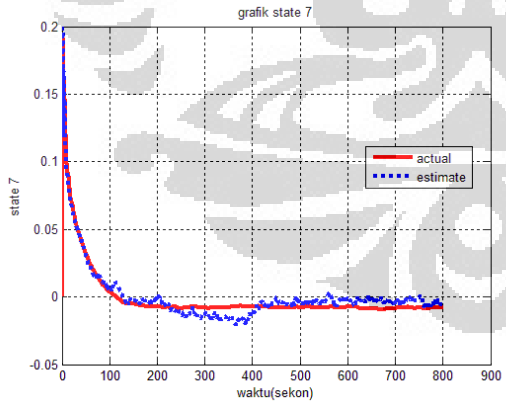
.



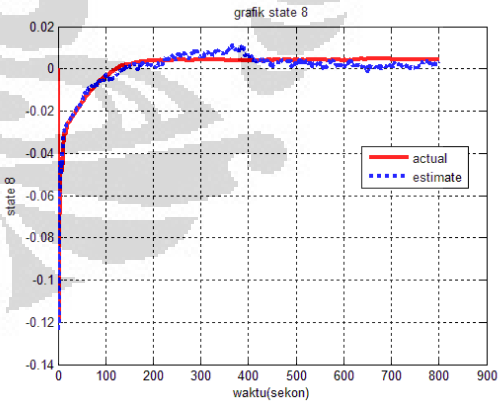
State Kelima
Untuk *Spectral Density*
Gaussian Noise 0.0001



State Keenam
Untuk *Spectral Density*
Gaussian Noise 0.0001

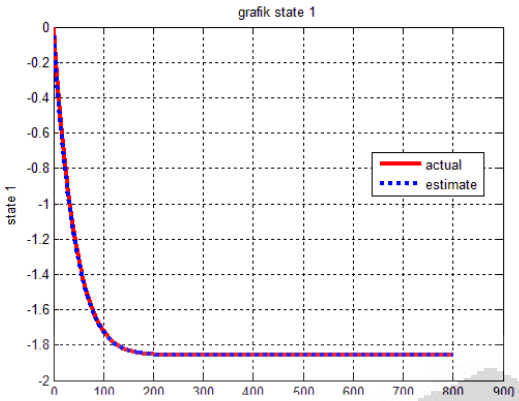


State Ketujuh
Untuk *Spectral Density*
Gaussian Noise 0.0001

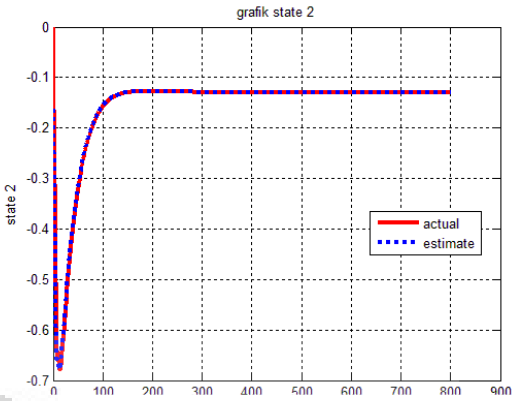


State Kedelapan
Untuk *Spectral Density*
Gaussian Noise 0.0001

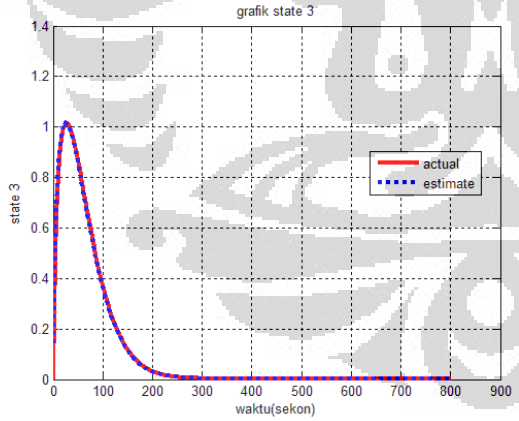
***Spectral Density Gaussian Noise* Nol**



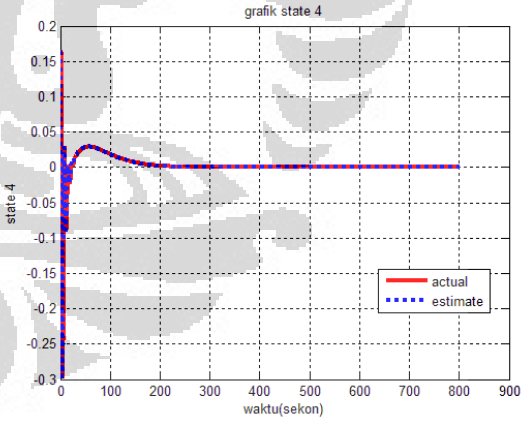
State Pertama
Untuk *Spectral Density Gaussian Noise* Nol



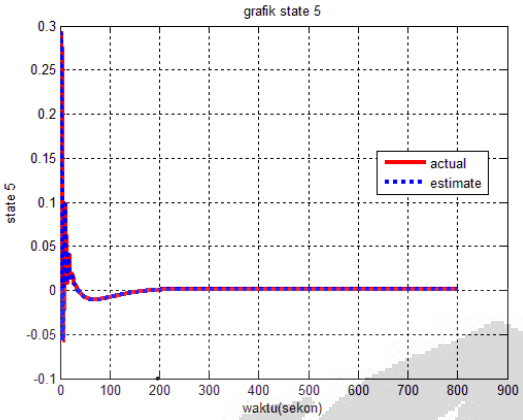
State Kedua
Untuk *Spectral Density Gaussian Noise* Nol



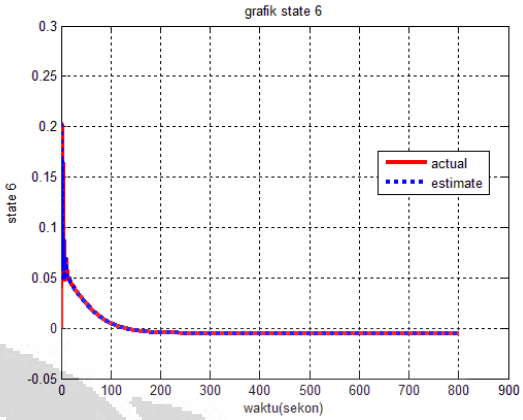
State Ketiga
Untuk *Spectral Density Gaussian Noise* Nol



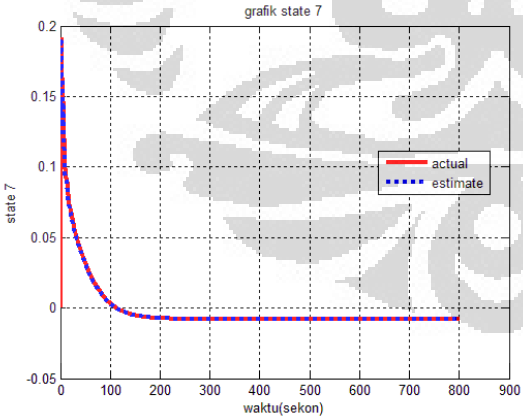
State Keempat
Untuk *Spectral Density Gaussian Noise* Nol



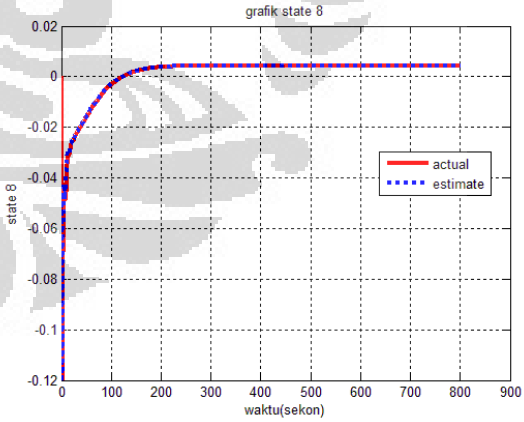
State Kelima
Untuk *Spectral Density*
Gaussian Noise Nol



State Keenam
Untuk *Spectral Density*
Gaussian Noise Nol



State Ketujuh
Untuk *Spectral Density*
Gaussian Noise Nol



State Kedelapan
Untuk *Spectral Density*
Gaussian Noise Nol

LAMPIRAN D

PERBANDINGAN BESARNYA KUADRAT KESALAHAN STATE ESTIMASI SISTEM TATA UDARA PRESISI ANTARA PENGGUNAAN ALGORITMA MATRIKS KOVARIAN DENGAN PENENTUAN SECARA MANUAL

Untuk *Spectral Density Gaussian Noise* 0.0001

No. *State*	Besarnya Kuadrat Kesalahan				
	Algoritma Kovarian	Q = 0.05 X eye(8) R = 0.005 X eye(2)	Q = 0.5 X eye(8) R = 0.005 X eye(2)	Q = 0.005 X eye(8) R = 0.005 X eye(2)	Q = 0.0005 X eye(8) R = 0.005 X eye(2)
State 1	0.0017	0.0014	0.0022	0.0045	0.0018
State 2	3.6804×10^{-4}	2.8982×10^{-4}	3.7365×10^{-4}	4.7387×10^{-4}	3.0330×10^{-4}
State 3	0.0016	0.0021	0.0053	0.0090	0.0016
State 4	1.6506×10^{-5}	1.6422×10^{-5}	2.5254×10^{-5}	2.2595×10^{-5}	1.8086×10^{-5}
State 5	1.0576×10^{-5}	1.2871×10^{-5}	2.0419×10^{-5}	1.0991×10^{-5}	9.5897×10^{-6}
State 6	3.9507×10^{-6}	6.1595×10^{-6}	1.9181×10^{-5}	1.6091×10^{-5}	3.6732×10^{-5}
State 7	6.6218×10^{-6}	7.2399×10^{-6}	1.4283×10^{-5}	2.7211×10^{-5}	1.6899×10^{-5}
State 8	1.3430×10^{-6}	1.7317×10^{-6}	3.3886×10^{-6}	7.0037×10^{-6}	2.0151×10^{-6}

Untuk *Spectral Density Gaussian Noise* 0.01

No. *State*	Besarnya Kuadrat Kesalahan				
	Algoritma Kovarian	Q = 0.05 X eye(8) R = 0.005 X eye(2)	Q = 0.5 X eye(8) R = 0.005 X eye(2)	Q = 0.005 X eye(8) R = 0.005 X eye(2)	Q = 0.0005 X eye(8) R = 0.005 X eye(2)
State 1	0.1413	0.2722	0.2121	0.1881	0.2308
State 2	0.0254	0.0528	0.0415	0.0300	0.0383
State 3	0.2577	0.2649	0.2951	0.3246	0.1301
State 4	0.0018	0.0020	0.0017	0.0019	0.0022
State 5	0.0010	0.0015	0.0018	0.0012	0.0022
State 6	4.7466×10^{-4}	7.2221×10^{-4}	0.0018	6.9232×10^{-4}	0.0012
State 7	7.3702×10^{-4}	0.0011	0.0012	0.0010	0.0012
State 8	1.9790×10^{-4}	2.0978×10^{-4}	2.1737×10^{-4}	2.4307×10^{-4}	1.9930×10^{-4}