TESIS-TE092099

DESAIN KONTROLER PD-LQR DENGAN UPSO UNTUK OPTIMALISASI PENGATURAN *CRANE* ANTI AYUN

MUH. CHAERUR RIJAL
NRP. 2212202001

DOSEN PEMBIMBING
Dr. Ir. Mochammad Rameli
Ir. Rusdhianto Effendie A.K., MT

PROGRAM MAGISTER
BIDANG KEAHLIAN TEKNIK SISTEM PENGATURAN
JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2015

TESIS-TE092099

PD-LQR CONTROLLER DESIGN USING UPSO FOR OPTIMIZING ANTI SWING CRANE CONTROL

MUH. CHAERUR RIJAL
NRP. 2212202001

SUPERVISOR
Dr. Ir. Mochammad Rameli
Ir. Rusdhianto Effendie A.K., MT

MAGISTER PROGRAM
CONTROL SYSTEM ENGINEERING
DEPARTMENT of ELECTRICAL ENGINEERING
FACULTY of INDUSTRIAL TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2015

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Teknik (M.T)
di
Institut Teknologi Sepuluh Nopember

oleh :
MUH. CHAERUR RIJAL
NRP. 2212202001

Tanggal Ujian : 26 Januari 2015
Periode Wisuda : Maret 2015

Disetujui oleh :

1 Dr. Ir. Mochammad Rameli (Pembimbing I)
NIP. 195412271981031002

2 Ir. Rusdhianto Effendie A.K., M.T. (Pembimbing II)
NIP: 195704241985021001

3 Prof. Ir. Abdullah Alkaff, M.Sc, Ph.D. (Penguji)
NIP: 195501231980031002

4 Dr. Trihastuti Agustinah, ST, MT. (Penguji)
NIP: 196808121994032001

5 Ir. Ali Fatoni, MT. (Penguji)
NIP: 196206031989031002

6 Ir. Josaphat Pramudijanto, M.Eng. (Penguji)
NIP: 196210051990031003

Direktur Program Pascasarjana,

Prof. Dr. Adi Soeprajitno,M.T,
NIP: 196404051990021001

DESAIN KONTROLER PD-LQR DENGAN UPSO UNTUK OPTIMALISASI PENGATURAN *CRANE* ANTI AYUN

Nama Mahasiswa	:	Muh. Chaerur Rijal
NRP	:	2212202001
Dosen Pembimbing	:	1. Dr. Ir. Mochamad Rameli
		2. Ir. Rusdhianto Effendie A.K., M.T

ABSTRAK

Crane merupakan salah satu alat yang digunakan untuk mengangkat dan memindahkan muatan pada jalur yang telah ditentukan. Pada saat alat ini bergerak memindahkan beban, maka beban akan terayun dengan besar sudut ayun tertentu mengikuti perubahan kecepatan perpindahan *crane*. Optimalisasi pengaturan diperlukan sehingga alat ini dapat memindahkan beban dengan cepat dengan ayunan.yang sekecil mungkin namun dengan fungsi biaya yang seminimal mungkin.

Pada tesis ini dikembangkan kontroler *crane* anti ayun yang mampu mengoptimalkan fungsi *fitnees* dan syarat kendala yang dihadapi. Kontroler PD-LQR dipilih karena kemampuannya mengatur *plant* dalam daerah *steady state*. Algoritma uPSO dipakai untuk mencari matriks *Q* dan *R* yang tepat untuk mendesain kontroler PD-LQR yang mampu mengkompromikan syarat fungsi biaya, persentase *overshoot*, waktu *settling* dan juga dapat mengurangi ayunan beban dan *steady state error* yang terjadi.

Dari hasil penelitian dan implementasi diperoleh bahwa matriks *Q* dan *R* hasil optimasi dengan uPSO akan menghasilkan kontroler PD-LQR yang lebih baik dalam menjaga ayunan beban *crane*. Sedangkan kontroler PD-LQR *type* I optimasi uPSO akan menurunkan pemakaian energi kontrol rata-rata dan menjaga ayunan beban semakin kecil.

Kata Kunci *: Gantry crane, crane anti ayun, kontrol crane, PD-LQR, LQR type I, LQI, pemilihan Matriks Q dan R, PSO, uPSO, State Space, Riccati Equation.*

-----Halaman ini sengaja dikosongkan-----

PD-LQR CONTROLLER DESIGN USING UPSO FOR OPTIMIZING ANTI SWING CRANE CONTROL

Name of Student	:	Muh. Chaerur Rijal
NRP	:	2212202001
Supervisor	:	1. Dr. Ir. Mochamad Rameli
		2. Ir. Rusdhianto Effendie A.K., M.T

ABSTRACT

Crane is one of the tools used to lift and move the existing load on a predetermined path. When the crane moved the load, it swinged with certain angle depends on speed change of the moving crane. Therefore optimization of settings required to be able moves cranecranes quickly and with swing as small as possible to load but with a minimum cost function is quite eminent.

In this thesis developed anti sway crane controller that is able to optimize the functions and requirements fitnees obstacles encountered. PD-LQR controller is chosen for its ability to regulate plant in steady state area. UPSO algorithm is used to find the matrix Q and R is right for PD-LQR controller design that is capable of compromising the cost function terms, the percentage of overshoot, settling time and also can reduce the load swing and steady state error that occurred.

This research implies that optimization of the matrix Q and R with uPSO would produce better PD-LQR controller in order to maintain the crane's swing load. While PD-LQR controller type I uPSO optimization would reduce the average energy consumption control and maintain smaller swing load.

Keywords: *Gantry cranes, anti sway crane, crane control, PD-LQR, LQR type I, LQI, the selection of matrix Q and R, PSO, uPSO, State Space, Riccati Equation.*

-----Halaman ini sengaja dikosongkan-----

KATA PENGANTAR

Alhamdulillah, puji syukur kehadirat Allah SWT yang telah memuliakan dan meninggikan derajat orang-orang berilmu pengetahuan, karena atas rahmat-Nya sehingga penulis dapat menyelesaikan tesis dengan judul:

"Desain Kontroler PD-LQR dengan uPSO untuk Optimalisasi Pengaturan *Crane* Anti Ayun"

Tesis ini merupakan bagian dari proses pembelajaran, khususnya bagi diri penulis sebagai syarat mendapatkan gelar Magister Teknik. Penyelesaian penelitian ini tidak terlepas dari bantuan banyak pihak, oleh sebab itu ucapan terima kasih yang tulus penulis sampaikan kepada:

1. Ayahanda Muh. Rum Arfah (Alm) beserta Ibunda Sugihati. Terima kasih atas cinta, kasih sayang, pengorbanan dan keikhlasan do'a yang telah dicurahkan.
2. Istri tercinta beserta anak-anak tersayang yang sabar menunggu dirumah.
3. Kedua dosen pembimbing, Bapak Dr. Ir. Mochammad Rameli dan Bapak Ir.Rusdhianto Effendie A.K., M.T.
4. Seluruh staf pengajar Program Pascasarjana Teknik Sistem Pengaturan FTI-ITS terkhusus Ibu Dr. Trihastuti Agustinah, M.T selaku Penasehat Akademik mahasiswa S2 TSP angkatan 2012.
5. Teman-teman TSP S2 angkatan 2012 yang banyak memberikan motivasi dan bantuan selama masa studi.
6. Rekan-rekan sejawat di kampus Politeknik Negeri Ujung Pandang.

Kritik dan saran dari semua pihak sangat diharapkan untuk peningkatan kualitas penelitian dan penulisan dimasa yang akan datang. Semoga Allah pemilik segala ilmu mencurahkan hidayah dan bimbingan-Nya kepada kita semua.

-----Halaman ini sengaja dikosongkan-----

DAFTAR ISI

-----Halaman ini sengaja dikosongkan-----

DAFTAR TABEL

-----Halaman ini sengaja dikosongkan-----

DAFTAR GAMBAR

DAFTAR PUSTAKA

1. E. Raubar, dan D. Vrancic, "Anti-Sway System for Ship-to-Shore Cranes", *Journal of Mechanical Engineering* vol. 58, 2012.
2. K.T. Hong, dan C.D. Huh, "Command Shaping Control for Limiting the Transient Sway Angle of Crane Systems", *Int. Journal of Control, Automation and System*, vol. 1, no. 1, 2003.
3. A. H. Keith, and E. S. William, "A Feedback Control System for Suppressing Crane Oscillations with On-Off Motor", *Intl. Journal of Control, Automation and System*, vol.5 no.3, Juni 2012.
4. Y.S. Kim, dan K.S. Hong, "Anti-Sway Control of Container Crane : Inclinometer, Observer, and State Feedback", *Int. Journal of Control, Automation and System*, vol. 2, no. 4, Desember, 2004.
5. M.A. Ahmad, "Sway Reduction on Gantry Crane System using Delayed Feedback Signal and PD-type Fuzzy Logic Controller: A Comparative Assessment", *Proceedings: Word Academy of Science, Engineering and Technology* 26, 2009.
6. J. Smoczek, dan J. Szpytko, "The Fuzzy Robust Anti-Sway Crane Control System", *Journal of KONBIN* 1-2, 2009.
7. M. Mahrueyan, dan H. Khaloozadeh, "Designing a Nonlinear Optimal Ani-Sway Controller for Container Crane System", *International Conference on Circuits, System and Simulation IPCSIT* vol.7, Singapore, 2011.
8. J. Smoczek, dan J. Szpytko, "Design of *Gain* Scheduling Anti-Sway Crane Controler using Genetic Fuzzy System", *IEEE Int. Jounal of Control,* 2012.
9. M.A Ahmad, dan M.S. Ramli, "Control Schemes for Input Tracking and Anti-Sway Control of a Gantry Crane", *Australian Journal of Basic and Applied Sciences* 4(8), 2010.
10. I. Rokhim, "Pengaturan Anti Swing pada Gantry Crane Menggunakan Sliding Mode Control dengan Kompensator Proporsional Integral", *Tesis*, Pascasarjana Jurusan Teknik Elektro-ITS, Surabaya, 2012.
11. F.L. Lewis, "*Optimal Control*", John-Wiley, 1986.
12. R. Effendi, "*Sistem Pengaturan Optimal*", tidak diterbitkan, Surabaya, 2012.

13. F.L. Lewis "*Applied Optimal Control & Estimation*",Prentice-Hall, 1992
14. F.Z. Ping, Z.J Mao,dan R. Allen, "Design of Continuous and Discrete LQI Control *System*s with Stable Inner Loops", *Journal of Shanghai Jiaotong University*, Vol .E212, No. 6, 2007.
15. R.M Murray, "*Control and Dynamical Systems*", California Institute of Technology, 2006.
16. C.F. Juang, C.M. Hsiao, dan C.H. Hsu, "Hierarchical Cluster Based Multispecies Particle Swarm Optimization for Fuzzy-System Optimization", *IEEE Trans Actions On Fuzzy Systems*, VOL.18, NO.1, 2010.
17. K. Hassani, dan W.S. Lee, "Optimal Tuning of Linear Quadratic Regulator Using Quantum Particle Swarm Optimization", *Proceeding of Int. Conference of Control, Dynamics System and Robotics*, Ottawa, Canada, 2014.
18. K.E. Parsopoulos., dan M.N. Vrahatis., "Unified Particle Swarm Optimization for Solving Constrained Engineering Optimization Problems", *Proceedings of International Conference, ICNC 2005*, Changsha, China, 2005.
19. K.E. Parsopoulos., dan M.N. Vrahatis., "Parameter Selection and Adaptation in Unified Particle Swarm Optimization", *Journal of Mathematical and Computer Modelling*, 46, 2007.
20. *Digital Pendulum Feedback Instrument Ltd., Control in a* MATLAB *environment, Feedback Instrument Ltd*, England, 2004.
21. A. Ashfahani, T. Agustinah, dan A. Jazidie, "Kontrol Tracking pada Sistem Pendulum Terbalik Berbasis Model Fuzzy Takagi-Sugeno Menggunakan Pendekatan BMI", *Proseding Seminar Tugas Akhir Jurusan Teknik Elektro*, ITS Surabaya, 2012.

RIWAYAT HIDUP

Muh. Chaerur Rijal, dilahirkan dan besar di kota Makassar tanggal 7 Oktober 1981. Penulis merupakan putra kedua dari enam bersaudara dari pasangan Muh, Roem Arfah (Alm) dan Sugihati. Pendidikan formal yang ditempuh antara lain, TK PG. Takalar, SD PG. Takalar, lulus pada tahun 1993, SMP Negeri 1 Makassar, Lulus tahun 1996, SMU Negeri 2 Makassar, lulus pada tahun 1999, Teknik Elektro, Fakultas Teknik, Universitas Hasanuddin Makassar, lulus pada tahun 2005, Saat buku ini disusun, penulis sedang melanjutkan studinya pada Pasca Sarjana Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember Surabaya dan pada bulan Januari 2015 telah mempertanggung-jawabkan Tesis dalam seminar dan ujian Tesis di bidang studi Teknik Sistem Pengaturan, Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknlogi Sepuluh Nopember Surabaya.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Crane merupakan salah satu alat yang banyak digunakan sebagai alat angkat yang digerakkan baik secara manual maupun semi-otomatis untuk memindahkan beban berat dalam kegiatan industri-industri besar seperti pada pelabuhan laut, pabrik dan konstruksi bangunan. Berdasarkan jenisnya, dapat digolongkan menjadi tiga jenis utama yaitu *mobile crane, gantry crane,* dan *tower arm crane*. Di industri perkapalan utamanya di pelabuhan peti kemas banyak digunakan jenis *gantry crane* yang berfungsi khusus untuk memindahkan barang, baik berupa peti kemas, kargo, maupun mobil ke suatu tempat yang diinginkan baik ke dalam maupun ke luar kapal. Sedangkan pada proyek-proyek konstruksi gedung bertingkat umumnya dipakai jenis *tower arm crane*. Dalam proses memindahkan barang atau peti kemas tersebut umumnya dilakukan oleh operator yang berpengalaman.

Tujuan utama dari pengaturan *crane* adalah memindahkan beban dari satu titik ke titik yang lain secepat mungkin tanpa menimbulkan adanya ayunan yang tidak terkendali pada posisi titik akhir. Sebagai alat pemindah barang, alat ini harus dapat bergerak cepat untuk memindahkan beban tanpa mengakibatkan goyangan yang terlalu besar pada beban yang dipindahkan. Pada saat alat ini bergerak untuk memindahkan beban, maka beban akan terayun dengan besar sudut ayun tertentu, mengikuti perubahan kecepatan akibat kombinasi dari energi gerak dan sudut ayun beban yang besar pada saat perpindahan. Beban pada *crane* juga akan mengalami ayunan saat posisi akhir sebagai akibat berhenti tiba-tiba setelah bergerak cepat. Meskipun waktu perpindahan yang minimal dapat tercapai dengan kecepatan tinggi, tetapi akan mengakibatkan sudut ayun yang terlalu besar. Pengendalian kecepatan *crane* dibutuhkan untuk dapat memindahkan alat ini dengan cepat serta dengan ayunan (osilasi) sekecil mungkin pada beban. Akibat pertimbangan operasi tersebut, kinerja alat ini sangat bergantung pada keahlian dan pengalaman operator untuk memperhatikan posisi serta goyangan yang dihasilkan dari perpindahannya.

Operator diharuskan mempunyai *skill* yang cukup untuk melihat perubahan ayunan beban dan meresponnya. Hanya operator yang berpengalaman yang dapat memindahkan barang dengan tepat. Dalam kenyataannya dengan pengaturan secara manual kemungkinan posisi barang yang tidak tepat dan berayun sangat sering terjadi.

Pengaturan manual sangat sulit dan membutuhkan keterampilan khusus dikarenakan *plant* ini merupakan sistem nonlinear dengan *single-input multi-output*, sehingga banyak dikembangkan metode-metode kontrol untuk membantu pengaturan *crane*.

Pengaturan *crane* secara otomatis telah banyak diimplementasikan oleh para peneliti menggunakan berbagai macam metode. Umumnya pengendalian *crane* anti ayunan terbagi menjadi dua kelompok yaitu *open loop* dan *close loop system* [l]. *Close loop system* didasarkan pada informasi umpan balik terkini berupa sudut ayunan beban, posisi *trolley* dan kecepatannya. Sedangkan *open loop system* berkerja dengan metode aksi *feed-forward* [l] [2]. Dengan menggunakan teknik *open loop* didapatkan hasil yang kurang optimal, dikarenakan sistem menjadi sangat sensitif terhadap parameter *plant*. Perbedaan nilai parameter sistem pada perancangan menyebabkan kontroler menjadi kurang responsif dan kurang sesuai. Kontroler *feedback* memberikan keunggulan dalam hal lebih kebal terhadap gangguan dan perubahan parameter sistem [3] [4]. Sistem kontrol PD dengan *feedback* juga diusulkan sebagai pengatur posisi dan anti ayun pada sistem *crane*. Akan tetapi diketahui bahwa kontroler PD untuk posisi akan membuat *steady state error* yang besar. Untuk sistem nonlinear *single-input multi-output* seperti halnya model *crane* ini, kontroler dengan metode *fuzzy logic* telah digunakan [5] [6]. Dengan metode *fuzzy logic* diharapkan kontroler dapat bekerja untuk hasil yang diinginkan. Hasil yang didapat menunjukkan peningkatan kinerja pengendali untuk sistem. Selain itu juga dikembangkan suatu metode optimalisasi [7] dalam hal men-*tuning* nilai *gain* pada *fuzzy*-PD kontroler berbasis *Genetic Algorithms* [8] termasuk juga optimalisasi *input tracking* dengan metode LQR-Hybrid [9]. Beberapa juga mengaplikasikan kontroler nonlinear semisal *Sliding Mode Control* yang ditambahkan kompensator PI untuk memperoleh hasil yang lebih baik [10].

1.2 Rumusan Masalah

Crane merupakan contoh *plant* dengan *single-input multi-output*. Telah banyak kontroler yang dikembangkan untuk meningkatkan efisiensi dan pengaturan anti ayun untuk diterapkan pada alat ini salah satunya dengan kontroler *Proportional-Derivative* (PD). Namun kontroler ini memiliki kekurangan pada *steady state error* yang terjadi pada posisi akhir. Oleh karenanya dikembangkan kontroler PD-LQR yang lebih baik dalam pengaturan di sekitar *steady state*. Pada kontroler PD-LQR, performansi *output*-nya sangat dipengaruhi oleh pemilihan matriks pembobot *Q* dan *R*. Ini menjadi masalah tersendiri bagi perancang dalam menentukan besaran matriks *Q* dan *R* yang sesuai sehingga diperoleh *output* sistem yang memenuhi kriteria awal perancangan. Masalah ini menjadi semakin kompleks jika jumlah *input state* dan sinyal kontrol *u* yang ada semakin banyak, ditambah dengan syarat kendala yang mesti dipenuhi dalam perancangan seperti bagaimana mengoptimalkan pemakaian energi kontrol tanpa mengurangi performansi dari sistem itu sendiri. Umumnya perancang memakai metode *trial and error* (TEM) untuk menemukan nilai matriks bobot *Q* dan *R* yang sesuai, namun dirasakan metode ini belum terlalu akurat dan cukup merepotkan.

1.3 Tujuan

Tujuan khusus yang ingin dicapai antara lain:

1. Mendesain kontroler PD-LQR yang mampu mengkompromikan masalah fungsi biaya, waktu *settling*, persentase *overshoot* dan meminimalkan *error* ayunan pada beban dan *steady state error* pada posisi akhir untuk diterapkan pada pengaturan *crane* anti ayun
2. Mendapatkan nilai matriks bobot *Q* dan *R* yang tepat dengan menerapkan algoritma uPSO sehingga diperoleh *gain K* optimal untuk kontroler PD-LQR yang didesain.
3. Mengetahui pemilihan jenis matriks bobot *Q* yang akan menghasilkan kontroler PD-LQR dengan optimasi uPSO paling optimal
4. Membandingkan metode kontrol yang didesain dengan metode kontrol lain yang sudah pernah diteliti sebelumnya oleh Ismail Rokhim [10].

1.4 Batasan Masalah

Oleh karena beragamnya permasalahan yang muncul, maka diperlukan adanya pembatasan. Dalam penelitian ini akan dibatasi beberapa pokok permasalahan yang terdiri dari:

1. Pengaruh gaya gesek selama *trolley* bergerak dan *rope* naik turun diabaikan
2. Pengaruh gaya eksternal seperti dorongan angin terhadap beban diabaikan
3. Pendulum *crane* hanya bergerak horizontal sejalur dengan pergerakan *trolley*
4. Energi yang dioptimalkan dibatasi pada sinyal kontrol *crane*. Tidak termasuk energi kontrol untuk pengaturan perubahan panjang tali beban dan lainnya.

1.5 Kontribusi

Kontribusi yang ingin dicapai dalam tesis ini antara lain:

1. Mengembangkan kontroler yang dapat mengoptimalkan dan mengkompromikan fungsi biaya, *settling time*, persentase *overshoot* dan juga mampu meminimalkan *error* ayunan beban dan *error* posisi *crane*.
2. Mengembangkan algoritma uPSO sebagai salah satu metode pemilihan matriks pembobot *Q* dan *R* pada kontroler PD-LQR sehingga diperoleh kontroler PD-LQR yang lebih optimal.

1.6 Metodologi Penelitian

Metodologi yang digunakan pada penelitian ini antara lain:

1. Studi Literatur

Studi literatur dilakukan dengan cara mengumpulkan dan mempelajari penelitian-penelitian yang relevan terhadap topik yang akan diteliti. Melalui pembelajaran dan perbandingan beberapa kajian penelitian, akan diperoleh topik yang fokus untuk diteliti. Setelah topik penelitian diperoleh, studi literatur dilanjutkan dengan mempelajari topik-topik yang sesuai. diantaranya dinamika sistem *crane* anti ayun, kontrol PD-LQR, algoritma optimasi cerdas uPSO.

2. Pemodelan Sistem

Pada tahap ini akan dilakukan pemodelan sistem *crane* anti ayun yang melibatkan dinamika pergerakan horizontal, pemodelan dan linearisasi *plant* untuk mendapatkan model referensi sesuai dengan spesifikasi desain yang diinginkan .

3. Perancangan Sistem

Berdasarkan hasil pemodelan *plant* akan dirancang suatu kontroler *crane* anti ayun yang terdiri dari kontroler PD-LQR baik *type* 0 maupun *type* I dengan menggunakan metode uPSO untuk menemukan nilai matriks *Q* dan *R* yang optimum.

4. Pengujian dan Analisa Sistem

Pada tahap ini, hasil perancangan kontroler akan disimulasikan pada *plant* pada kondisi ideal dan tanpa pengaruh gangguan. Hasil simulasi tersebut kemudian dianalisis untuk diperoleh performansi sistem secara keseluruhan. Selain itu, pada tahap ini juga akan dibandingkan unjuk kerja kontroler yang didesain dengan unjuk kerja kontroler yang sudah pernah diteliti sebelumnya.

5. Kesimpulan

Pada tahap ini kesimpulan diperoleh sesuai dengan hasil pengujian dan analisis yang dilakukan

6. Penulisan Laporan Tesis

Penulisan laporan tesis dilakukan sebagai dokumentasi dan hasil penelitian yang dilakukan

-----Halaman ini sengaja dikosongkan-----

BAB II
KAJIAN PUSTAKA DAN DASAR TEORI

2.1. Kajian Pustaka

Tujuan utama dari pengaturan *crane* adalah memindahkan beban secepat mungkin tanpa menimbulkan adanya ayunan yang tidak terkendali pada posisi akhir. Suatu *crane* harus dapat bergerak cepat untuk memindahkan beban tanpa mengakibatkan goyangan yang terlalu besar pada beban yang dipindahkan.

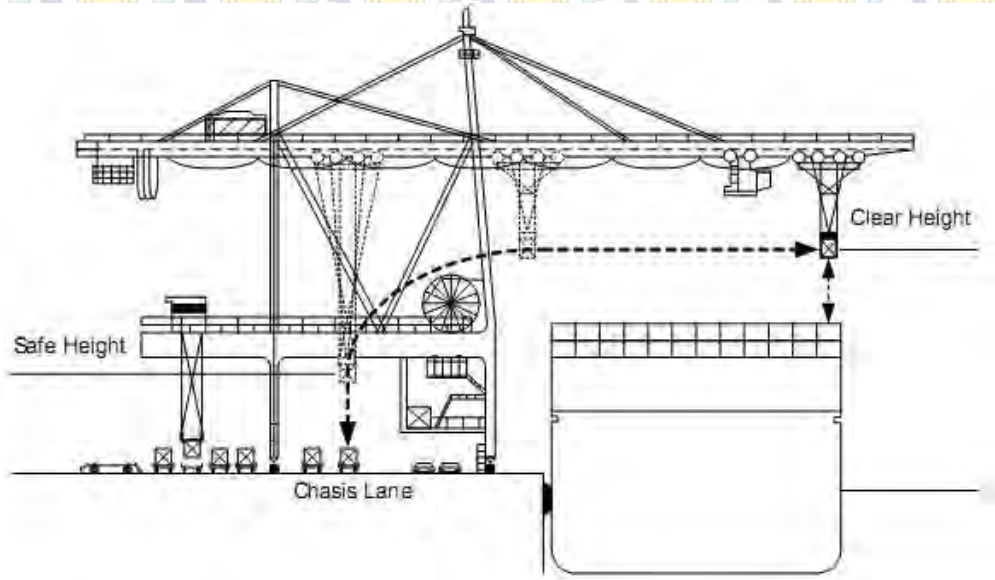
Adapun permasalahan yang akan dipecahkan dalam pengaturan *crane* antara lain adalah:

1. Sudut ayun beban yang besar pada saat bergerak, serta ayunan yang terlalu lama pada posisi akhir akan membahayakan kegiatan operasi. Kontrol secara manual pada *crane* menyebabkan beban yang diangkut cenderung berayun (berosilasi) dengan sudut ayun yang besar. Hal ini terjadi karena kemampuan/keahlian manusia yang terbatas dalam pengoperasian *crane* dalam memberikan respon, sehingga dibutuhkan kontrol otomatis yang dapat menjaga sudut ayun beban tetap minimal.
2. Kecepatan *crane* pada saat operasi seharusnya bernilai maksimal, sehingga waktu operasi dapat dihemat, yang berujung pada peningkatan efektivitas kerja. Akibat sudut ayun beban yang semakin besar pada kecepatan tinggi, maka besar kecepatan gerak *crane* menjadi terbatas. Kontroler yang bekerja pada *crane* seharusnya dapat meminimalkan waktu transfer untuk mencapai posisi akhir dan juga menjaga agar sudut ayun beban bernilai minimal.

Pengaturan secara manual *crane* sangat sulit dan membutuhkan keterampilan khusus, untuk itu perlu diterapkan suatu pengaturan *crane* secara otomatis dengan menggunakan teori-teori kontrol yang telah ada baik yang klasik maupun modern.

Pengaturan *crane* secara otomatis yang akan mengurangi ayunan beban telah banyak dipublikasikan oleh para peneliti dengan berbagai metode. misalnya, penelitian tentang penerapan *input shaping* sinyal kontrol pada sistem kontrol *gantry crane* metode *open loop* untuk menghilangkan efek impuls yang dapat

menyebabkan osilasi (ayunan) pada *crane* ketika sudah sampai pada posisi akhirnya [1].



Gambar 2.1. *Gantry Crane* di Pelabuhan Kontainer

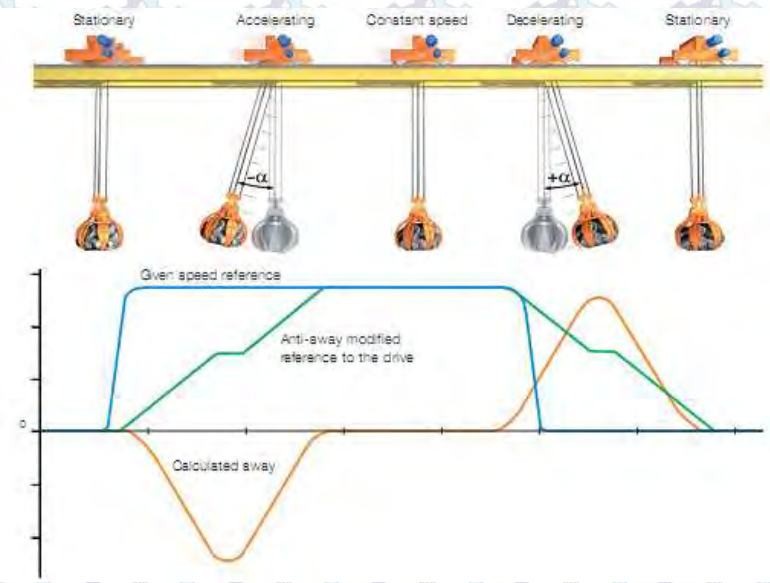
Mereka juga mengkombinasikan metode pemberian sinyal kontrol *input shaping* dengan kontroler *Linear Quadratic Regulator* untuk menghasilkan kontroler yang optimal [1].

Beberapa peneliti juga mendesain dan memodifikasi metode sinyal kontrol *command/input shaping* untuk mengurangi getaran sisa pada titik akhir *crane* dan membatasi sudut ayunan beban sewaktu beban berjalan [2]. Mereka juga mencoba metode ini untuk mengendalikan sudut ayunan beban pada model pendulum ganda.

Metode yang hampir sama juga dilakukan untuk mengaplikasikan sinyal kontrol *On* dan *Off* untuk mengurangi osilasi (ayunan) pada beban *crane* jenis *bridge* (jembatan) yang menggunakan DC motor sebagai penggeraknya [3]. Metode ini mampu mereduksi osilasi sewaktu *crane* bergerak maupun saat berhenti.

Selain menerapkan metode kontrol *open-loop*, beberapa peneliti juga memakai metode *close-loop*. Misalnya penelitian yang membandingkan antara metode kontrol *delayed feedback* dengan metode kontrol PD-*Fuzzy logic controler*

[5]. Berdasarkan hasil penelitian ini terlihat bahwa kedua metode kontroler tersebut memiliki kehandalan yang hampir sama dan tidak jauh berbeda.

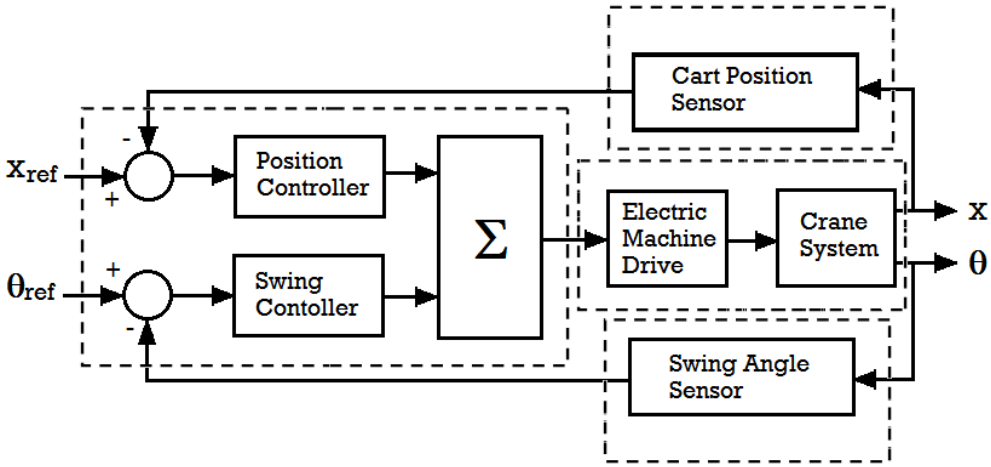


Gambar 2.2. Ilustrasi Model Pergerakan Crane

Begitu pula dengan penelitian tentang metode kontrol *fuzzy* dengan kemampuan kontroler yang kokoh dalam menghadapi perubahan-perubahan variabel beban dan panjang tali. Pada penelitian ini kontroler *fuzzy robust* dibentuk dengan menggabungkan kontroler *Proportional-Derivative* (PD) dan *Pole Placement Method* (PPM) yang kemudian di implementasikan pada Takagi-Sugeno-Kang *inference system* [6]. Kontroler ini merupakan kontroler berbasis sistem waktu diskrit dimana nilai *gain K* untuk masukan posisi *crane*, kecepatan dan ayunan beban diturunkan dengan memakai PPM untuk titik-titik operasi yang dipilih dengan menggunakan persamaan tertentu.

Selain itu juga diteliti dan didesain suatu kontroler anti ayun yang menerapkan penjadwalan *gain* yang dioptimalkan dengan memakai metode algoritma cerdas berupa *Genetic Alghoritma* [8].

Adapun diagram blok yang merupakan konsep dasar suatu kontroler *crane* anti ayun dapat dilihat pada Gambar 2.3.



Gambar 2.3. Diagram Blok Kontroler Dasar *Crane* Anti Ayun

2.2 Landasan Teori

2.2.1 Sistem *Crane* dengan Kereta

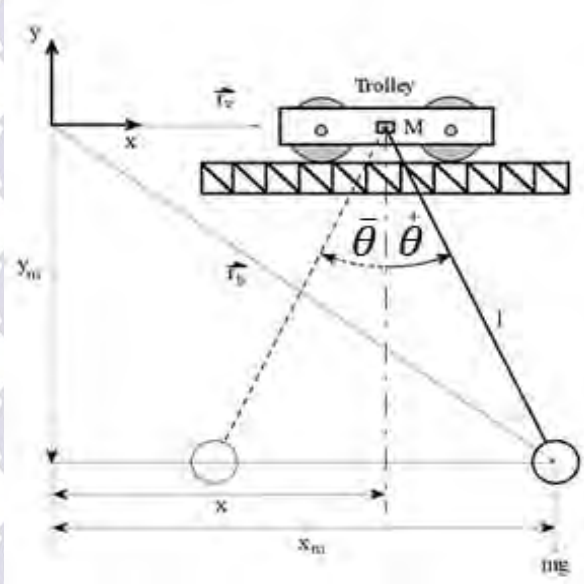
Sistem *Crane* dengan sebuah kereta (*trolley*) sebagai objek pada penelitian ini merupakan sistem *crane* yang paling sederhana dengan hanya satu tali (*rope*) yang terpasang pada *trolley* sehingga beban pada tali tersebut dapat berayun bebas pada bidang vertikal.

Pada penelitian ini, *trolley* digerakkan oleh motor DC yang dihubungkan dengan *belt* yang memungkinkan *trolley* digerakkan ke kiri atau ke kanan pada lintasan rel yang panjangnya terbatas untuk memindahkan beban yang dimaksud. Selama proses pemindahan dan kembalinya *trolley* ke posisi awal, terjadi perubahan-perubahan parameter sistem seperti panjang tali dan massa beban.

Posisi *trolley* pada lintasan dapat dipantau melalui sensor posisi yaitu *position encoder* dan sebagai tambahan pengaman, digunakan limit switch pada masing-masing ujung lintasan rel. Ketika *trolley* berada di ujung lintasan, maka limit switch tertekan oleh *trolley* dan motor DC akan mati, sehingga *trolley* akan berhenti. Sedangkan posisi sudut pendulum terhadap sumbu vertikal dipantau oleh sebuah sensor sudut.

2.2.1.1 Model Dinamik *Crane* [1]

Diagram fisik *crane* yang digunakan serta gaya-gaya yang terjadi pada sistem ditunjukkan pada Gambar 2.4. Secara fisik, *crane* terdiri dari dua bagian utama, yaitu kereta (*trolley*) dan beban pendulum. *Trolley* hanya dapat bergerak pada rel dalam bidang horizontal dan pendulum berotasi pada bidang vertikal yang bersumbu pada sisi *trolley*.



Gambar 2.4. Model *Crane* Sederhana

Dengan :

x = posisi horizontal *trolley* [m]

$\dot{x}$ = kecepatan *trolley* [m/s]

θ = *deviasi* sudut beban [rad]

$\dot{\theta}$ = kecepatan sudut beban [rad/s]

l = panjang tali *hoisting* [m]

M = massa *trolley* [kg]

m = massa beban [kg]

g = percepatan gravitasi [m/s^2]

Berdasarkan gerak *crane* pada bidang dua dimensi, energi kinetik dan energi potensial pada sistem *crane* dirumuskan dengan persamaan [1]:

$$W_k = W_v + W_b$$
$$= \frac{1}{2} M \vec{r}_v . \vec{r}_v + \frac{1}{2} m \vec{r}_b . \vec{r}_b$$
$$= \frac{1}{2} M \dot{x}^2 + \frac{1}{2} m (\dot{x}^2 + l^2 \dot{\theta}^2 + 2 l \dot{x} \dot{\theta} \cos\theta) \quad (2.1)$$

$$W_p = m g y_m$$
$$= -mg \cos\theta \quad (2.2)$$

Untuk memenuhi model dinamik dari *crane*, persamaan energi pada (2.1) dan (2.2) digunakan pada persamaan Lagrangian [3] $L = W_k - W_p$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{r}_j}\right) - \frac{\partial L}{\partial r_j} = F_j \qquad j = 1,2 \quad (2.3)$$

Sehingga diperoleh persamaan dinamika gerak nonlinear sistem orde 2 sebagai berikut [3]:

$$F_x = (M + m)\ddot{x} + ml(\ddot{\theta}\cos\theta - \dot{\theta}^2 \sin\theta) + 2m\dot{l}\dot{\theta}\cos\theta + m\ddot{l}\sin\theta \quad (2.4)$$

$$l\ddot{\theta} + 2\dot{l}\dot{\theta} + \ddot{x}\cos\theta + g\sin\theta = 0 \quad (2.5)$$

Selanjutnya turunan dari Persamaan (2.4) dan (2.5) didapatkan dengan menggunakan fungsi Lagrangian (2.3), diperoleh persamaan [1]:

$$\ddot{x} = \frac{F_x + mg\cos\theta\sin\theta + ml\dot{\theta}^2 \sin\theta}{(M + m - m\cos^2\theta)} \quad (2.6)$$

$$\ddot{\theta} = \frac{-F_x \cos\theta - ml\dot{\theta}^2 \cos\theta\sin\theta - Mg\sin\theta - mg\sin\theta}{l(M + m\sin^2\theta)} \quad (2.7)$$

2.2.1.2 Model Linearisasi *Crane* [1]

Persamaan (2.6) dan (2.7) menggambarkan model nonlinear *plant*. Model nonlinear ini dapat dilinearisasikan dengan beberapa asumsi. Untuk penelitian ini

diasumsikan bahwa ayunan beban *crane* (θ) selama *trolley* bergerak memindahkan beban dianggap kecil [1].

Berdasarkan asumsi tersebut diperoleh ekspansi fungsi sinus dan cosinus dengan deret Taylor pada sekitar titik *equilibrium* sudut 0 derajat, adalah sebagai berikut:

$$\sin\theta = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \frac{\theta^7}{7!} + \ldots \approx \theta$$
$$\cos\theta = 1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \frac{\theta^6}{6!} + \ldots \approx 1 \quad (2.8)$$

Sehingga diperoleh persamaan differential yang baru sebagai berikut:

$$\ddot{x} = \left(\frac{m}{M}\right) g\theta + \frac{F_x}{M} \quad (2.9)$$

$$\ddot{\theta} = -\left[\left(\frac{M+m}{Ml}\right) g\theta + \frac{F_x}{Ml}\right] \quad (2.10)$$

Untuk vektor *input* berupa $x = [x \quad \theta \quad \dot{x} \quad \dot{\theta}]^T$, diperoleh persamaan *state* berupa:

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

Dengan matriks A, B dan matriks C, D adalah:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{mg}{M} & 0 & 0 \\ 0 & -\frac{(M+m)g}{Ml} & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{M} \\ -\frac{1}{Ml} \end{bmatrix} \quad (2.11)$$

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

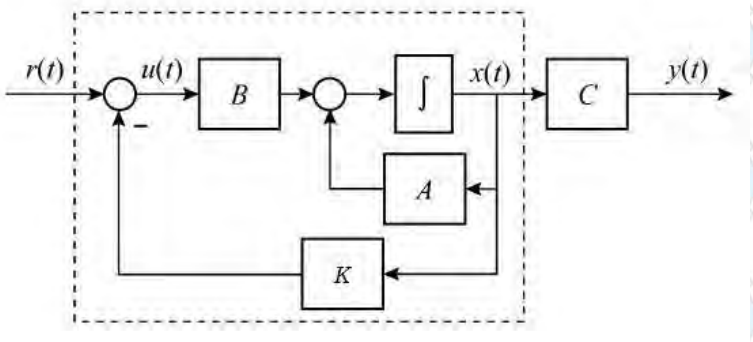
2.2.2 Kontroler *Linear Quadratic Regulator* (LQR) [11]

Kontrol optimal merupakan perluasan dari kalkulus variasi, yaitu metode optimasi matematika untuk menurunkan kebijakan pengendalian. Metode ini sebagian besar dikembangkan oleh Lev Pontryagin dan rekan-rekannya di Uni Soviet, serta Richard Bellman di Amerika Serikat.

Kontrol optimal berkaitan dengan masalah menemukan hukum kontrol untuk sistem tertentu dengan pencapaian kriteria optimalitas tertentu. Istilah optimal mempunyai maksud hasil paling baik yang dapat dicapai dengan memperhatikan kondisi dan kendala dari suatu sistem.

Dalam sistem kontrol istilah optimal seringkali merujuk pada nilai minimal, misalnya meminimalkan energi (*input*), waktu dan kesalahan (*error*). Sistem kontrol yang baik adalah sistem kontrol yang mempunyai respon tanggap yang cepat dan stabil, tetapi tidak memerlukan energi yang berlebihan. Sistem kontrol demikian dapat dicapai melalui pengaturan indeks performansi (*performance index*) yang tepat. Sistem kontrol yang dirancang berdasarkan optimasi indeks performansi disebut sistem kontrol optimal. Pada suatu sistem, indeks performansi dipilih sesuai dengan bagian yang akan dioptimalkan.

Salah satu jenis kontrol optimal adalah kontroler *Linear Quadratic Regulator* (LQR). Pada dasarnya regulator pada suatu sistem pengaturan bertujuan untuk pengendalian perubahan *state* sistem pada sekitar *steady state*



Gambar 2.5. Diagram Blok Kontroler LQR

Pada sistem kontrol optimal berdasarkan indeks performansi kuadratis, optimasi kontrol dicapai dengan meyelesaikan terlebih dahulu persamaan Riccati untuk suatu matriks bobot *S(t)* menggunakan nilai kondisi akhir dari $S(t_a)$ yang

dipilih untuk mengoptimalkan indeks performansi tersebut sehingga diperoleh matriks *gain* umpan balik *state* optimal *K*.

Untuk sistem *crane* anti ayun berlaku [12]:

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) \tag{2.12}$$

dengan $x(t) \in R^n$, $u(t) \in R^p$, $A \in R^{nxn}$, $B \in R^{nxp}$

Untuk kasus *free final state*, dengan keadaan akhir tidak diketahui, diinginkan untuk memilih suatu sistem kontrol $u(t)$ yang memenuhi Persamaan (2.12) serta meminimumkan indeks performansi berbentuk:

$$J(t) = \frac{1}{2}[x^T(t_a)Qx(t_a)] + \frac{1}{2}\int_{t_0}^{t_a}[x^T(t)Qx(t) + u^T(t)Ru(t)]dt \tag{2.13}$$

Dengan matriks pembobot kontrol *R* merupakan matriks simetri definit positif (R>0), dan matriks pembobot keadaan *Q* adalah matriks simetri semidefinit positif (Q≥0).

$$\text{dengan matriks } Q = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \\ q_2 & q_5 & q_6 & q_7 \\ q_3 & q_6 & q_8 & q_9 \\ q_4 & q_7 & q_9 & q_{10} \end{bmatrix} \text{ dan } R = [r_1] \tag{2.14}$$

Untuk vektor *input* berupa $x = [x \quad \theta \quad \dot{x} \quad \dot{\theta}]^T$ dan sinyal kontrol $u(t)$, Persamaan (2.13) dapat ditulis ulang menjadi:

$$J(t) = \frac{1}{2}[x^T(t_a)Qx(t_a)] + \frac{1}{2}\int_{t_0}^{t_a}[x^T(t)Qx(t) + u^2(t)R]dt \tag{2.15}$$

Matriks *Q* dan *R* ini akan di-*tuning* untuk mendapatkan nilai yang paling optimal dengan menggunakan metode uPSO sebagaimana yang akan dibahas selanjutnya.

Untuk menyelesaikan masalah LQR, dibentuk persamaan Hamiltonian yang diberikan sebagai berikut [12]:

$$H(x,u,\lambda,t) = L(x,u,t) + \lambda^T f(x,u,t) \tag{2.16}$$

untuk

$$f(x,u,t) = Ax(t) + Bu(t), \; x_0 = x(t_0) \tag{2.17}$$

$$L(x,u,t) = \tfrac{1}{2}(x^T Qx + u^T Ru) \tag{2.18}$$

Sehingga diperoleh persamaan Hamiltonian berupa:

$$H = \frac{1}{2}x^T(t)Qx(t) + \frac{1}{2}u^T(t)Ru(t) + \lambda^T(t)[Ax(t) + Bu(t)]dt \tag{2.19}$$

Dari Persamaan (2.19) tersebut, dengan syarat kondisi stationer:

$$\frac{\partial H}{\partial u} = Ru(t) + B^T\lambda(t) = 0$$

$$Ru(t) = -B^T\lambda(t)$$

$$u(t) = -R^T B^T\lambda(t) \tag{2.20}$$

Dari Persamaan (2.19) tersebut, untuk syarat perlu dan cukup berlaku:

1. Persamaan *state*

$$\frac{\partial H}{\partial x} = \dot{x}$$

$$\frac{\partial H}{\partial x} = \dot{x} = Ax + Bu \tag{2.21}$$

2. Persamaan *co-state*

$$\frac{\partial H}{\partial \lambda} = -\dot{\lambda}$$

$$\frac{\partial H}{\partial \lambda} = -\dot{\lambda} = \tfrac{1}{2}Qx + \tfrac{1}{2}Q^T x + A^T\lambda \;\; (Q = Q^T = simetris)$$

$$\frac{\partial H}{\partial \lambda} = -\dot{\lambda} = Qx + A^T\lambda \tag{2.22}$$

Dari Persamaan (2.20), (2.21) dan (2.22), dapat dibentuk matriks Hamiltonian sebagai berikut:

$$\begin{pmatrix} \dot{x}(t) \\ \dot{\lambda}(t) \end{pmatrix} = \begin{pmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{pmatrix} \begin{pmatrix} x(t) \\ \lambda(t) \end{pmatrix} \tag{2.23}$$

Untuk kasus *free final state*, berlaku persamaan:

$$Q(x,u,t)\big|_{t=ta} = \tfrac{1}{2}x^T Sx$$

$$\psi(x,u,t)\big|_{t=ta} = 0 \tag{2.24}$$

Sehingga diperoleh persamaan:

$$\lambda(t_a) = S(t_a)x(t_a) \tag{2.25}$$

Untuk kasus *free final state* maka t_a diganti dengan t , sehingga Persamaan (2.25) menjadi:

$$\lambda(t) = S(t)x(t) \tag{2.26}$$

Jika Persamaan (2.26) diturunkan, akan menjadi:

$$\dot{\lambda} = \dot{S}x + S\dot{x} \tag{2.27}$$

Jika Persamaan (2.27) disubsitusi dengan Persamaan (2.21) dan (2.22), diperoleh:

$$\dot{S}x + S(Ax - BR^{-1}B^T\lambda) = -Qx - A^T\lambda$$

$$\dot{S}x + SAx - SBR^{-1}B^T\lambda = -Qx - A^T\lambda \tag{2.28}$$

Selanjutnya Persamaan (2.28) disubsitusi dengan Persamaan (2.22), didapatkan:

$$\dot{S}x + SAx - SBR^{-1}B^T Sx = -Qx - A^T Sx$$

$$SAx - SBR^{-1}B^T Sx + Qx + A^T Sx = -\dot{S}x$$

$$(SA - SBR^{-1}B^T S + Q + A^T S)x = -\dot{S}x$$

$$SA - SBR^{-1}B^T S + Q + A^T S = -\dot{S} \tag{2.29}$$

Persamaa (2.29) ini disebut dengan persamaan diferensial Riccati yang merupakan persamaan linier dalam *S(t).* Solusi persamaan Riccati, *S(t)* bisa diperoleh dengan menyelesaikan persamaan:

$$-\dot{S}(t) = A^T S(t) + S(t)A + Q - S(t)BR^{-1}B^T S(t) \tag{2.30}$$

Jika Persamaan (2.31) memiliki penyelesaian *S(t)*, maka solusi kontrol optimal *u(t)* diberikan oleh:

$$u^*(t) = -R^{-1}B^T\lambda(t)$$

$$u^*(t) = -R^{-1}B^T S(t)x(t) \tag{2.31}$$

dapat didefinisikan matriks *gain K* sebagai:

$$K = R^{-1}B^T S(t) \tag{2.32}$$

Sehingga solusi kontrol optimal *u(t)* dapat dituliskan ulang sebagai:

$$u^*(t) = -K^* x(t) \tag{2.33}$$

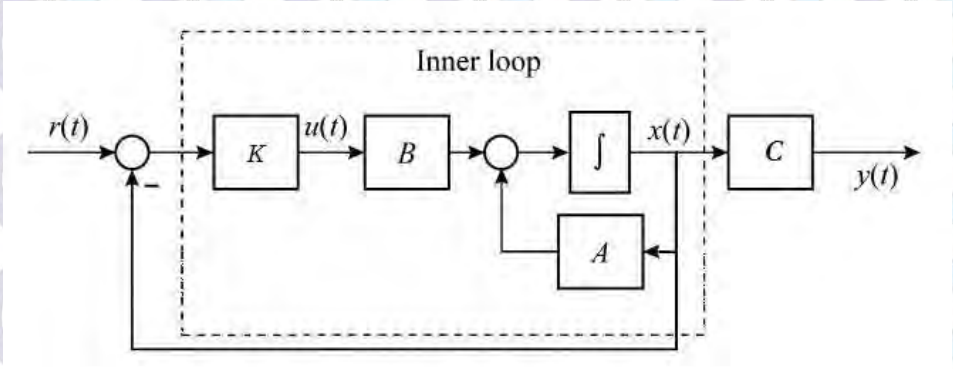
2.2.2.1 LQR untuk *Command Tracking* [13]

Pada kasus-kasus tertentu, banyak dijumpai suatu sistem pengaturan regulator yang memiliki *input* referensi berupa *unit step*. Sistem yang demikian biasa disebut dengan sistem regulasi dengan *command tracking* atau *unit step tracking.*

Untuk kasus LQR dengan *command tracking* ada beberapa metode yang dipakai, antara lain LQR *type* 0 dan LQR *type* I. Untuk LQR *type* 0 merupakan LQR dasar standar, sedangkan LQR *type* I merupakan LQR dasar dengan tambahan integrator dan *gain* integrasi (K_i) pada sisi *error* selisih antara *input* referensi dan *output* aktual.

2.2.2.2 LQR *Type* 0 [13]

Sebuah LQR standar dapat digunakan untuk pengaturan *input step tracking/command tracking*. Kelemahan dari LQR jenis ini yaitu bahwa *steady state error* tidak persis mencapai nol tapi masih dalam nilai yang dapat diterima. Keuntungan jenis ini yaitu rangkaiannya yang sederhana tanpa adanya kompensator, juga tidak adanya saturasi terkait penambahan kompensator *integrator*.



Gambar 2.6. LQR *Command Tracking Type* 0

Walaupun demikian, permasalahan *steady state error* pada kontroler jenis ini dapat dibuat sama dengan nol dengan melakukan penyesuaian yang tepat pada beberapa nilai *gain* kontrol *K*.

Untuk kontroler LQR *command tracking* berlaku:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t)\\ y(t) &= Cx(t) + Fr(t)\\ z(t) &= Hx(t)\end{aligned} \tag{2.34}$$

dengan $x(t), y(t), z(t) \in R^n$, $u(t) \in R^p$, $A \in R^{nxn}$, $B \in R^{nxp}$,

$x(t)$ adalah variabel *state* sistem sedangkan $y(t)$ adalah variabel *state feed-foward* sistem dan $z(t)$ adalah *output plant* yang terukur.

Untuk solusi kontrol optimal $u(t)$ dapat ditulis sebagai:

$$u^*(t) = -K^* y(t) \tag{2.35}$$

dengan *gain* umpan balik *state* K didefinisikan sebagai:

$$K = R^{-1}B^T S(t) \tag{2.36}$$

Sehingga diperoleh persamaan *state close loop system* untuk Gambar 2.6 sebagai berikut:

$$\begin{aligned}\dot{x}(t) &= (A - BKC)x(t) + (-BKF)u(t)\\ z(t) &= Hx(t)\end{aligned} \tag{2.37}$$

2.2.2.3 LQR *Type* I (LQR-*Integrator*) [14]

Untuk kasus sistem LTI berlaku:

$$\begin{aligned}\dot{x}(t) &= Ax(t) + Bu(t)\\ y(t) &= Cx(t)\end{aligned} \tag{2.38}$$

dengan $x(t) \in R^n$, $u(t) \in R^p$, $A \in R^{nxn}$, $B \in R^{nxp}$

Pada kontroler LQR *type* I (LQR-I), didefinisikan integral *error* sebagai:

$$\begin{aligned}\varepsilon(t) &= \int_0^t (r(\tau) - y(\tau))d\tau\\ \dot{\varepsilon}(t) &= r(t) - y(t)\\ \dot{\varepsilon}(t) &= r(t) - Cx(t)\end{aligned} \tag{2.39}$$

untuk $r(t) \in R^m$ merupakan *input* referensi.

Dari kombinasi Persamaan (2.38) dan (2.39) diperoleh persamaan *augmented state* sebagai berikut:

$$\begin{bmatrix}\dot{x}(t)\\ \dot{\varepsilon}(t)\end{bmatrix}=\begin{bmatrix}A & 0\\ -C & 0\end{bmatrix}\begin{bmatrix}x(t)\\ \varepsilon(t)\end{bmatrix}+\begin{bmatrix}B\\ 0\end{bmatrix}u(t)+\begin{bmatrix}0\\ I\end{bmatrix}r(t) \quad (2.40)$$

dengan I merupakan matriks identitas,

dan $E=\begin{bmatrix}0\\ I\end{bmatrix}$ (2.41)

Suatu kontroler LQR-I *command tracking* seperti pada Gambar 2.7, berlaku persamaan:

$$\dot{x}(t)=Ax(t)+Bu(t)+Er(t)$$
$$y(t)=Cx(t) \quad (2.42)$$

yang merupakan sebuah kontroler umpan balik *state* optimal yang memenuhi persamaan:

$$u(t)=K.x(t)+Ki.\varepsilon(t) \quad (2.43)$$

yang akan menstabilkan *plant augmented* sesuai Persamaan (2.40) dan meminimumkan fungsi biaya kuadratik

$$J(t)=\int_{t_0}^{\infty}[[\tilde{x}(t)^T\ \tilde{\varepsilon}(t)^T]^T.Q.[\tilde{x}(t)^T\ \tilde{\varepsilon}(t)^T]+u(t)^T R.u(t)]dt \quad (2.44)$$

Untuk *input* referensi berupa $r(t) \equiv r\ (t>0)$, di mana $K \in R_{n\times n}$ dan $K_i \in R_{m\times m}$ merupakan matriks *gain* umpan balik keadaan *plant* dan *gain error* integral, yang memenuhi $Q \in R_{(n+m)\times(n+m)}$ dan $R \in R_{m\times m}$ adalah matriks bobot

$$\tilde{x}(t)=x(t)-x(\infty)$$
$$\tilde{u}(t)=u(t)-u(\infty)$$
$$\tilde{\varepsilon}(t)=\varepsilon(t)-\varepsilon(\infty) \quad (2.45)$$

untuk $x(\infty)$, $u(\infty)$, dan $\varepsilon(\infty)$ masing-masing adalah nilai *steady state* dari *state* sistem, sinyal kontrol dan integral *error*, di mana:

$$\dot{\tilde{x}}(\infty)=Ax(\infty)+Bu(\infty)$$
$$\dot{\tilde{\varepsilon}}(\infty)=r-Cx(\infty)=0 \quad (2.46)$$

maka Persamaan (2.40) dapat dituliskan sebagai

$$\begin{bmatrix}\dot{\tilde{x}}(t)\\ \dot{\tilde{\varepsilon}}(t)\end{bmatrix}=\begin{bmatrix}A & 0\\ -C & 0\end{bmatrix}\begin{bmatrix}\tilde{x}(t)\\ \tilde{\varepsilon}(t)\end{bmatrix}+\begin{bmatrix}B\\ 0\end{bmatrix}\tilde{u}(t)$$

Jika dimisalkan:

$$X(t)=\begin{bmatrix}\tilde{x}(t)\\ \tilde{\varepsilon}(t)\end{bmatrix} \quad (2.47)$$

$$\tilde{A}=\begin{bmatrix}A & 0\\ -C & 0\end{bmatrix} \quad (2.48)$$

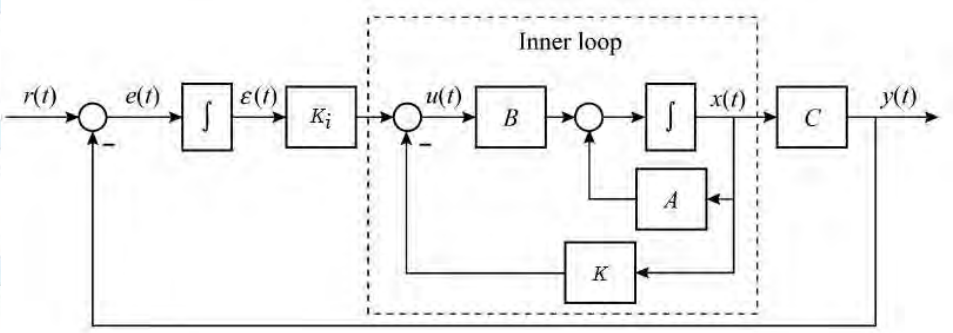
$$\tilde{B}=\begin{bmatrix}B\\ 0\end{bmatrix}$$

Maka Persamaan (2.47) dapat dituliskan sebagai:

$$\dot{X}(t)=\tilde{A}X(t)+\tilde{B}u(t) \quad (2.49)$$

Sehingga K dan K_i dapat diperoleh seperti halnya pada kontroler LQR dasar berupa:

$$K_{CL}=[K\ K_i]=R^{-1}\tilde{B}^T S(t) \quad (2.50)$$



Gambar 2.7. LQR *Command Tracking Type* I

Sehingga diperoleh persamaan *state* untuk kondisi *close loop system* Gambar 2.7 sebagai berikut:

$$\dot{X}(t)=(A-BK_{CL}C)X(t)+(E-BK_{CL})u(t)$$
$$z(t)=HX(t) \quad (2.51)$$

2.2.2.4 Pemilihan Matriks Pembobot *Q* dan *R* [15]

Nilai dari matriks pembobot *Q* dan *R* akan menentukan hasil dari persamaan Riccati, hal ini akan berpengaruh pada keseluruhan performansi sistem.

Matriks *Q* yang berupa matriks semidefinit positif ($Q\geq 0$) merupakan koefesien pembobot yang akan mempengaruhi sistem untuk mencapai keadaan *steady state*, semakin besar matriks *Q* maka semakin mempercepat sistem mencapai kondisi *steady state*. Sedangkan matriks *R* yang berupa matriks definit

positif (R>0) merupakan koefisien pembobot yang akan mempengaruhi energi sinyal kontrol u, dimana semakin besar harga matriks R akan semakin meminimumkan sinyal kontrol u.

Tetapi perlu diperhatikan bahwa apabila sinyal kontrol terlalu besar dapat menyebabkan saturasi pada peralatan aktuator sehingga perlu adanya batasan tertentu. Beberapa cara yang dapat digunakan dalam pemilihan matriks pembobot Q dan R adalah :

a. Cara sederhana dengan memilih matriks Q = I dan $R = \rho$I yang memenuhi persamaan L = $\|x\|^2 + \rho\|u\|^2$, untuk nilai ρ dipilih pada nilai tertentu sehingga diperoleh respon yang baik.
b. Dengan memilih matriks Q dan R berupa matriks diagonal, dengan Q = diag[q] dan $R=\rho$.diag[r]. Pilih setiap nilai q_i sehingga diperoleh respon yang bersesuaian untuk batas *error* tiap variabel yang masih ditolerir, misalnya untuk x_1 adalah jarak dengan *error* yang ditolerir sebesar 1 cm maka nilai q_1 dipilih sebesar $(1/1)^2$= 1 yang berarti bahwa ketika x_1 =1 maka $q_{1*} x_1^2$ =1. Begitu pula untuk matriks R namun dengan tambahan ρ yang dipilih untuk mengatur *input state* jadi seimbang.
c. Dengan menggunakan bobot *output z*, untuk $z = Hx$ merupakan *output* yang diharapkan tetap kecil, jika diasumsikan (A,H) dapat diamati, maka $Q = H^T.H$ dan $R = \rho$I.
d. Dengan metode *trial and error* (coba-coba) hingga diperoleh respon yang memuaskan.
e. Dengan algoritma cerdas.

Pada penelitian kali ini nilai matriks bobot Q dan R yang optimal akan dicari dengan menggunakan algoritma cerdas *unified Particle Swarm Optimization* (uPSO) untuk memperoleh nilai yang optimum.

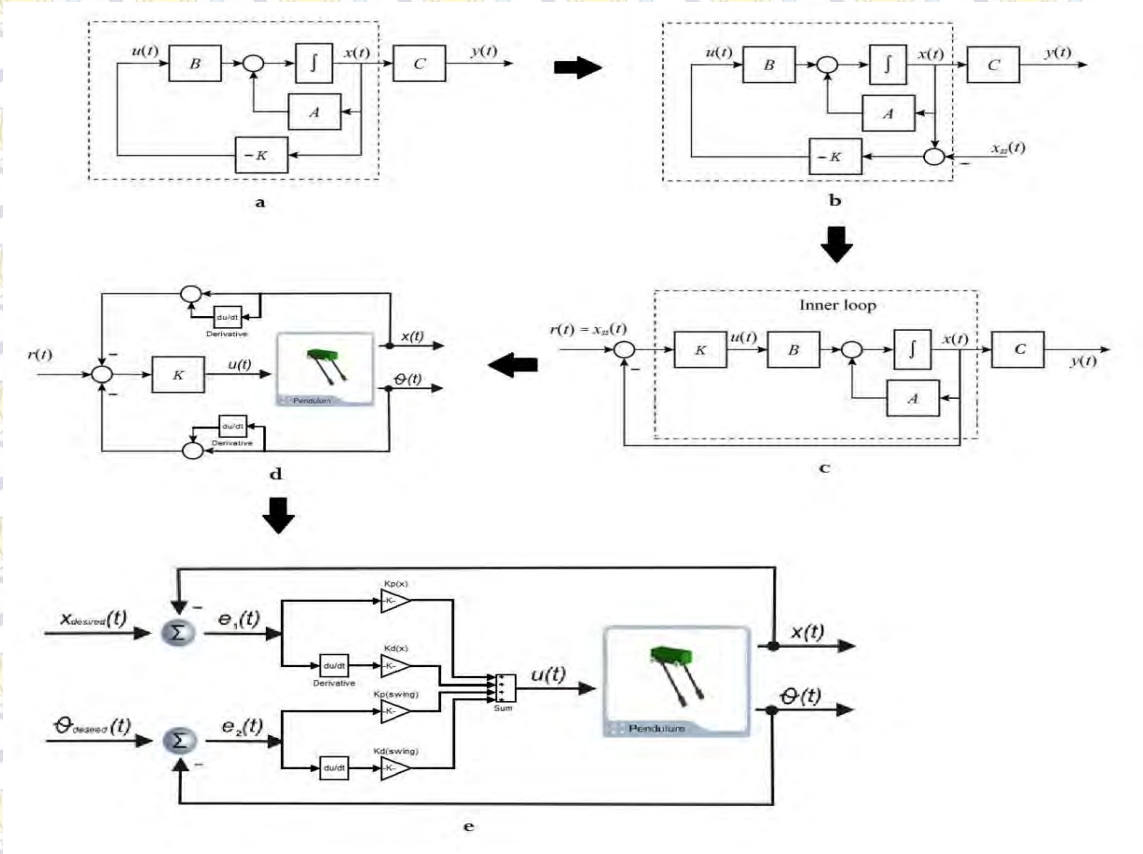
2.2.3 Konsep PD-LQR

Dalam aplikasi-aplikasi kontrol di industri umumnya kontroler yang digunakan adalah kontroler PD ataupun PID konvensional. Untuk mendapatkan *gain* pada kontroler PD/PID ini biasanya digunakan metode seperti teknik *root locus*, aturan Ziegler-Nichols, atau teknik-teknik penentuan lainnya.

Penerapan kontroler LQR untuk diaplikasikan langsung pada *plant* yang sudah lama ada (*eksisting*) di suatu industri relatif susah untuk diterapkan. Untuk itu coba diaplikasikan suatu metode *tuning* kontroler PD yang dihasilkan dari suatu perhitungan kontroler LQR, sehingga nilai-nilai *gain* suatu kontroler PD dapat diperoleh dari nilai *gain* kontroler LQR.

Adapun prinsip adaptasi *gain* suatu kontroler LQR untuk diterapkan pada kontroler PD dapat dilihat seperti pada Gambar 2.8.

Prinsip kerja suatu kontroler LQR adalah mengatur perubahan *state* yang terjadi pada daerah sekitar *steady state* (Gambar 2.8-a). Jika kondisi akhir *state plant* tidak berada pada daerah *steady state*, maka perlu adanya referensi eksternal yang memaksa kondisi akhir *state* untuk menuju *steady state* (Gambar 2.8-b).



Gambar 2.8. Adaptasi Kontroler LQR Menjadi Kontroler PD

Selisih antara *state* sistem dengan input referensi *state* eksternal disebut dengan *error state* pada sekitar daerah *steady state* (Gambar 2.8-c). *Error state* ini

identik dengan *error input* referensi (*error set point*) pada kontroler PD jika dimisalkan referensi nilai *steady state* x_{ss} pada suatu kontroler LQR identik dengan sinyal *set point* (SP) pada kontroler PD (Gambar 2.8-d dan Gambar 2.8-e).

Pada kontroler LQR *command tracking* berlaku persamaan:

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Fr(t)$$
$$z(t) = Hx(t)$$

dengan $x(t), y(t), z(t) \in R^n$, $u(t) \in R^p$, $A \in R^{nxn}$, $B \in R^{nxp}$,

Maka untuk proses adaptasi pada kontroler PD, berlaku

$$y(t) = Cx(t) + Fr(t) = e_{ss}(t) = -e_{ref}(t) \quad (2.52)$$

Untuk vektor *input* referensi $r_{ref} = [x_{ref} \quad \theta_{ref} \quad \dot{e}_{ref} \quad \dot{\theta}_{ref}]^T$ dan sinyal *state* $x = [x \quad \theta \quad \dot{x} \quad \dot{\theta}]^T$

diperoleh sinyal *error state* $e_{ss} = -e_{ref} = [-e_x \quad -e_\theta \quad -e_{\dot{x}} \quad -e_{\dot{\theta}}]^T \quad (2.53)$

dengan besarnya $e_{\dot{x}} = \frac{de_x}{dt}$ dan $e_{\dot{\theta}} = \frac{de_\theta}{dt}$

Sehingga $e_{ref} = [e_x \quad e_\theta \quad \frac{de_x}{dt} \quad \frac{de_\theta}{dt}]^T \quad (2.54)$

Untuk solusi kontrol optimal *u(t)* pada LQR dapat dirumuskan sebagai:

$$u^*(t) = -K^* y(t) = -K^* e_{ss}(t) \quad (2.55)$$

Maka pada kontroler PD untuk solusi kontrol optimal *u(t)* dapat ditulis sebagai:

$$u^*(t) = -K^* - e_{ref}(t)$$
$$u^*(t) = K^* e_{ref}(t)$$
$$u^*(t) = [K_1 K_2 K_3 K_4].[e_x \quad e_\theta \quad \frac{de_x}{dt} \quad \frac{de_\theta}{dt}]^T$$
$$u^*(t) = (K_1.e_x + K_2.e_\theta + K_3 \frac{de_x}{dt} + K_4 \frac{de_\theta}{dt})$$
$$u^*(t) = K_{p1}.e_x + K_{p2}.e_\theta + sK_{d1}e_x + sK_{d2}.e_\theta \quad (2.56)$$

2.2.4 *Particle Swarm Optimization* (PSO) [16]

2.2.4.1 Konsep Dasar PSO

Berdasarkan definisi, kata *swarm* memiliki arti segerombolan atau sekelompok yang pada umumnya mengacu pada segerombolan atau sekelompok hewan. Jadi pada dasarnya *swarm intelligence* adalah salah satu teknik kecerdasan buatan (*artificial intelligence)* yang dirancang untuk menyelesaikan masalah pada sistem terdistribusi yang didasarkan pada perilaku kolektif (*collective behaviour*) pada sebuah koloni serta kemampuan untuk mengatur dirinya sendiri (*self organizing*). Perilaku kolektif dimaksudkan pada perilaku semua individu pada sebuah koloni yang saling membantu dan melengkapi satu sama lain dalam menyelesaikan sebuah persoalan yang dihadapi koloni, sedangkan *self organizing* adalah bagaimana perilaku setiap individu menempatkan dirinya pada koloni tersebut. Koloni hewan biasanya tidak memiliki pemimpin formal yang mengatur secara pasti tugas dan kewajiban dari setiap individu. Walaupun tidak ada pemimpin secara definitif, keteraturan dapat timbul. Keteraturan di sini muncul secara spontan, misalnya secara bergantian burung-burung dalam kawanan tadi mengambil posisi di depan. Sifat seperti ini sering disebut sebagai *self organizing*.

Setiap partikel dalam *swarm intelligence* memiliki tiga sifat mendasar, yaitu:

1. Perilaku yang kolektif dalam setiap aktivitas. Setiap partikel selalu bergerak secara kolektif dalam menyelesaikan tugasnya. Perilaku kolektif ini akan mempengaruhi perilaku tiap individu di dalam *swarm*.
2. Adanya desentralisasi gerak. Setiap partikel di dalam *swarm* akan bergerak menyebar ke segala arah untuk mencari solusi, namun pergerakan tersebut masih bersifat terpusat dalam *swarm* sebagai koloninya. Tiap partikel memiliki keteraturan dalam bergerak.
3. *Swarm* bersifat *self-organizing* sehingga tiap individu bebas mengatur pola pergerakan masing-masing. Pola pergerakan *swarm* muncul sebagai akibat dari proses komunikasi antar individu dalam *swarm*.

Salah satu algoritma *swarm intelligence* yang cukup banyak digunakan adalah *Particle Swarm Optimization* (PSO). PSO berdasarkan pada teknik optimasi stokastik yang dikembangkan oleh Dr Eberhart dan Dr Kennedy pada tahun 1995,

terinspirasi oleh perilaku sosial dari kelompok burung (*flock of bird*) atau sekumpulan ikan (*school of fish*). PSO merupakan algoritma berbasis populasi yang mengeksploitasi individu dalam populasi menuju daerah penyelesaian dalam daerah pencarian.

Dalam PSO populasi disebut dengan *swarm* dan individu disebut dengan particle. Tiap partikel berpindah dengan kecepatan yang diadaptasi dari daerah pencarian dan menyimpannya sebagai posisi terbaik yang pernah dicapai.

Algoritma dasar PSO terdiri dari tiga tahap, yaitu:

1. Pembangkitan posisi serta kecepatan partikel
2. *Update velocity* (perbaruan kecepatan), dan
3. *Update position* (perbaruan posisi).

Tiga tahapan tersebut akan diulang sampai kriteria kekonvergenan terpenuhi, kriteria kekonvergenan sangat penting dalam menghindari penambahan fungsi evaluasi setelah solusi optimum didapatkan, namun kriteria kekonvergenan tidak selalu mutlak diperlukan, penetapan jumlah iterasi maksimal juga dapat digunakan sebagai *stopping condition* dari algoritma ini.

Setiap partikel akan melacak nilai koordinat dalam ruang masalah yang berhubungan dengan solusi terbaik yang telah dicapai olehnya. Nilai terbaik sejauh yang diperoleh oleh setiap partikel ini disebut *pbest*. Sedang nilai terbaik sejauh yang diperoleh oleh setiap partikel di tetangga partikel disebut *lbest*. Saat sebuah partikel mengambil semua populasi sebagai tetangga topologinya, nilai terbaik tersebut adalah sebuah terbaik global dan disebut *gbest*.

Pada kondisi awal setiap partikel terbang bebas dalam posisi yang acak. Konsep optimasi partikel terdiri dari kerumunan, pada setiap langkah waktu, mengubah kecepatan (mempercepat) setiap partikel terhadap perubahan *pbest* dan lokasi *lbest* (versi lokal dari PSO). Percepatan dibobot dengan istilah acak, dengan nomor acak terpisah yang dihasilkan untuk percepatan ke arah *pbest* dan lokasi *lbest*.

Pada algoritma PSO, parameter dasar yang pasti digunakan adalah posisi partikel, *inertia weight*, konstanta, kecepatan partikel, posisi terbaik partikel dan posisi global terbaik.

Dikarenakan PSO merupakan algoritma berbasis populasi. Populasi ini disebut dengan *swarm* dan individu yang disebut partikel.

Suatu populasi *swarm* dalam ruang pencarian A didefinisikan sebagai suatu set kelompok himpunan $S = \{ x_1, x_2, x_3, \dots , x_n \}$ dengan N partikel (kandidat solusi), yang didefinisikan sebagai:

$$x_i = \{ x_{i1}, x_{i2}, x_{i3}, \dots , x_{in} \}^T \in A \text{ dengan } i = 1,2,\dots, N. \quad (2.57)$$

Indeks secara bebas diberikan untuk partikel, sedangkan *N* adalah parameter yang diberikan dari algoritma. Fungsi tujuan (*fitnees function*), f(x), diasumsikan akan tersedia untuk semua titik dalam A.

Dengan demikian, setiap partikel memiliki nilai fungsi unik $f_i = f(x_i) \in Y$. Partikel diasumsikan bergerak dalam ruang pencarian A secara iteratif. Hal ini dimungkinkan dengan menyesuaikan posisi mereka menggunakan pergeseran posisi yang tepat, disebut kecepatan, dan dilambangkan sebagai

$$v_i = \{ v_{i1}, v_{i2}, v_{i3}, \dots , v_{in} \}^T \quad (2.58)$$

Kumpulan posisi tiap-tiap partikel P didefinisikan dengan

$$P = \{ p_1, p_2, p_3, \dots , p_N \} \quad (2.59)$$

yang berisi posisi terbaik

$$p_1 = \{ p_{i1}, p_{i2}, p_{i3}, \dots , p_{iN} \}^T \in A \text{ dengan } i = 1,2,\dots, N \quad (2.60)$$

Posisi terbaik tiap-tiap partikel dalam ruang A ini didefinisikan dengan persamaan

$$p_i(t) = \text{argmin } f_i(t). \quad (2.61)$$

Posisi *global minimum* ($g_{\min}$) didefinisikan dengan persamaan

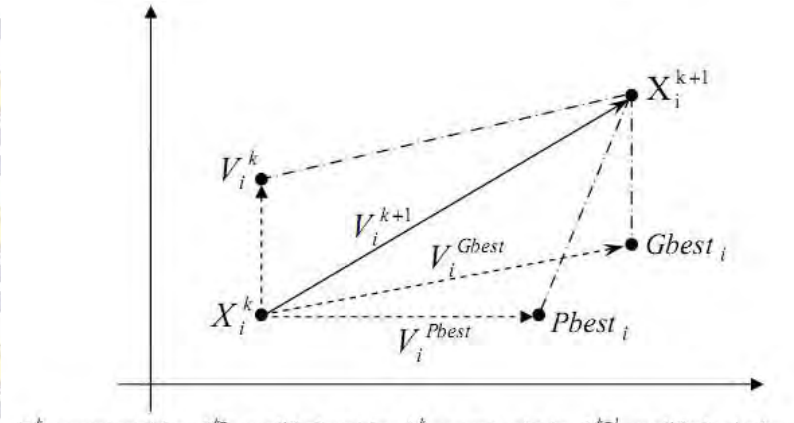
$$p_g(t) = \text{argmin } f_i(p_i(t)) \text{ dengan t adalah jumlah iterasi} \quad (2.62)$$

Update posisi dilakukan secara serempak di tiap tiap partikel. Update partikel ini dilakukan dengan menggunakan persamaan dasar PSO, yaitu:

$$\mathcal{V}_{ij}(t+1) = \omega \mathcal{V}_{ij}(t) + c_1 r_1 \left(P_i(t) - X_{ij}(t) \right) + c_2 r_2 \left(\mathcal{G}_{gj}(t) - X_{ij}(t) \right) \quad (2.63)$$

Sedangkan pemutakhiran posisi partikel dapat dilakukan dengan menggunakan persamaan:

$$X_{ij}(t+1) = X_{ij}(t) + \mathcal{V}_{ij}(t+1) \quad (2.64)$$



(X^k: current position, X^{k-} : modified position, V^k: current velocity, V^{k-1}: modified velocity, V^{Pbest}: velocity based on *Pbest*, V^{Gbest}: velocity based on *Gbest*)

Gambar 2.9. *Update* Posisi *Particle Swarm Optimization*

Dalam penerapannya, PSO memiliki beberapa parameter yang dapat diatur, yaitu: [17]

1. Jumlah Partikel.

   Berkisar antara 20 sampai 40. Carlisle menyarankan jumlah partikel sekitar 30, cukup kecil untuk efisiensi waktu dan sudah cukup besar untuk menghasilkan solusi yang baik (mendekati optimum *global*)

2. Dimensi partikel

   Bergantung pada besarnya variabel masalah yang ingin dioptimasi.

3. Rentang nilai partikel

   Juga bergantung pada masalah yang ingin dioptimasi.

4. Kecepatan maksimum partikel (*Vmax*)

   Variabel ini menentukan perubahan maksimum yang bisa dilakukan oleh suatu partikel dalam satu iterasi.

5. *Learning Rate* φ (Laju belajar)

   Umumnya dipakai $\varphi_1 = \varphi_2 = 2{,}05$ (dengan $\varphi_1 + \varphi_2 = 4{,}1$). Carlisle menyarankan $\varphi_1 = 2{,}8$ dan $\varphi_2 = 1{,}3$ dengan sedikit modifikasi pada persamaan *update velocity*.

6. Kondisi berhenti

   Kondisi yang tercapai untuk membuat proses *looping* PSO dihentikan, biasanya jika sudah mencapai nilai iterasi maksimum atau mencapai nilai minimum yang diinginkan.

7. Versi global atau versi lokal

   Versi global lebih cepat, tetapi mungkin kovergen pada optimum lokal, sedangkan versi lokal lebih lambat, tetapi tidak mudah terjebak pada optimum lokal.
8. *Synchronus* atau *asynchronus update*.

2.2.4.2 Algoritma PSO

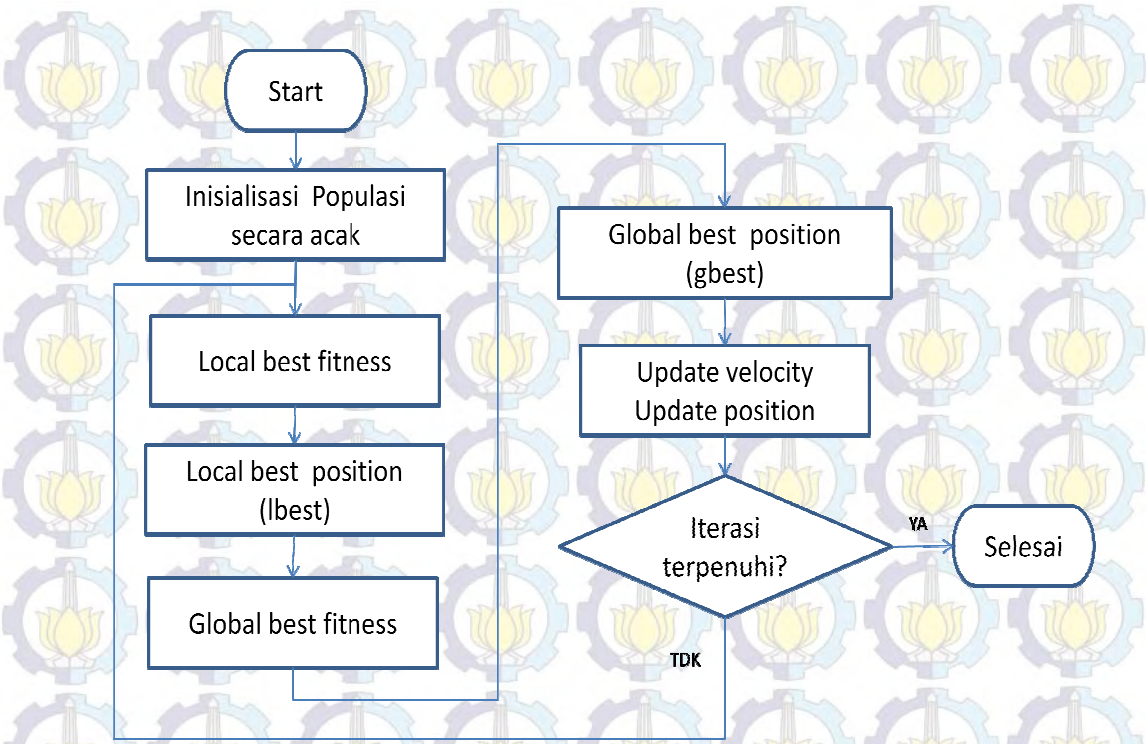
Pada implementasi algoritma PSO dasar, ada beberapa langkah dasar yang selalu dilakukan yaitu:

1. Inisialisasi populasi secara acak.
2. Melakukan perhitungan nilai tujuan dari tiap partikel (*fitness value*) berdasarkan fungsi tujuan (*fitness value*) yang diinginkan.
3. Dari perhitungan nilai *fitness*, dapat diketahui *local best fitness* dan *local best position*.
4. Mencari nilai *global best fitness*, yaitu nilai minimum dari *local best fitness* dengan menggunakan Persamaan (2.61).
5. Menentukan *global best position* menggunakan Persamaan (2.62), dengan mengganti tiap kandidat solusi partikel dengan *local best position* dari partikel yang memenuhi persyaratan *global best fitness*.
6. Memperbarui kecepatan (*update velocity*) dan posisi (*update position*) partikel dengan menggunakan Persamaan (2.63) dan (2.64).
7. Ulangi langkah 2 sampai 6 sehingga memenuhi iterasi atau diperoleh nilai kondisi berhenti yang telah ditentukan.

Adapun *flowchart* algoritma PSO dasar seperti tampak pada Gambar 2.10.

2.2.4.3 *Unified Particle Swarm Optimization* (uPSO) [18]

Algortima PSO sebagai salah satu teknik optimasi stokastik yang cukup banyak digunakan, dalam penerapannya banyak mengalami perkembangan dan pemutakhiran algoritma dengan tujuan untuk menyempurnakan proses pencarian nilai optimal pada suatu fungsi kendala yang semakin rumit. Beberapa algoritma yang merupakan pengembangan dari PSO seperti CPSO (*Canonical* PSO), *Off the Shelf* PSO (OTSPSO), *Neighborhood Best* PSO (NBPSO), dan lain seba*gain*ya.



Gambar 2.10. *Flowchart* PSO

UPSO merupakan salah satu pengembangan dari algoritma dasar PSO. uPSO diperkenalkan oleh K. E. Parsopoulos dan M. N. Vrahatis untuk menyelesaikan optimalisasi masalah-masalah keteknikan dengan syarat batas/kendala.

Berbeda dengan PSO dasar, pada uPSO juga diperhitungkan posisi terbaik ketetanggaan. Algoritma uPSO diusulkan sebagai sebuah alternative dengan menggabungkan sifat eksplorasi dan eksploitasi yang merupakan sifat dari dua varian PSO lokal (NBPSO) dan varian PSO global

Update kecepatan global (eksploitasi) dilakukan dengan menggunakan persamaan dasar PSO, yaitu:

$$\mathcal{G}_i(t+1) = \chi\left[w\mathcal{V}_i(t) + c_1 r_1\left(P_i(t) - X_i(t)\right) + c_2 r_2\left(P_{bg}(t) - X_i(t)\right)\right] \quad (2.65)$$

Update kecepatan lokal (eksplorasi) dilakukan dengan memperhitungkan posisi terbaik ketetanggaan menggunakan persamaan berikut:

$$\mathcal{L}_i(t+1) = \chi\left[w\mathcal{V}_i(t) + c_1 r_1\left(P_i(t) - X_i(t)\right) + c_2 r'_2\left(P_{bN}(t) - X_i(t)\right)\right] \quad (2.66)$$

Agregasi dari arah pencarian didefinisikan oleh Persamaan. (2.60) dan (2.61) menghasilkan persamaan update kecepatan untuk uPSO dengan persamaan:

$$\mathcal{V}_i(t+1) = u\mathcal{G}_i(t+1) + (1-u)\mathcal{L}_i(t+1), \qquad u \in [0,1] \tag{2.67}$$

Dengan u adalah *unified faktor* yang besarnya dari 0 sampai 1. Jika dipilih nilai u=1, maka algoritma pencarian akan berlaku seperti algoritma PSO varian global, sedang jika u=0 maka pencarian akan seperti algoritma PSO varian lokal. Besarnya nilai u dapat dibuat konstan maupun dibuat variabel/berubah untuk setiap iterasi tergantung kebutuhan perancangan [19].

Sedangkan pemutakhiran posisi partikel dilakukan dengan menggunakan persamaan :

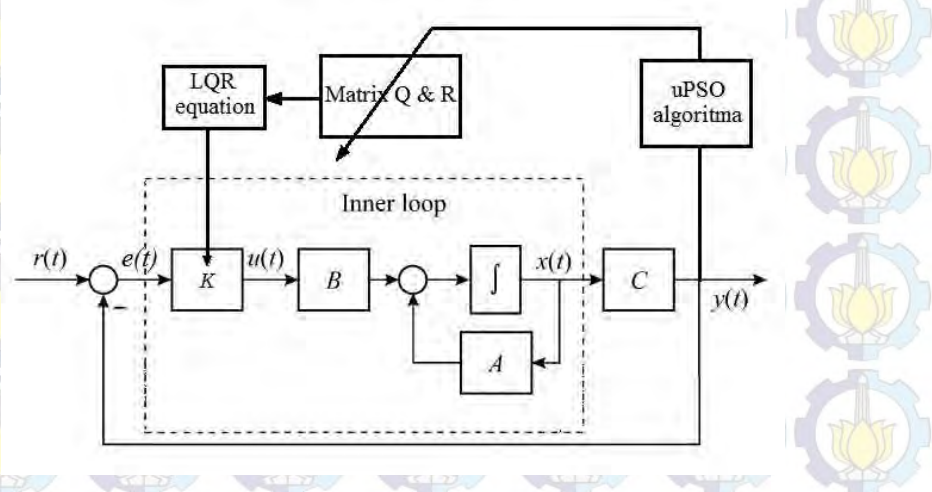
$$X_i(t+1) = X_i(t) + \mathcal{V}_i(t+1) \tag{2.68}$$

-----Halaman ini sengaja dikosongkan-----

BAB III
PERANCANGAN SISTEM

Crane merupakan salah satu contoh *plant* nonlinear dengan *Single Input Multi Output* (SIMO). Beberapa penelitian telah dilakukan untuk membuat kontroler yang bertujuan untuk mengurangi ayunan pada beban saat *crane* bekerja. Salah satu kontroler yang banyak diterapkan adalah kontroler LQR. Pada penelitian ini dirancang suatu kontroler PD-LQR dan pengembangan dari kontroler PD-LQR yaitu PD-LQR *type* I (LQRI) yang akan dioptimalkan dengan algoritma uPSO.

Algoritma uPSO dipakai untuk memilih matriks bobot *Q* dan *R* yang paling optimal pada suatu kontroler LQR. Untuk itu akan dipilih dua bentuk model dasar matriks pembobot *Q* yaitu matriks *Q*(4x4) bebas dan matriks *Q*(4x4) diagonal untuk kemudian dibandingkan dari kedua jenis matriks *Q* tersebut manakah yang memiliki *Fitness value* mencapai nilai minimum. Nilai *Q* dan *R* optimal ini akan menghasilkan *gain K* optimal sebagai pembentuk dari suatu kontroler PD-LQR dan PD-LQRI.



Gambar 3.1. Blok Diagram Kontroler PD-LQR Optimasi UPSO

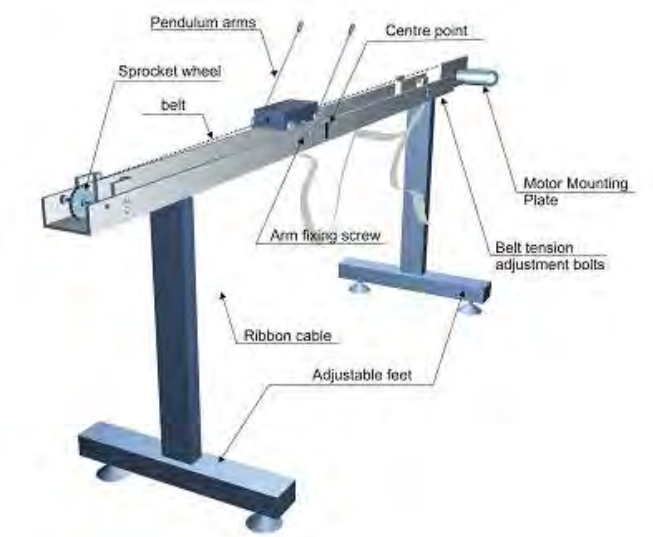
Desain kontroler PD-LQR optimal yang dihasilkan ini akan dibandingkan dengan kontroler yang sudah pernah diteliti sebelumnya seperti kontroler PD *basic* dan kontroler SMC-PI, selain itu juga akan dibandingkan dengan kontroler PD-LQR dengan matriks *Q* dan *R* diperoleh dengan metode *trial and error* (TEM).

Selanjutnya kontroler PD-LQR optimal akan dikembangkan menjadi kontroler PD-LQR *type* I, dan akan dibandingkan antara kontroler PD-LQR

optimal dengan kontroler PD-LQR-I dan kontroler PD *State Feedback* dengan kompensator Integral.

3.1 Linearisasi *Plant* Nonlinear

Untuk tahap simulasi dan implementasi, akan digunakan *plant* berupa sistem pendulum kereta (SPK) dari *Feedback Instruments Ltd* [20]. Adapun tampilan *plant* seperti tampak pada Gambar 3.2



Gambar 3.2. Model *Digital Pendulum Mechanical Unit*

Model dinamik *crane* dapat dituliskan kembali dalam bentuk persamaan *state space*

$$\dot{x} = f(x,u) \tag{3.1}$$

Jika dipilih vektor *input state*

$$x = [x_1 \ x_2 \ x_3 \ x_4]^T \text{ dan sinyal kontrol } u = F_x \tag{3.2}$$

dengan:

$x_1 = x$

$x_5 = \theta$

$x_3 = \dot{x}$

$x_4 = \dot{\theta}$

Berdasarkan [21] untuk *plant* SPK dari *Feedback Instruments Ltd.*, jenis *Digital Pendulum Mechanical Unit* 33-200, untuk kondisi penggunaan sebagai *crane* dimana untuk $\theta = \pi$ diperoleh persamaan *state-space* sebagai berikut:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_3 \\ x_4 \\ \dfrac{\alpha(F_x - T_c - \mu x_4{}^2 \sin x_2) - l\cos x_2(\mu g \sin x_2 - f_p x_4)}{(J + \mu l \sin^2 x_2)} \\ \dfrac{-l\cos x_2(F_x - T_c - \mu x_4{}^2 \sin x_2) + \mu g \sin x_2 - f_p x_4}{(J + \mu l \sin^2 x_2)} \end{bmatrix} \tag{3.3}$$

dengan:

$\mu = (m_c + m_p)l$ dan $\alpha = l^2 + \dfrac{J}{m_c + m_p}$

$m_c = 1{,}12 \ kg$

$m_p = 0{,}12 \ kg$

$l = 0{,}402$ m

$J = 0{,}0135735$ kg.m^2

$f_p = 0{,}000107$ kg.m^2/m

$T_c = 2{,}5316$ N

$\mu = 0{,}49848$

$\alpha = 0{,}00177$

Dengan menggunakan metode Matriks Jacobian, persamaan *state* diatas dilinearisasikan disekitar titik asal (0,0) dengan menggunakan Persamaan (3.4) berikut.

$$A = \frac{\partial f(x)}{\partial x}|_{x=0} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \frac{\partial f_1}{\partial x_4} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \frac{\partial f_2}{\partial x_4} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \frac{\partial f_3}{\partial x_4} \\ \frac{\partial f_4}{\partial x_1} & \frac{\partial f_4}{\partial x_2} & \frac{\partial f_4}{\partial x_3} & \frac{\partial f_4}{\partial x_4} \end{bmatrix} \quad \text{dan} \quad B = \frac{\partial f(x)}{\partial u}|_{x=0} = \begin{bmatrix} \frac{\partial f_1}{\partial u} \\ \frac{\partial f_2}{\partial u} \\ \frac{\partial f_3}{\partial u} \\ \frac{\partial f_4}{\partial u} \end{bmatrix} \tag{3.4}$$

Akan diperoleh matriks *state* A dan matriks *state* B hasil linearisasi pada titik (0,0,0,0) sebagai berikut:

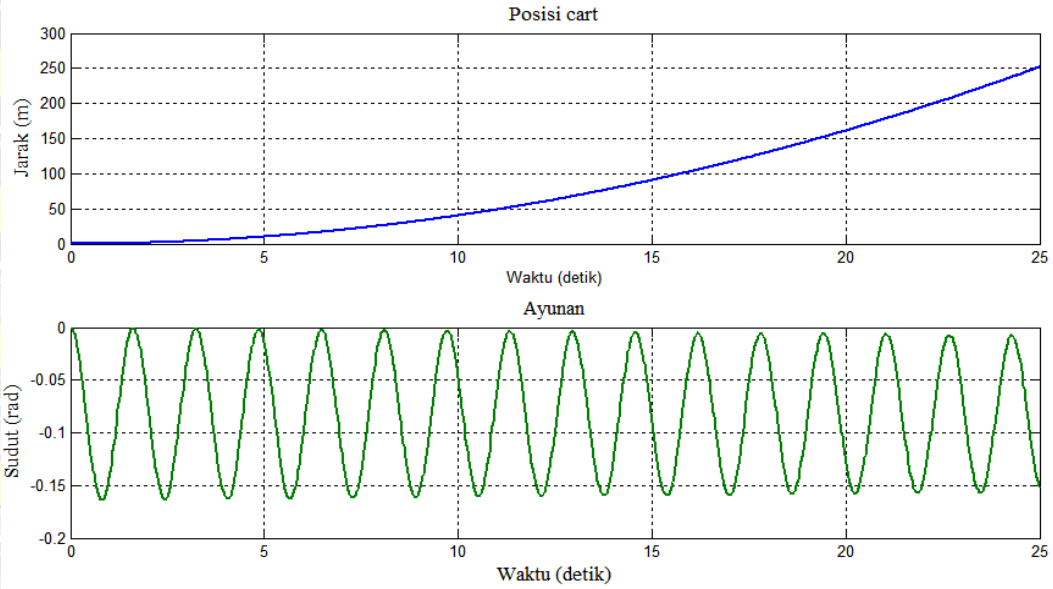
$$A=\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0{,}2525 & 0 & 0{,}00013 \\ 0 & -15{,}0421 & 0 & -0{,}0079 \end{bmatrix} \text{ dan } B=\begin{bmatrix} 0 \\ 0 \\ 0{,}8272 \\ -1{,}2396 \end{bmatrix} \quad (3.5)$$

3.1.1 Karakteristik *Open Loop*

Untuk mengetahui kestabilan suatu sistem, dapat diketahui dengan melihat posisi akar-akar matriks *state* A. Posisi ini dapat diketahui dengan melihat nilai eigen matriks A. Untuk sistem *crane* dengan matriks *state* A sesuai Persamaan (3.5) diperoleh:

$$eigen\ A=\begin{bmatrix} 0 \\ 0 \\ -0.0039+3.8771i \\ -0{,}0039-3{,}8771i \end{bmatrix}$$

Dapat dilihat jika sistem *crane* termasuk sistem dengan *output* yang tidak stabil dan berosilasi. respon *step* model linearisasi *crane* SPK *Digital Pendulum Mechanical Unit* 33-200, seperti Gambar 3.3 berikut.



Gambar 3.3. *Step Respon* Model Linearisasi SPK

3.2 Perancangan Kontroler PD-LQR *Type* 0

Untuk mendesain suatu kontroler PD-LQR (*type* 0), berikut adalah langkah-langkahnya:

a. Mengubah persamaan nonlinier *plant* menjadi bentuk persamaan linier melalui mekanisme linierisasi pada titik tertentu sehingga diperoleh persamaan *state*: $\dot{x} = Ax + Bu$

b. Menentukan matriks pembobot Q dan R berdasarkan minimisasi fungsi kriteria energi minimum melalui indeks performansi kuadratik berikut ini

$$J(t) = \frac{1}{2}[x^T(t_a)S.x(t_a)] + \frac{1}{2}\int[x^T(t)Qx(t) + u^2(t)R]dt \quad (3.6)$$

c. Menentukan matriks S dari persamaan Riccati sebagai berikut:

$$A^T S(t) + S(t)A + Q - S(t)BR^{-1}B^T S(t) = 0 \quad (3.7)$$

d. Menghitung nilai *gain* K dan diperoleh persamaan sinyal kontrol

$$K = -R^{-1}B^T S(t) \quad (3.8)$$

$$u = -K.(x - x_{ref})$$

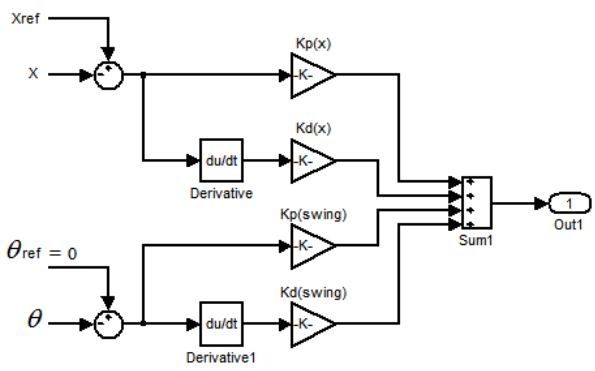
$$u = -K.x_{ss} = -K.e_{ss} \quad (3.9)$$

e. Untuk penerapan pada kontroler PD, untuk $e_{ss} = -e_{ref}$, maka berlaku

$$u = -K.-e_{ref} = K.e_{ref} \quad (3.10)$$

dimana $K = [K_{px}\ K_{p\theta}\ K_{dx}\ K_{d\theta}]$

Adapun blok diagram kontroler PD-LQR dasar seperti pada Gambar 3.4 berikut.



Gambar 3.4. Kontroler PD-LQR Dasar

3.3 Penentuan Matriks Pembobot *Q* dan *R*

Nilai dari matriks pembobot *Q* dan *R* akan menentukan hasil dari persamaan Riccati pada suatu kontroler LQR, dan akan berpengaruh pada keseluruhan performansi sistem. Pada penelitian ini akan diselidiki pengaruh metode pemilihan matriks *Q*. Untuk *input state* $x = [x_1\ x_2\ x_3\ x_4]^T$ dan sinyal kontrol *u* maka diperlukan matriks bobot *Q* dengan ukuran (4x4) dan matriks *R*(1x1) dengan matriks *state Q* adalah matriks simetris semidefinit positif ($Q \geq 0$) dan matriks kontrol *R* merupakan matriks definit positif ($R > 0$).

Untuk proses *tuning* dan optimasi dengan uPSO, nilai *Q* dibatasi antara 0 sampai 100, sedang nilai *R* dibatasi dari 0,001 sampai 100. Dalam penentuan matriks pembobot *Q* dan *R* ini, ada beberapa cara yang dapat digunakan.
Pada penelitian kali akan digunakan dua metode penentuan yaitu:

- Metode *trial and error* (TEM), dan
- Metode algoritma cerdas menggunakan uPSO

3.3.1 *Trial and Error Method* (TEM)

Merupakan metode yang sederhana dan praktis, dilakukan dengan memilih komponen matriks dengan cara mencoba harga sembarang sesuai keluaran yang diinginkan dibandingkan terhadap keluaran sebelumnya.

Ada beberapa kaidah dalam menentukan matriks pembobot agar mendekati harga yang diinginkan.

- Harga matriks pembobot *Q* dipilih yang besar agar penguatan umpan balik membesar.
- Apabila matriks pembobot *R* yang dipilih besar, maka penguatan kontrol umpan balik *K* mengecil sehingga respon sistem menjadi lebih lamban.

Adapun prosedur metode *Trial and Error* (TEM), adalah sebagai berikut:

1. Pilih matriks *Q* dan *R* sehingga $Q \geq 0$ dan $R = \rho I > 0$.
2. Dapatkan solusi persamaan Riccati

   $A^T S(t) + S(t)A + Q - S(t)BR^{-1}B^T S(t) = 0$

   dan hitung $K = -R^{-1}B^T S(t)$

3. Simulasikan respon sistem $\dot{x} = (A - BK)x$ untuk tiap-tiap kondisi awal yang berbeda.
4. Jika kondisi respon sistem dan atau besaran batasan yang diinginkan tidak terpenuhi, kembali ke tahap 1 dan pilih matriks *Q* dan atau matriks *R* yang lain.

Untuk metode TEM, matriks *Q* akan dibentuk dengan 2 cara, yaitu:

1. **Dengan menggunakan bobot *output z***

   dengan $z = Hx$, maka $Q = H^T.H$ dan $R = \rho I$. (3.11)

   dan $\rho > 0$

   ρ = konstanta penyeimbang pengaturan terhadap upaya kontrol

2. **Dengan memilih matriks *Q* berupa matriks diagonal**

   Matriks $Q = \text{diag}[q] = \begin{bmatrix} q_1 & 0 & 0 & 0 \\ 0 & q_2 & 0 & 0 \\ 0 & 0 & q_3 & 0 \\ 0 & 0 & 0 & q_4 \end{bmatrix}$ dan $R = \rho[r_1]$ (3.12)

   dengan $q_i = \frac{1}{t_{si}(x_{imax}{}^2)}$ dan $r_i = \frac{1}{(u_{imax}{}^2)}$, $\rho > 0$ (3.13)

   t_{si} = *settling time output state* yang diharapkan

   x_{imax} = batasan besaran maksimum *state* x_i

   u_{imax} = batasan besaran maksimum sinyal kontrol u_i

   ρ = konstanta penyeimbang pengaturan terhadap upaya kontrol

3.3.2 Metode Algoritma Cerdas uPSO

Adapun prosedur dengan menggunakan metode algoritma uPSO adalah sebagai berikut:

1. Inisialisasi secara acak komponen matriks *Q* dan *R* dalam bentuk matriks *swarm* (partikel) dimensi x jumlah partikel, beserta dengan matriks kecepatan *swarm* masing-masing.

2. Dapatkan solusi persamaan Riccati untuk masing-masing partikel komponen matriks *Q* dan *R* tersebut dengan menggunakan persamaan:

   $A^T S(t)+S(t)A + Q - S(t)BR^{-1}B^T S(t) = 0$

   dan hitung nilai *gain* $K = -R^{-1}B^T S(t)$

3. Simulasikan respon sistem $\dot{x} = (A - BK)x$ untuk tiap-tiap pasangan *Q* dan *R* yang bersesuaian.
4. Hitung nilai *fitness* (*fitness value*) *output* dari sistem untuk tiap-tiap pasangan *Q* dan *R* tersebut. Pemberian nilai *fitness* berdasarkan kriteria *output* parameter yang diinginkan.
5. Tentukan partikel dengan nilai *fitness* terbaik dari tiap-tiap partikel *Q* dan *R* yang ada. Dimana akan dicari tiga nilai terbaik untuk tiap iterasi yaitu terbaik global (*gbest*), terbaik lokal (*lbest*), dan terbaik ketetanggaan (*nbest*)
6. *Update* kecepatan partikel dengan menggunakan persamaan:

   $$\mathcal{G}_i(t+1) = \chi\left[\mathrm{w}\mathcal{V}_i(t) + c_1 r_1\left(P_i(t) - X_i(t)\right) + c_2 r_2\left(P_{bg}(t) - X_i(t)\right)\right]$$
   $$\mathcal{L}_i(t+1) = \chi\left[\mathrm{w}\mathcal{V}_i(t) + c_1 r_1\left(P_i(t) - X_i(t)\right) + c_2 r'_2\left(P_{bN}(t) - X_i(t)\right)\right]$$
   $$\mathcal{V}_i(t+1) = u\mathcal{G}_i(t+1) + (1-u)\mathcal{L}_i(t+1), \qquad u \in [0,1] \tag{3.14}$$

7. *Update* kecepatan partikel dengan menggunakan persamaan:

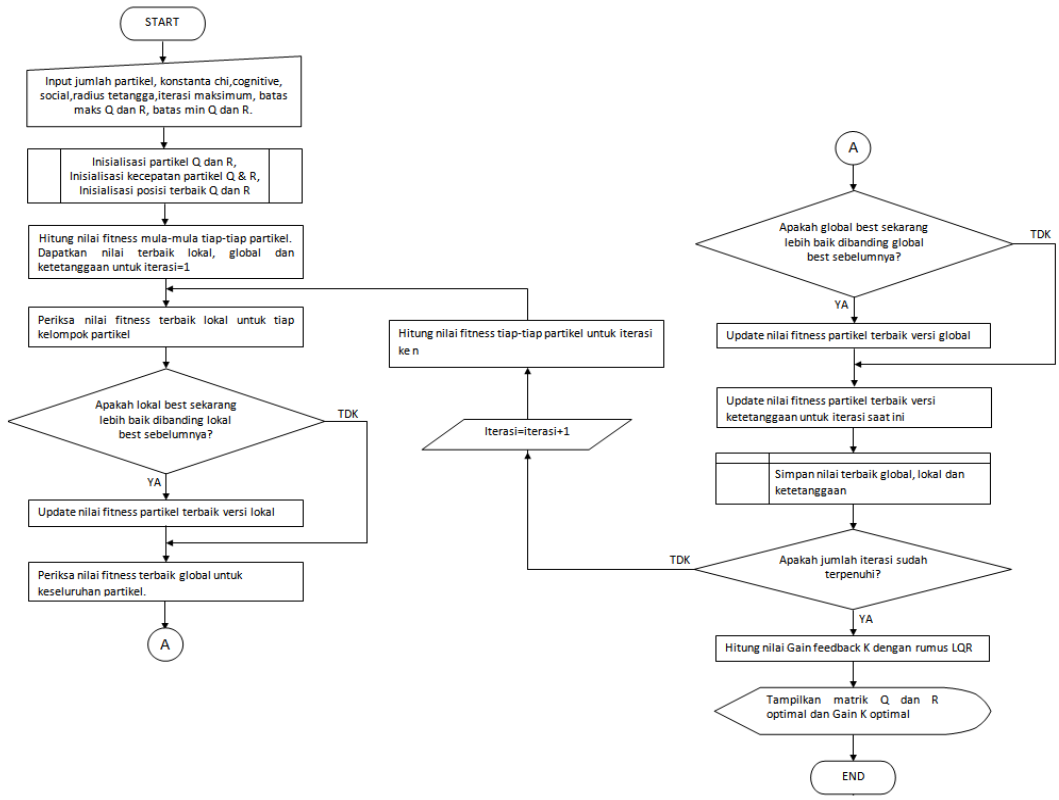
   $$X_i(t+1) = X_i(t) + \mathcal{V}_i(t+1) \tag{3.15}$$

8. Ulangi prosedur diatas dimulai dari point 2 hingga point 7 diatas sampai diperoleh nilai *fitness* minimum yang diinginkan atau jumlah maksimum iterasi sudah terpenuhi.

Pada metode uPSO ini akan dipilih dua bentuk matriks *Q* untuk di-*tuning*, yaitu:

1. Matriks *Q* berupa matriks bebas, dengan *Q*(4x4)
2. Matriks *Q* berupa matriks diagonal, dengan *Q* = diag[*q*]

Adapun *flowchart* proses *tuning* PD-LQR dengan algoritma uPSO seperti tampak pada Gambar 3.5 berikut.



Gambar 3.5. *Flowchart Tuning* PD-LQR dengan UPSO

Untuk metode uPSO ini akan dipilih dua bentuk matriks Q untuk di-*tuning*, yaitu:

1. Matriks Q berupa matriks bebas, dengan Q(4x4)

Untuk matriks Q bebas, diperlukan 10 variabel q_i, mulai dari q_1 sampai q_{10} untuk membentuk matriks Q bebas dan 1 variabel r_1 untuk matriks R.

$$\text{dengan matriks } Q = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \\ q_2 & q_5 & q_6 & q_7 \\ q_3 & q_6 & q_8 & q_9 \\ q_4 & q_7 & q_9 & q_{10} \end{bmatrix} \text{ dan } R = [r_1] \qquad (3.16)$$

sehingga pada algoritma uPSO akan di-*input*-kan jumlah variabel Q = 10 dimensi dan R = 1 dimensi, dengan total dimensi variabel uPSO = 11.

2. Matriks *Q* berupa matriks diagonal, dengan *Q* = diag[*q*(4x1)]

Untuk matriks *Q* diagonal, akan diperlukan 4 variabel penyusun, dari q_1 sampai q_4 untuk membentuk matriks *Q* diagonal dan 1 variabel *R*.

$$\text{dengan matriks } Q = \begin{bmatrix} q_1 & 0 & 0 & 0 \\ 0 & q_2 & 0 & 0 \\ 0 & 0 & q_3 & 0 \\ 0 & 0 & 0 & q_4 \end{bmatrix} \text{ dan } R = [r_1] \qquad (3.17)$$

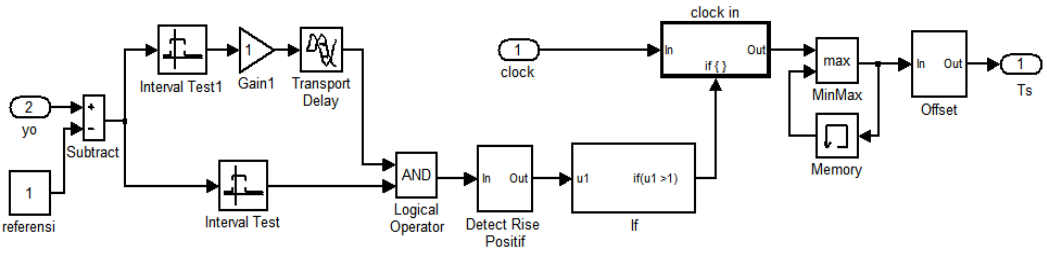
sehingga pada algoritma uPSO akan di-*input*-kan jumlah variabel *Q* = 4 dan *R* = 1, dengan total dimensi variabel uPSO = 5 dimensi.

3.4 Fungsi Tujuan (*Fitness Function*) dan Syarat Batasan (C*onstraint*)

Kontrol optimal berkaitan dengan masalah menemukan hukum kontrol untuk sistem tertentu dengan pencapaian kriteria optimalitas tertentu. Istilah optimal mempunyai maksud hasil paling baik yang dapat dicapai dengan memperhatikan kondisi dan kendala dari suatu sistem sehingga perlu didefinisikan rumusan nilai *fitness* yang sesuai. Hal ini harus mengacu pada tujuan pengoptimalan sistem tersebut. Untuk parameter dengan tujuan pengoptimalan utama (pada penelitian ini adalah ayunan beban *crane*) akan diberikan bobot yang lebih besar dibandingkan parameter lainnya.

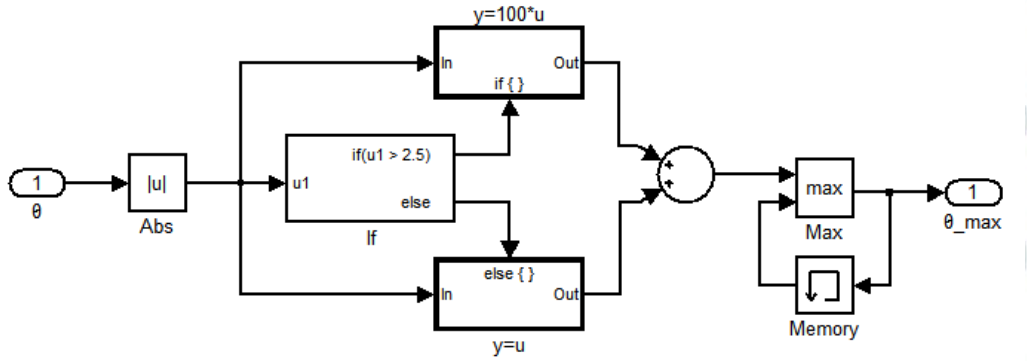
Adapun beberapa parameter yang ingin dioptimalkan pada penelitian kali ini yaitu:

1. ***Settling Time* (Ts),** merupakan waktu yang diperlukan *crane* untuk bergerak dari titik awal ke titik tujuan untuk memindahkan muatan beban yang dibawa dimana semakin kecil nilai Ts semakin baik. Adapun blok *sub-system* simulink di Matlab untuk mendeteksi *settling time* seperti tampak pada Gambar 3.6 berikut.



Gambar 3.6. Blok Simulink Detektor *Settling Time*

2. **Ayunan Maksimum (*Max Swing*),** dengan batasan ayunan maksimum yang diterima kurang atau sama dengan 0,0436 rad (2,5 derajat). Jika ayunan beban lebih dari 0,0436 rad (2,5 derajat) maka nilai parameter *swing* akan dikali 100. Parameter ini akan dideteksi oleh blok *sub-system* simulink di Matlab seperti tampak pada Gambar 3.7 berikut.

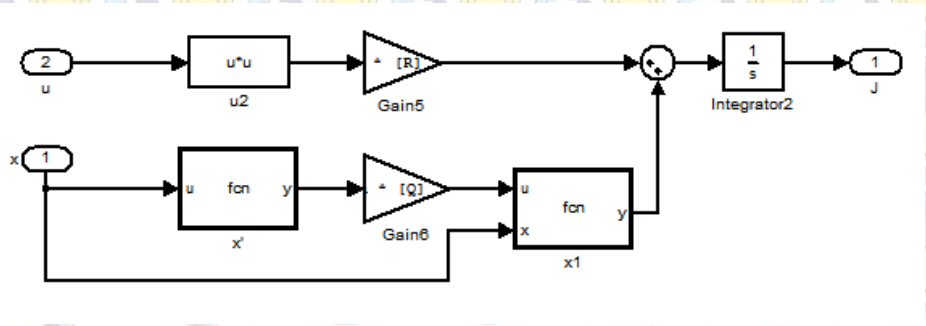


Gambar 3.7. Blok Simulink Detektor *Max Swing*

Di mana pada tahapan ini digunakan mekanisme pemberian nilai parameter dengan persamaan:

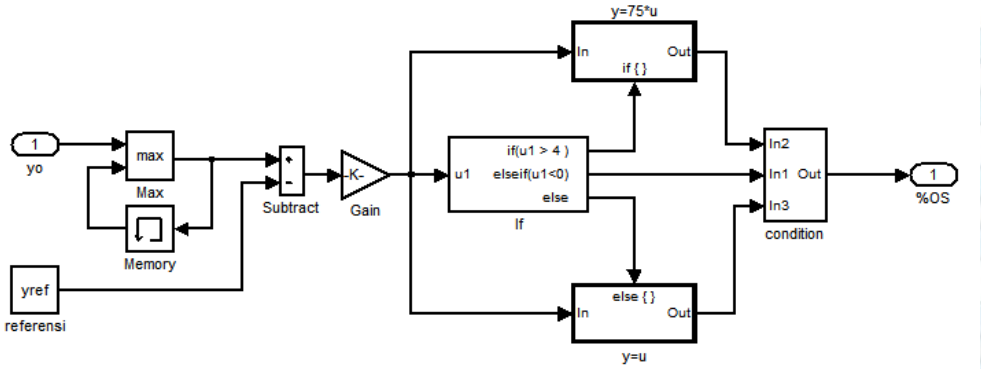
$$\theta\_max = f(\theta) = \begin{cases} \theta, & if\ \theta \leq 0{,}0436\ rad\ (2{,}5\ derajat) \\ 100 * \theta, & if\ \theta > 0.0436\ rad\ (2{,}5\ derajat) \end{cases} \quad (3.18)$$

3. ***Index Performance* (*J*),** merupakan tujuan utama dari suatu kontroler LQR yaitu untuk meminimumkan fungsi energi sistem, dimana semakin kecil nilai J maka *output* sistem semakin baik.



Gambar 3.8. Blok Simulink Detektor *Index Performance*

4. **Persentase *Overshoot* (%OS),** spesifikasi respon sistem yang diamati mulai saat *crane* melampaui jarak referensi sampai mencapai simpangan terjauhnya, dimana tolak ukur dari %OS ini adalah jika melebihi batas 4% akan ditolak. Adapun blok sub-sistem simulink untuk mendeteksi %OS adalah sebagai berikut

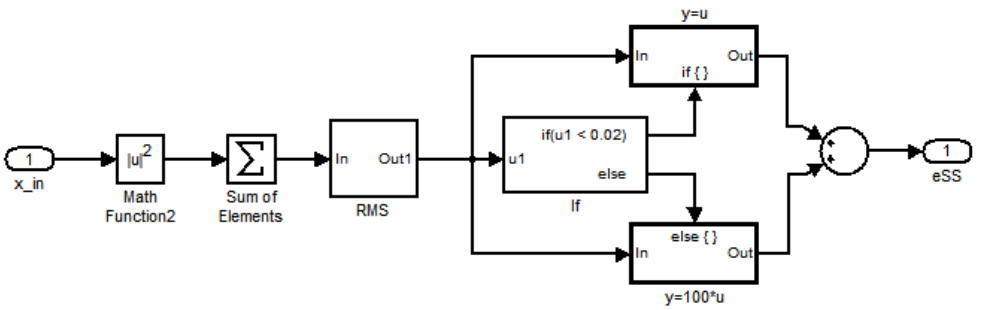


Gambar 3.9. Blok Simulink Detektor %OS

Di mana pada tahapan ini digunakan mekanisme pemberian nilai parameter dengan persamaan:

$$\%OS = f(y_o) = \begin{cases} 0, & if\ y_o \leq 0 \\ y_o, & if\ 0 < y_o < 4\% \\ 75 * y_o, & if\ y_o \geq 4\% \end{cases} \quad (3.19)$$

5. ***Steady State Error* (e_{ss}),** merupakan spesifikasi respon sistem yang diamati mulai saat respon masuk dalam keadaan *steady state* sampai waktu tak terbatas. Tolak ukur yang diharapkan adalah *steady state error* sistem dibatasi tidak melebihi 0,02 (2%) dari nilai referensi. Berikut blok sub-sistem simulink untuk mendeteksi *steady state error*:



Gambar 3.10. Blok Simulink Detektor *Steady State Error*

dengan mekanisme pemberian nilai parameter dengan persamaan:

$$Ess = f(x) = \begin{cases} x, & if\ x < 0{,}02 \\ 100.x, & if\ x \geq 0{,}02 \end{cases} \quad (3.20)$$

Kelima parameter ini akan membentuk fungsi tujuan (*fitnees function)* parameter *output* sistem dengan persamaan:

$$Fitness = \frac{\omega_1.Ts}{N_1} + \frac{\omega_2.Swing}{N_2} + \frac{\omega_3.Jm}{N_3} + \frac{\omega_4.\%OS}{N_4} + \frac{\omega_5.ESS}{N_5} \quad (3.21)$$

$$\omega_1 + \omega_2 + \omega_3 + \omega_4 + \omega_5 = 100\%$$

Dengan $\omega_1, \omega_2, \omega_3, \omega_4$ dan ω_5 adalah bobot nilai tujuan (*fitness value*) parameter yang ingin dioptimalkan sedang N_1, N_2, N_3, N_4 dan N_5 adalah faktor normalisasi parameter.

Untuk penelitian ini dipilih nilai bobot optimalisasi sebagai berikut:

ω_1=ω_3=ω_4=20%,

ω_2=30%, dan ω_5=10%

Sedangkan berdasarkan kriteria batasan (*constraint*) performansi yang diinginkan dari *output* sistem, maka dipilih nilai normalisasi sebagai berikut:

- Faktor normalisasi *settling time*, N_1 = 20 detik dengan melihat unjuk kerja sistem adalah baik jika *settling time* besarnya tidak lebih dari 20 detik.
- Faktor normalisasi *Swing*, N_2 = 0,0436 rad (2,5 derajat) dengan harapan bahwa kriteria swing pada beban dibatasi tidak melebihi sebesar 0,0436 rad.
- Faktor normalisasi Indeks Performansi, N_3 = 10 J, dimana jumlah energi rata-rata *state* diharapkan tidak lebih besar dari 10 J.
- Faktor normalisasi *%OS*, N_4 = 4% dengan harapan bahwa besarnya o*vershoot* pada *crane* saat melampaui titik tujuan tidak lebih dari 4% dari nilai referensi.
- Faktor normalisasi *Ess*, N_5 = 0,02 m dengan harapan bahwa kriteria besar *steady state error* pada *crane* tidak lebih atau kurang dari sebesar 0,002 m dari titik tujuan referensi.

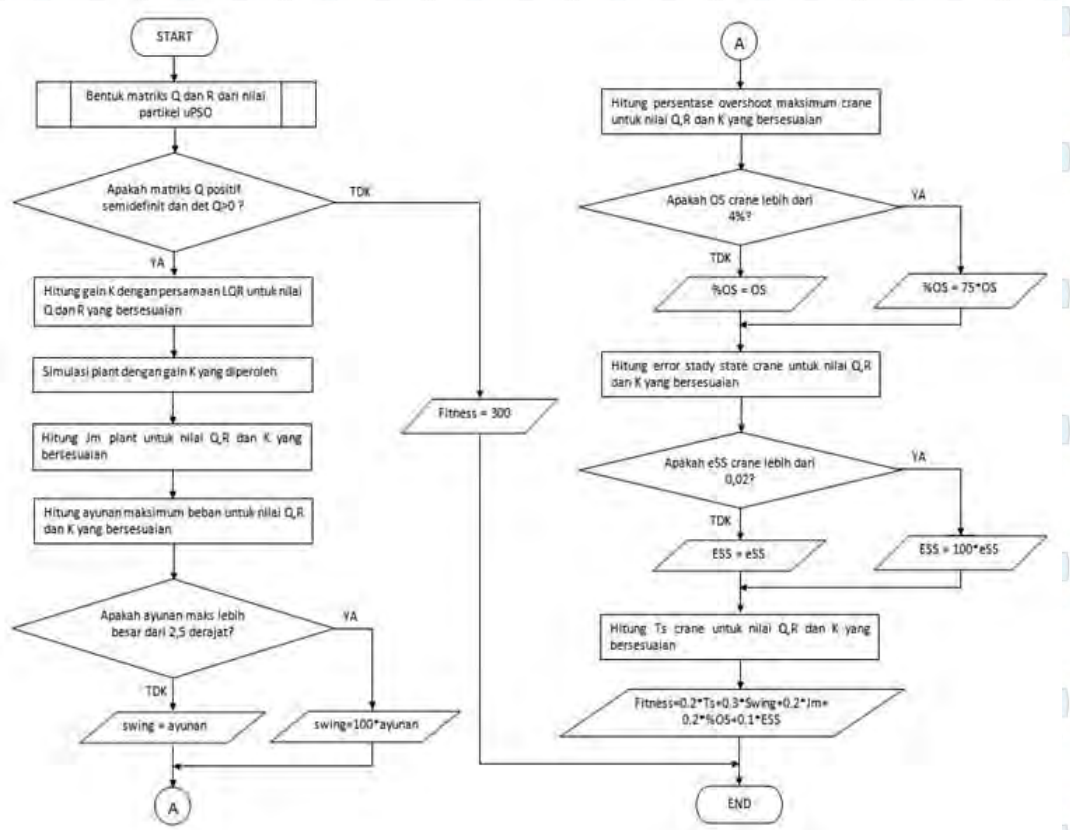
6. Syarat Matriks *Q* Semidefinit Positif dan Matriks *R* Definit Positif

Selain syarat nilai batas parameter yang diinginkan di atas, syarat lain yang harus dipenuhi untuk membentuk kontroler PD-LQR ialah matriks *Q* merupakan

matriks semidefinit positif dan R matriks definit positif. Jika nilai Q dan R yang dihasilkan tidak memenuhi persyaratan ini, maka pada algoritma uPSO akan memberikan nilai *fitness* kepada *swarm* sebesar 300. Untuk persyaratan ini digunakan persamaan:

$$Fitnees\ Value = \begin{cases} 300, & if\ Q < 0\ or\ R \leq 0 \\ Fitness\ (Pers.\ 3.21), & if\ Q \geq 0\ and\ R > 0 \end{cases} \quad (3.22)$$

Adapun *flowchart* proses pemberian nilai *fitness* dapat dilihat pada Gambar 3.11.



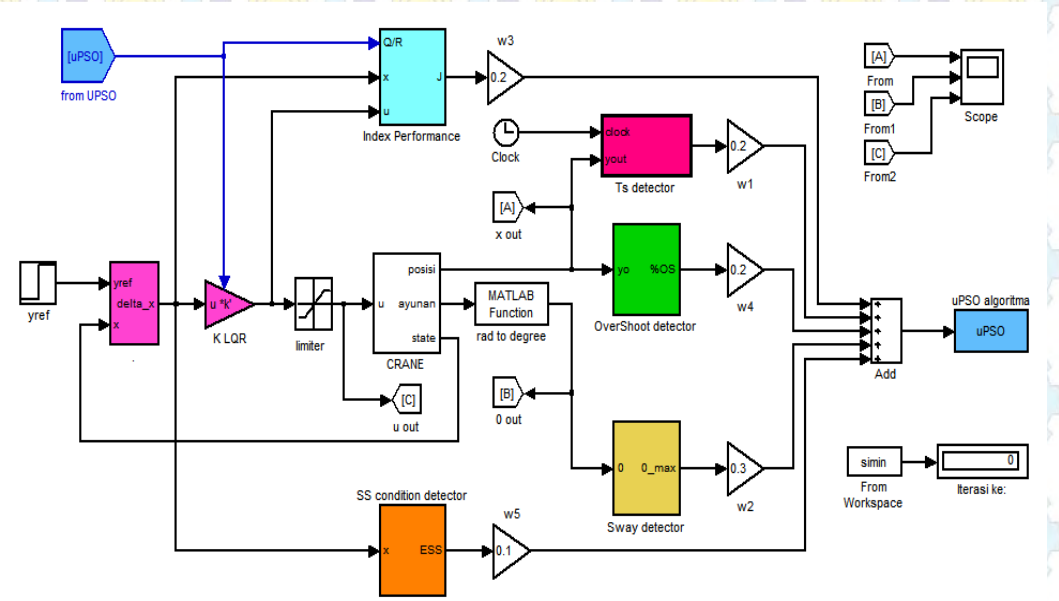
Gambar 3.11. *Flowchart* Perhitungan Nilai *Fitness* pada UPSO

3.5 Simulasi Sistem dan Desain Program PD-LQR Optimasi uPSO

Untuk membentuk kontroler PD-LQR yang dioptimalkan dengan algoritma uPSO, perlu diperoleh nilai Q dan R yang optimal sehingga dihasilkan nilai *gain K* optimal yang akan diimplementasikan pada kontroler *real*. Dalam proses pencarian (*tuning*) bobot matriks Q dan R optimal, diperlukan suatu aplikasi dan model simulasi yang akan

melakukan proses tersebut. Pada penelitian kali ini digunakan software simulink Matlab untuk proses simulasinya dan aplikasi GUI Matlab untuk aplikasi antarmukanya.

Gambar 3.12 berikut ini merupakan blok simulink untuk proses simulasi optimasi kontroler PD-LQR.



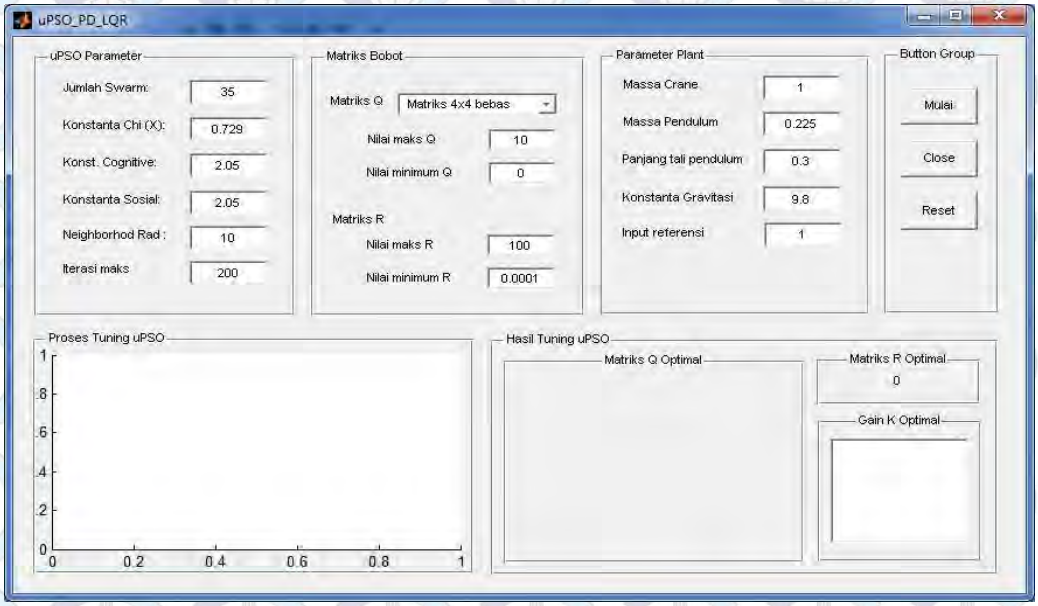
Gambar 3.12. Model Simulink Optimasi Kontrol PD-LQR dengan UPSO

Untuk memudahkan proses optimasi kontroler PD-LQR, program aplikasi dibuat dalam bentuk *Graphic User Interface* (GUI). Adapun tampilan aplikasi GUI program optimasi PD-LQR dengan uPSO di Matlab dapat dilihat pada Gambar 3.13.

Pada aplikasi GUI ini terdapat 6 panel utama., yaitu:

1. Panel pertama merupakan panel uPSO parameter, dipakai untuk mengisi nilai-nilai parameter uPSO yang diterapkan. Ada empat *input box* .
2. Panel kedua merupakan panel matriks bobot, dipakai untuk mengisi batasan besar maksimum dan minimum komponen penyusun matriks bobot *Q* dan matriks *R*. Selain itu terdapat juga *pull-down menu* yang dipakai untuk memilih jenis matris *Q* yang akan digunakan dan juga jenis kontroler LQR (*type* 0 atau *type* I).
3. Panel ketiga merupakan panel Parameter *plant*, berupa kolom isian nilai-nilai parameter *plant* seperti massa *crane*, massa beban, panjang tali dan konstanta gravitasi.

4. Panel keempat merupakan panel *button group*, yang berisi tombol-tombol perintah diantaranya tombol mulai untuk eksekusi, *close* untuk menutup dan *reset* untuk mengembalikan nilai parameter ke nilai *default*.
5. Panel kelima merupakan panel untuk menampilkan proses berlangsungnya *tuning* PD-LQR-uPSO. Disini akan ditampilkan grafik yang menggambarkan progres pencapaian nilai optimal tiap-tiap iterasi.
6. Panel keenam adalah panel hasil *tuning* uPSO, yang merupakan bagian yang akan menampilkan hasil *output* optimasi, dimana yang ditampilkan adalah matriks *Q* optimal, matriks *R* optimal dan *gain K* optimal.



Gambar 3.13. Tampilan GUI Aplikasi *Tuning* PD-LQR-UPSO

3.6 Perancangan Kontroler PD-LQR *Type* I (PD-LQRI)

Kontroler ini merupakan pengembangan dari kontroler PD-LQR dasar (*type*-0).

Adapun prosedur perancangan kontroler jenis ini adalah sebagai berikut:

1. Mengubah persamaan nonlinier *plant* menjadi bentuk persamaan linier melalui mekanisme linierisasi pada titik tertentu sehingga diperoleh persamaan *state*:

$$\dot{x} = Ax + Bu$$
$$y = Cx$$

2. Persamaan *state* pada point satu diatas ditransformasikan menjadi persamaan *augmented state* $\dot{X}(t) = \tilde{A}_a X(t) + \tilde{B}_a u(t)$ dengan formula sebagai berikut:

$$\begin{bmatrix} \dot{\tilde{x}}(t) \\ \dot{\tilde{\varepsilon}}(t) \end{bmatrix} = \begin{bmatrix} A & 0 \\ -C & 0 \end{bmatrix} \begin{bmatrix} \tilde{x}(t) \\ \tilde{\varepsilon}(t) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(t) \quad (3.23)$$

3. Menentukan matriks pembobot Q_a dan R berdasarkan minimisasi fungsi kriteria energi minimum melalui indeks performansi kuadratik berikut:

$$J(t) = \int_{t_0}^{\infty} [[\tilde{x}(t)^T \tilde{\varepsilon}(t)^T]^T . \tilde{Q}_a . [\tilde{x}(t)^T \tilde{\varepsilon}(t)^T] + u(t)^T R . u(t)] dt \quad (3.24)$$

4. Untuk memudahkan pemilihan matriks *Qa* yang optimal untuk kontroler PD-LQRI, dapat dipilih matriks pembobot Q_{opt} pada kontroler PD-LQR dasar, selanjutnya matriks Q_{opt} tersebut ditransformasikan menjadi matriks Q_a dengan persamaan:

$$\tilde{Q}_a = \begin{bmatrix} Q_{opt} & 0 \\ 0 & q_i \end{bmatrix} \quad (3.25)$$

Untuk memperoleh Q_a yang optimal, maka q_i akan di-*tuning* dengan menggunakan algoritma uPSO.

5. Menentukan matriks *S(t)* dari persamaan Riccati berikut:

$$\tilde{A}_a^T S(t) + S(t) \tilde{A}_a + \tilde{Q}_a - S(t) \tilde{B}_a R^{-1} \tilde{B}_a^T S(t) = 0 \quad (3.26)$$

6. Menghitung nilai *gain K* untuk mendapatkan sinyal kontrol berikut:

$$K = [K_{opt} \;\; K_i] = R^{-1} \tilde{B}_a^T S(t) \quad (3.27)$$

$$u = -K.(x - x_{ref})$$

$$u = K.(x_{ref} - x)$$

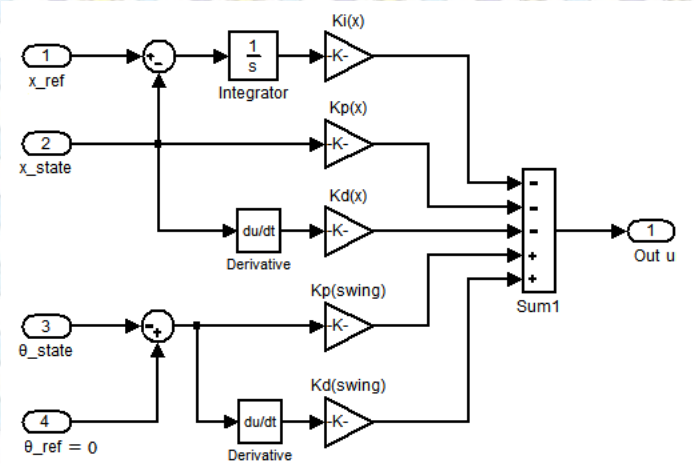
$$u = K_{LQI} . e_{ss} \quad (3.28)$$

Persamaan *close loop* LQR-I menjadi:

$$\dot{X}(t) = (A - BK_{CL}C) X(t) + (E - BK_{CL}) u(t)$$

$$z(t) = HX(t) \quad (3.29)$$

Adapun blok diagram kontroler PD-LQR *type* I dapat dilihat seperti pada Gambar 3.14.



Gambar 3.14. Blok Diagram Kontroler PD-LQR *Type* I (Integrator)

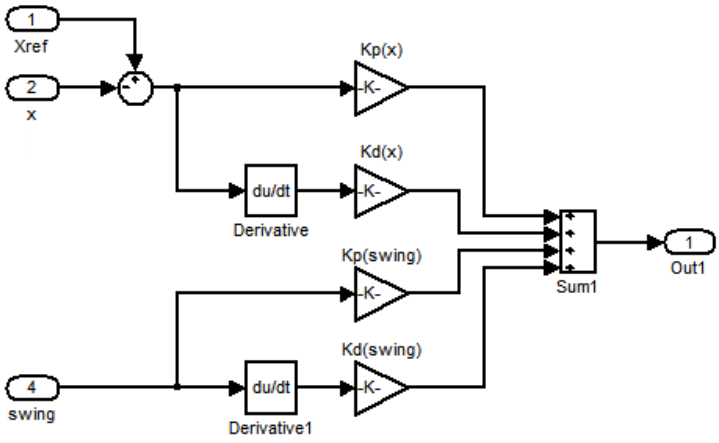
3.7 Pengujian Kontroler

Untuk menguji dan membandingkan performa kontroler PD-LQR optimasi uPSO yang dihasilkan, dipilih beberapa kontroler yang pernah diteliti sebelumnya.

Adapun beberapa kontroler lain yang dipakai sebagai pembandingnya antara lain:

3.7.1 Kontroler PD-*Basic*

Berikut blok simulink kontroler PD *basic* untuk pengaturan *crane* anti ayun berdasarkan Ismail Rokhim [10].



Gambar 3.15. Blok Simulink Kontroler PD *Basic*

dengan besarnya nilai *gain K* adalah:

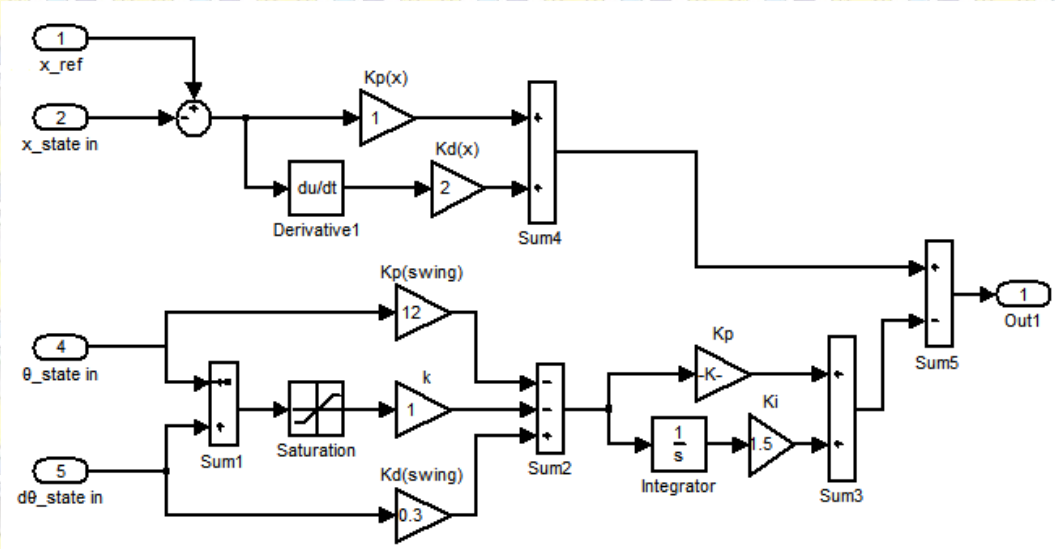
$$K_{dx} = 2\sqrt{K_{px}}$$
$$K_{d\theta} = 2\sqrt{l.M.[K_{p\theta} + (m+M).g]} \quad (3.30)$$

dan sinyal kontrol *u* dirumuskan dengan:

$$u = (K_{px} + K_{dx}s)(x_{ref} - x) + (K_{p\theta} + K_{d\theta}s).\theta \quad (3.31)$$

3.7.2 Kontroler SMC-PI (*Sliding Mode Control–PI*)

Berikut blok simulink kontroler *Sliding Mode Control* dengan kompensator *PI* untuk pengaturan *crane* anti ayun berdasarkan Ismail Rokhim, 2012 [10].



Gambar 3.16. Blok Simulink Kontroler SMC-PI

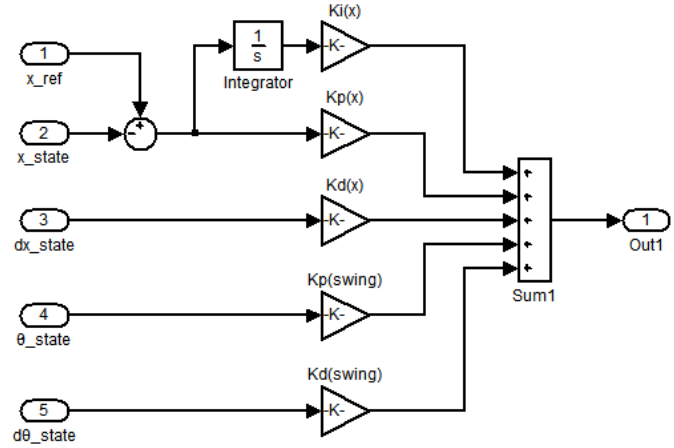
Dengan besarnya nilai *gain K* dan sinyal kontrol *u* dirumuskan dengan:

$$K_{dx} = 2\sqrt{K_{px}}$$
$$K_{p\theta} = (m+M)g$$
$$K_{d\theta} = \lambda.l.M \quad (3.32)$$
$$K_p = 0{,}75$$
$$K_i = 1{,}5$$

$$u_1 = (K_{px} + sK_{dx})(x_{ref} - x_1)$$
$$u_{SMC} = (K_p + \frac{K_i}{s})[-K_{p\theta}x_2 + K_{d\theta}x_4 - k.sat(S,\varepsilon)]$$
$$S = x_4 + \lambda x_2$$
$$u = u_1 - u_{SMC} \quad (3.33)$$

3.7.3 Kontroler PD *State Feedback* dengan Integrator

Kontroler ini berdasarkan penelitian Yong-Seok Kim, dkk [6]. Gambar 3.17 berikut merupakan blok simulink kontroler PD *State Feedback* dengan integrator untuk pengaturan *crane* anti ayun.



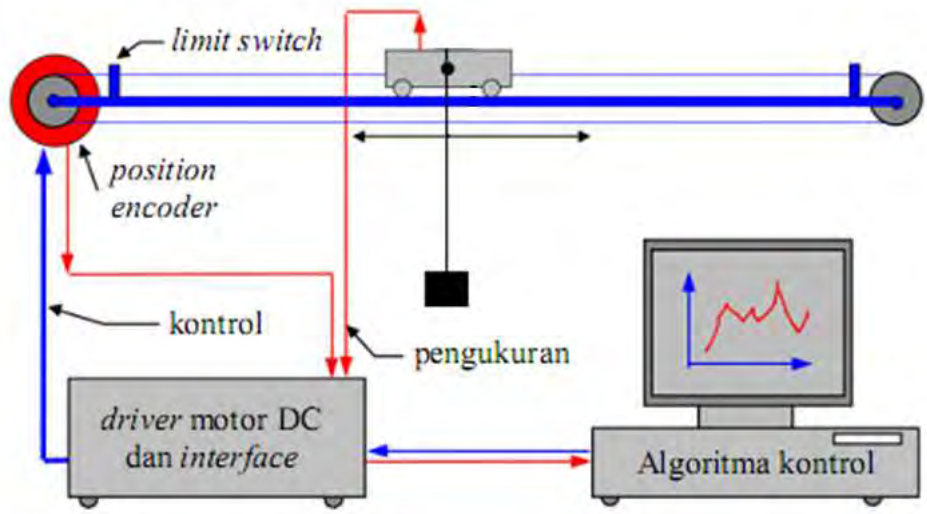
Gambar 3.17. Blok Simulink Kontroler PD *State Feedback* dengan Integrator

Dengan besarnya nilai *gain K* dan sinyal kontrol *u* dirumuskan dengan:

$$K_{px} = 2\xi\sqrt{\frac{l}{g}}K_0 + K_1$$
$$K_{dx} = 2\xi\sqrt{\frac{l}{g}}K_1 + K_2$$
$$K_{p\theta} = 2\xi\sqrt{\frac{l}{g}}(lK_0 + gK_2) + gm$$
$$K_{d\theta} = 2\xi\sqrt{\frac{l}{g}}(lK_1 + gM) \quad (3.34)$$
$$K_{ix} = K_0$$
$$K = [K_{px}\ K_{dx}\ K_{p\theta}\ K_{d\theta}\ K_{ix}]$$
$$u = K.(x - x_{ref}) \quad (3.35)$$

3.8 Implementasi dan Pengujian di Laboratorium

Untuk proses implementasi dan pengujian *real* kontroler yang dirancang, akan digunakan *plant* berupa sistem pendulum kereta (SPK) dari *Feedback Instruments Ltd*., jenis *Digital Pendulum Mechanical Unit* 33-200 [20]. Adapun konsep *plant*-nya dapat dilihat pada Gambar 3.18.



Gambar 3.18. Konsep Implementasi *Plant Crane* Anti Ayun

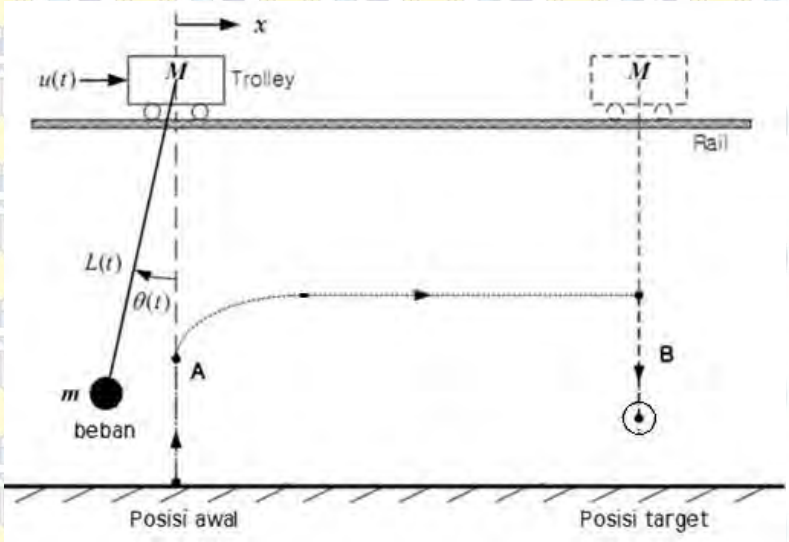
Penerapan sistem kontrol dilakukan pada komputer dengan bantuan *software* Simulink/MATLAB. Komputer dan *plant* terhubung melalui modul "*Digital Pendulum Controller 33-201*" sebagai kontroler antarmuka, serta board akuisisi data (DAQ) sebagai I/O pada komputer. Sinyal kontrol dari komputer keluar melalui *Digital to Analog Converter* (DAC) yang terdapat pada DAQ. *Power amplifier* yang terhubung dengan port keluaran DAQ akan menerima sinyal kontrol yang kemudian dikirim ke motor DC untuk menggerakkan kereta. Sinyal respons dari kereta terbaca oleh encoder posisi dan akan dikirim ke komputer melalui *Analog to Digital Converter* (ADC) pada DAQ. Adapun sudut ayunan beban akan terbaca oleh *angle sensor* yang akan diterjemahkan oleh program sebagai besarnya sudut ayun beban.

Parameter *plant* yang diperhitungkan antara lain massa beban, massa kereta dan panjang tali/tungkai pendulum. Kemudian dengan algoritma uPSO akan dicari nilai matriks *Q* dan matriks *R* yang paling optimal untuk membentuk kontroler PD-

LQR/I dengan besaran *gain K* optimal. *Gain K* ini kemudian akan diterapkan pada kontroler yang akan dipakai untuk mengendalikan *plant* eksperimen di laboratorium.

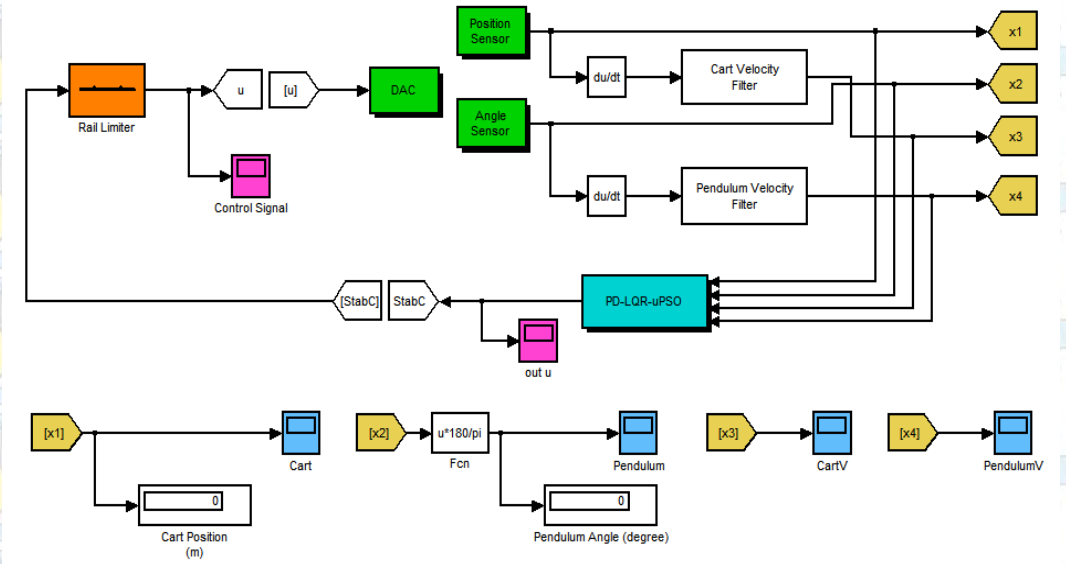
Berikut mekanisme implementasi dan pengujian sistem:

1. Kontroler yang didesain telah disimulasikan terlebih dahulu pada simulator, dan jika dianggap telah memenuhi syarat dapat diterapkan pada *plant* eksperimen di laboratorium. Begitu juga dengan kontroler lain yang sudah pernah diteliti sebelumnya.
2. Pengujian dilakukan dengan menjalankan *crane* dari titik awal ke titik akhir di rel lintasan yang telah ditentukan seperti pada Gambar 3.19.
3. Respon *plant* berupa pergerakan *crane* dan besarnya ayunan beban akan dicatat untuk dibandingkan untuk tiap-tiap kontroler yang diujikan.



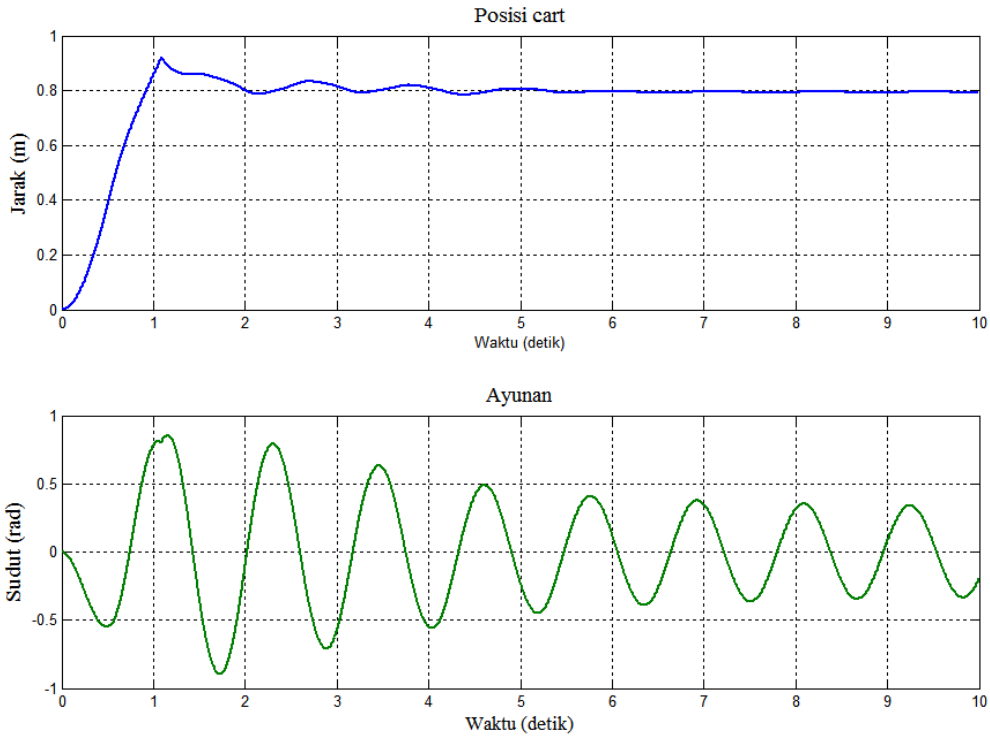
Gambar 3.19. Jalur Lintasan Pengujian *Crane*

Adapun blok diagram simulink untuk implementasi eksperimen kontroler pada *plant* SPK di laboratorium dapat dilihat pada Gambar 3.20 berikut ini.



Gambar 3.20. Blok Simulink Implementasi Kontroler pada *Plant* SPK

Berikut respon *plant* SPK *Digital Pendulum Mechanical* Unit 33-200 saat diberikan sinyal *input* step berupa *u*=10.



Gambar 3.21. Respon *Step* SPK untuk u=10

-----Halaman ini sengaja dikosongkan-----

BAB IV
HASIL DAN PEMBAHASAN

Dalam bab ini akan dipaparkan hasil optimasi proses pemilihan matriks pembobot *Q* dan *R* dengan metode uPSO pada kontroler PD-LQR dibandingakan dengan pemilihan matriks pembobot *Q* dan *R* dengan metode *trial and error* (TEM). Juga akan dijelaskan pengaruh pemilihan jenis matriks pembobot *Q* terhadap hasil optimasi tersebut.

Hasil optimasi ini akan diuji dengan kontroler lainnya yang sudah pernah diteliti sebelumnya, juga akan diuji pengaruh penambahan Integral pada kontroler PD-LQR sehingga membentuk kontroler PD-LQR *type* I.

4.1 Proses *Tuning Q* dan *R* dengan uPSO

Untuk proses *tuning*, program uPSO diatur dengan parameter sebagai berikut:

uPSO Parameter:

- Jumlah *swarm* = 35
- Konstanta Chi = 0,729
- Konstanta Kognitif = 2,05
- Konstanta Sosial = 2,05
- *Neighborhood Radius* = 10
- Iterasi Maksimum = 200

Parameter Matriks Pembobot:

- Nilai Maksimum Q = 100
- Nilai Minimum Q = 0
- Nilai Maksimum R = 100
- Nilai Minimum R = 0,0001

Parameter *Plant*:

- Massa *crane* = 1,12 kg
- Massa beban = 0,12 kg
- Panjang tungkai pendulum = 0,402 m
- Konstanta gravitasi = 9,8 m/s^2
- *Input* referensi = 0,7 m

Untuk matriks *state* hasil linearisasi diperoleh:

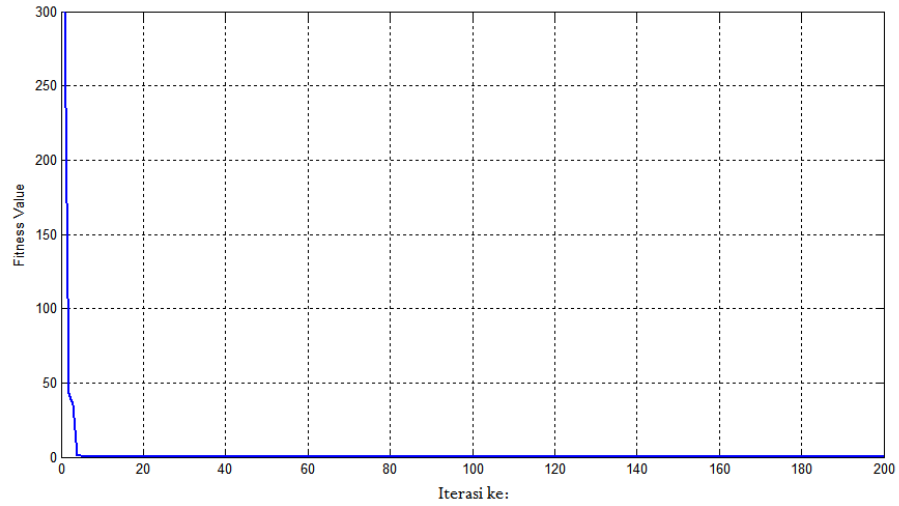
$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0{,}2525 & 0 & 0{,}00013 \\ 0 & -15{,}0421 & 0 & -0{,}0079 \end{bmatrix} \text{ dan } B = \begin{bmatrix} 0 \\ 0 \\ 0{,}8272 \\ -1{,}2396 \end{bmatrix}$$

4.1.1 Proses Optimasi Matriks *Q* Bebas

Berdasarkan hasil eksekusi program optimasi kontroler PD-LQR menggunakan algoritma uPSO dengan pilihan matriks *Q* bebas diperoleh nilai *fitness* terkecil sebesar 0,1437.

Dalam progres menuju pencapaian nilai optimal, terlihat bawah untuk matriks *Q type* bebas, nilai *fitness* mula-mula masih berada diharga 300. Hal ini dikarenakan pada kondisi awal *tuning* untuk matriks *Q* bebas yang dihasilkan masih belum mencapai syarat utama yaitu matriks *Q* harus merupakan matriks semidefinit positif.

Adapun progres pencapaian nilai optimal dari ketiga hasil optimasi ini dapat dilihat pada grafik sesuai Gambar 4.1 berikut ini.



Gambar 4.1. Progres Pencapaian Nilai *Fitness* Matriks *Q* Bebas pada PD-LQR-uPSO

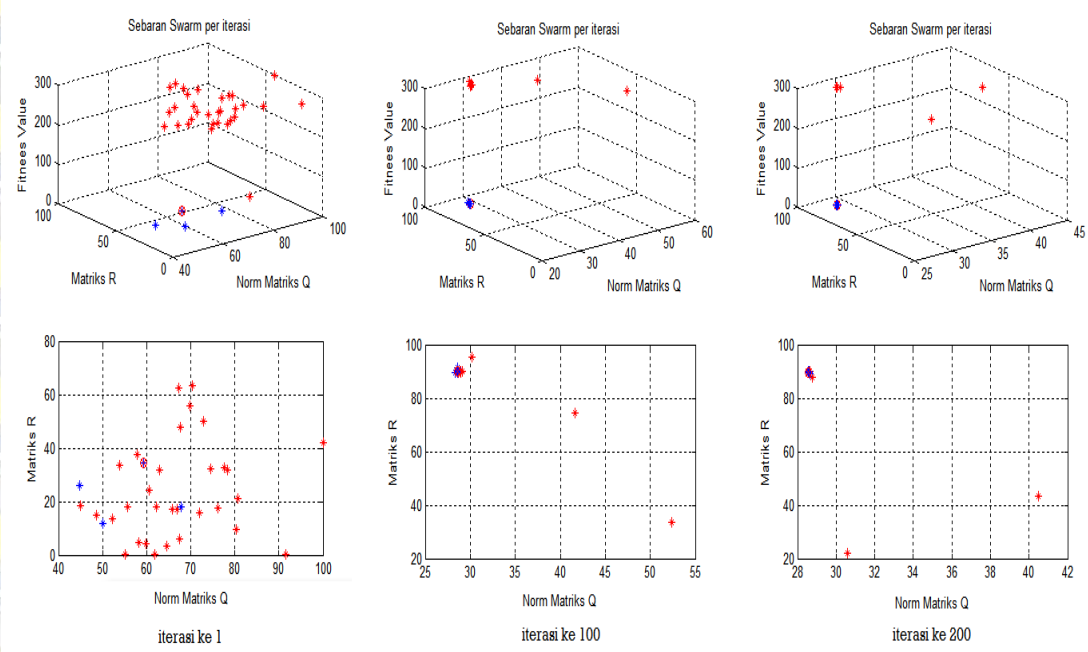
Terlihat bahwa pada iterasi ke-2 dan ke-3 nilai *fitness* turun menjadi 43 dan 34,9, hal ini dikarenakan matriks *Q* telah memenuhi syarat matriks semidefinit positif, namun

sistem belum mencapai batas parameter *output* yang diinginkan (misalnya ayunan beban maksimum pada beban masih melewati batasan 0,0436 rad / 2,5 derajat). Nilai *fitness* ini terus berkurang menuju minimum dimana pada iterasi ke-4 nilai *fitness*-nya mencapai 1,09 yang menandakan bahwa syarat batasan parameter *output* sudah terpenuhi.

Setelah iterasi ke-100 hingga iterasi ke-200 nilai *fitness* terus berkurang dan mulai mencapai nilai minimumnya dikisaran 0,14 yang berarti bahwa matriks *Q* dan *R* telah menuju nilai optimal sesuai batasan-batasan yang kita inginkan.

Dapat pula dilihat proses sebaran *swarm* menuju nilai *Q* dan *R* yang optimal, dimana terlihat pada saat iterasi ke-1 masih didominasi oleh *swarm* berwarna merah yang menandakan *swarm* belum memenuhi syarat batasan kriteria parameter *output* sistem, dan saat iterasi ke-100 tersisa 5 *swarm* yang belum memenuhi syarat batasan kriteria *output* yang diinginkan, dan tampak bahwa *swarm* yang memenuhi syarat (*swarm* dengan warna biru) berkumpul pada satu daerah yang sama. Saat iterasi ke-200 hanya tersisa 4 *swarm* yang belum memenuhi kriteria.

Adapun progres sebaran *swarm* menuju pencapaian nilai *Q* dan *R* yang optimal dapat dilihat pada ilustrasi Gambar 4.2 berikut ini.



Gambar 4.2. Sebaran *Swarm R* dan *Q* Bebas per Iterasi

Sehingga diperoleh besarnya matriks Q dan R yang merupakan matriks pembobot optimal untuk Q bebas, sebesar:

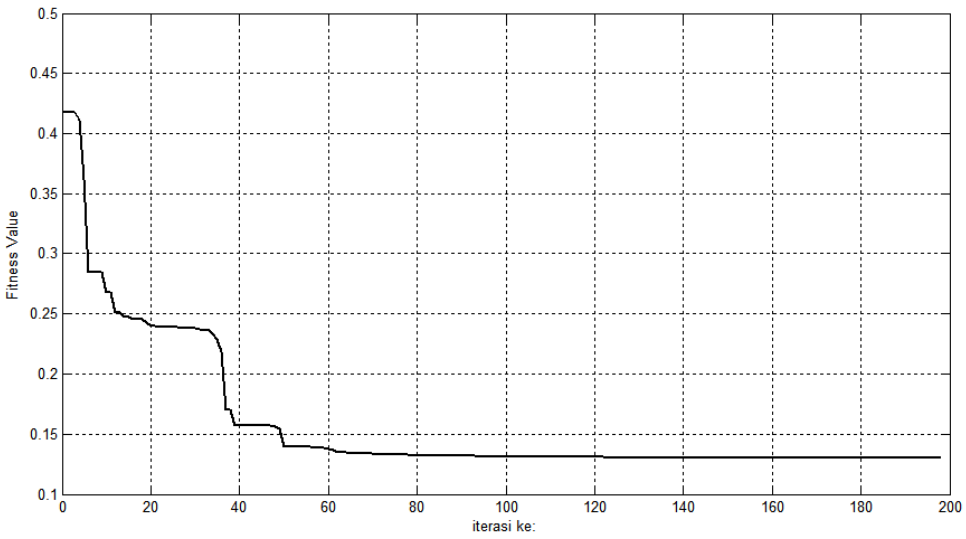
$$Q_{opt} = \begin{bmatrix} 0{,}1110 & 0 & 0{,}0011 & 0 \\ 0 & 56{,}0047 & 3{,}4592 & 40{,}9377 \\ 0{,}0011 & 3{,}4592 & 0{,}3938 & 0 \\ 0 & 40{,}9377 & 0 & 94{,}2459 \end{bmatrix} \quad \text{dan} \quad R = [0{,}0036]$$

4.1.2 Proses Optimasi Matriks *Q* Diagonal

Berdasarkan hasil *tuning* PD-LQR dengan metode optimasi uPSO untuk pemilihan jenis matriks Q diagonal diperoleh nilai *fitness* sebesar 0,1303.

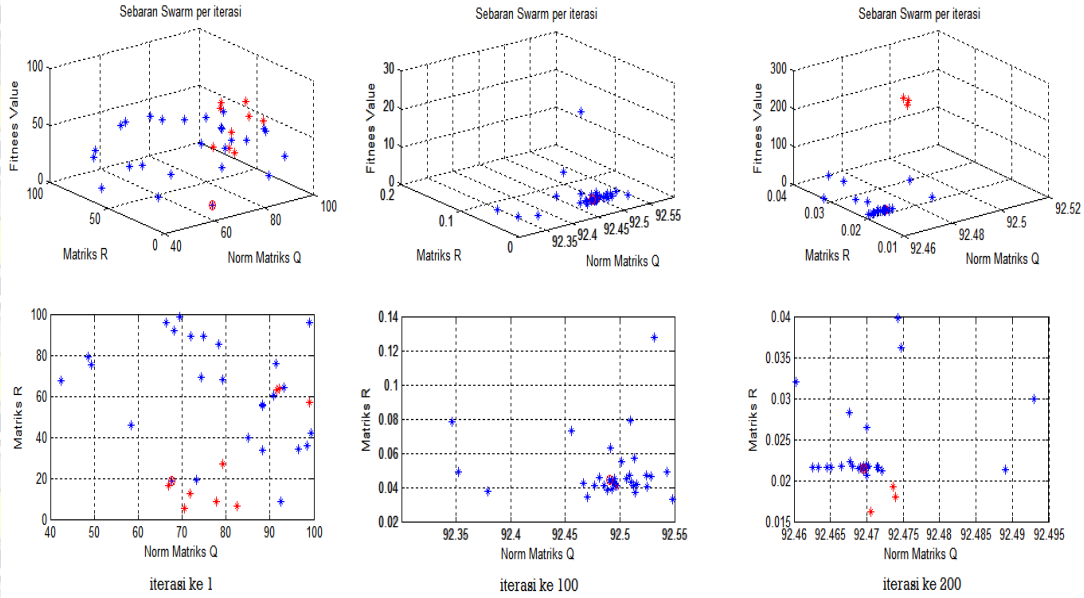
Untuk matriks Q tipe diagonal, progres optimasinya terlihat lebih cepat mencapai nilai minimum. Pada matriks Q *type* diagonal, nilai awal matriks Q yang dibangkitkan secara acak telah memenuhi syarat utama berupa matriks Q semi-definit positif, oleh karenanya pada saat iterasi ke-1 nilai *fitness* sudah mencapai 0,4175. Nilai *fitness* ini akan terus berkurang dimana pada saat iterasi ke-5 nilai *fitness* mencapai 0,36. Pada iterasi ke-28 nilai *fitness* mencapai 0,17 dan pada iterasi ke-40 turun menjadi 0,154. Pada iterasi ke 50 nilai *fitness* 0,130 dan mulai stabil sejak iterasi ke-129 hingga iterasi ke-200 pada nilai 0,130.

Untuk lebih jelas, grafik progres pencapaian nilai optimal dapat dilihat pada Gambar 4.3 di bawah ini.



Gambar 4.3. Progres Pencapaian Nilai *Fitness* Matriks Q Diagonal PD-LQR-uPSO

Selain itu dapat dilihat pula proses sebaran *swarm* menuju daerah *Q* dan *R* yang optimal seperti Gambar 4.4 dibawah ini.



Gambar 4.4. Sebaran *Swarm R* dan *Q* Diagonal per Iterasi

Dimana terlihat pada saat iterasi ke-1 ada sekitar 8 *swarm* yang belum memenuhi syarat batasan kriteria *output* sistem, dan saat iterasi ke-100 hingga iterasi ke-200, semua partikel tampak berkumpul pada daerah yang sama dan telah memenuhi syarat batasan kriteria *output* yang diberikan semisal ayunan yang tidak lebih dari 0,0436 rad (2,5 derajat).

Sehingga untuk *type* matriks *Q* diagonal, akan dipilih matriks *Q* dan *R* optimal yang merupakan nilai dengan *fitness* terkecil yaitu:

$$Q_{opt} = \begin{bmatrix} 0{,}0393 & 0 & 0 & 0 \\ 0 & 0{,}0006 & 0 & 0 \\ 0 & 0 & 0{,}1560 & 0 \\ 0 & 0 & 0 & 92{,}4696 \end{bmatrix} \quad \text{dan} \quad R = [0{,}0215]$$

4.2 Kontroler PD-LQR dengan uPSO untuk Matriks *Q* Bebas

Untuk proses optimasi kontroler PD-LQR dengan uPSO, dengan pemilihan jenis matriks Q bebas, dipilih matriks *Q* dan *R* yang menghasilkan nilai *fitness* paling kecil, dimana diperoleh untuk tiap-tiap matriks *Q* dan *R* sebagai berikut:

$$Q_{opt} = \begin{bmatrix} 0{,}1110 & 0 & 0{,}0011 & 0 \\ 0 & 56{,}0047 & 3{,}4592 & 40{,}9377 \\ 0{,}0011 & 3{,}4592 & 0{,}3938 & 0 \\ 0 & 40{,}9377 & 0 & 94{,}2459 \end{bmatrix} \quad \text{dan} \quad R_{opt} = [0{,}0036]$$

Selanjutnya, untuk matriks *state* A dan B berikut

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0{,}2525 & 0 & 0{,}00013 \\ 0 & -15{,}0421 & 0 & -0{,}0079 \end{bmatrix} \text{ dan } B = \begin{bmatrix} 0 \\ 0 \\ 0{,}8272 \\ -1{,}2396 \end{bmatrix}$$

Dengan menyelesaikan Persamaan Ricatti (3.7), menggunakan perintah di Matlab K=LQR(A,B,Q_{opt},R_{opt}), diperoleh besarnya *gain K* optimal sebesar:

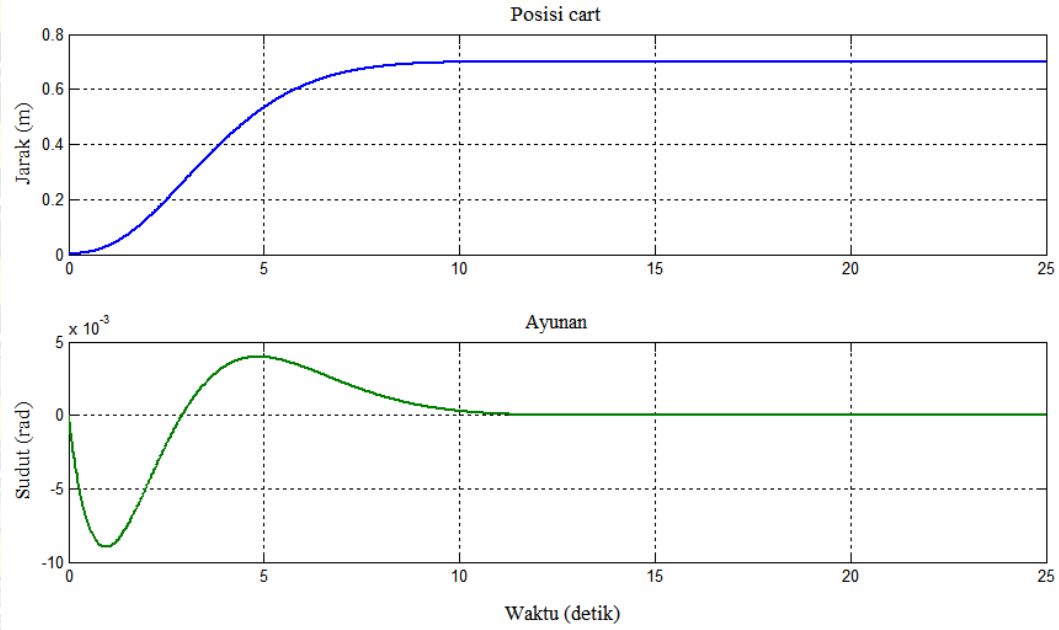
$K_{opt} = [5{,}5844 \quad -271{,}5268 \quad 20{,}8321 \quad -155{,}1819]$

dimana untuk penerapan pada kontroler PD digunakan :

$K_{px} = 5{,}5844 \qquad K_{dx} = 20{,}8321$

$K_{p\theta} = -271{,}5268 \qquad K_{d\theta} = -155{,}1819$

Sehingga jika nilai *gain K* ini selanjutnya disimulasikan dengan simulink dan akan menghasilkan *output* sistem *crane* seperti Gambar 4.5 dibawah ini.



Gambar 4.5. *Output* Sistem untuk Kontroler PD-LQR-UPSO dengan Matriks *Q* Bebas

4.3 Kontroler PD-LQR dengan uPSO untuk Matriks *Q* Diagonal

Untuk proses optimasi kontroler PD-LQR dengan uPSO, dengan pemilihan jenis matriks *Q* bebas, dipilih matriks *Q* dan *R* yang menghasilkan nilai *fitness* paling kecil, dimana dipilih untuk tiap-tiap matriks *Q* dan *R* sebagai berikut:

$$Q_{opt} = \begin{bmatrix} 0{,}0393 & 0 & 0 & 0 \\ 0 & 0{,}0006 & 0 & 0 \\ 0 & 0 & 0{,}1560 & 0 \\ 0 & 0 & 0 & 92{,}4696 \end{bmatrix} \quad \text{dan} \quad R = [0{,}0215]$$

Selanjutnya, untuk matriks *state* A dan B berikut

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0{,}2525 & 0 & 0{,}00013 \\ 0 & -15{,}0421 & 0 & -0{,}0079 \end{bmatrix} \text{ dan } B = \begin{bmatrix} 0 \\ 0 \\ 0{,}8272 \\ -1{,}2396 \end{bmatrix}$$

Dengan menyelesaikan persamaan Ricatti (3.7), menggunakan perintah di Matlab K=LQR(A,B,Q_{opt},R_{opt}), diperoleh besarnya *gain K* optimal sebesar:

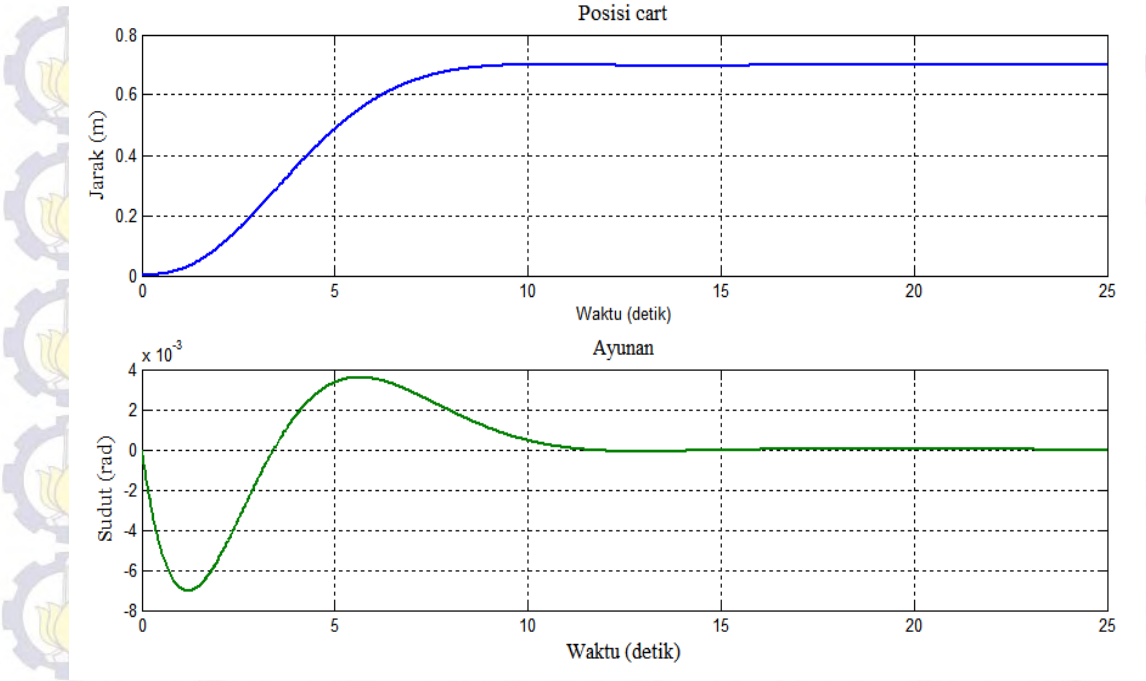
$$K = [1{,}3523 \;\; -72{,}7614 \;\; 5{,}5375 \;\; -64{,}3140]$$

dimana untuk penerapan pada kontroler PD digunakan

$K_{px} = 1{,}3523$ $\qquad$ $K_{dx} = 5{,}5375$

$K_{p\theta} = -72{,}7614$ $\qquad$ $K_{d\theta} = -64{,}3140$

Sehingga jika nilai *gain K* ini selanjutnya disimulasikan dengan simulink dan akan diperoleh *output* sistem seperti Gambar 4.6.



Gambar 4.6. *Output* Sistem untuk Kontroler PD-LQR UPSO dengan *Q* Diagonal

4.4 Kontroler PD-LQR-TEM dengan Matriks Q = $H^T H$

Dari persamaan *state space*, diperoleh matriks H

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Berdasarkan persamaan $Q=H^T H$, dan matriks R dengan metode TEM, diperoleh matriks pembobot:

$$Q_{opt} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{dan} \quad R = [2{,}090]$$

Selanjutnya, untuk matriks *state* A dan B berikut

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0{,}2525 & 0 & 0{,}00013 \\ 0 & -15{,}0421 & 0 & -0{,}0079 \end{bmatrix} \text{ dan } B = \begin{bmatrix} 0 \\ 0 \\ 0{,}8272 \\ -1{,}2396 \end{bmatrix}$$

Dengan menyelesaikan Persamaan Ricatti (3.7), menggunakan perintah di Matlab K=LQR(A,B,Q_{opt},R_{opt}) , diperoleh besarnya *gain K* sebesar:

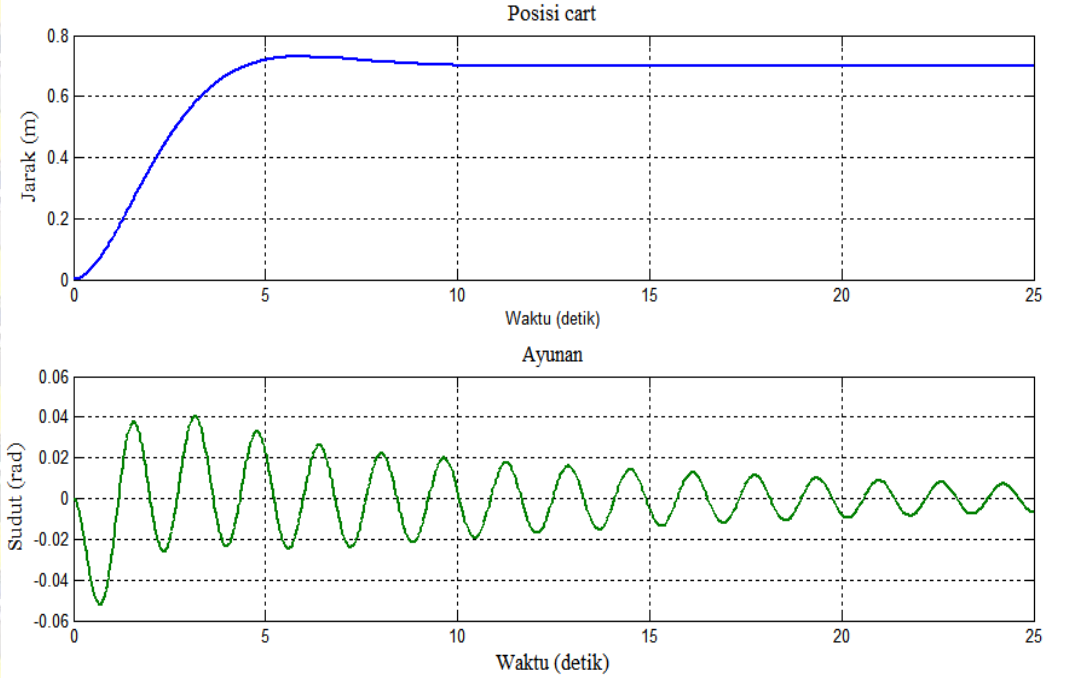
$K = [0{,}6917 \quad -0{,}1322 \quad 1{,}3169 \quad -0{,}0784]$

dimana untuk penerapan pada kontroler PD digunakan

$K_{px} = 0{,}6917 \qquad K_{dx} = 1{,}3169$

$K_{p\theta} = -0{,}1322 \qquad K_{d\theta} = -0{,}0784$

Nilai *gain K* ini selanjutnya disimulasikan dengan simulink dan menghasilkan *output* sistem *crane* seperti Gambar 4.7.



Gambar 4.7. *Output* Sistem untuk Kontroler PD-LQR TEM dengan Matriks $Q = H^T H$

4.5 Kontroler PD-LQR-TEM dengan Matriks Q Diagonal

Jika dipilih $t_{s1} = t_{s3} = 1{,}5$ detik dan $t_{s3} = t_{s4} = 2$ detik,

$x_{1max} = 0{,}02$ m = 2 cm (e_{ss} maksimum)

$x_{2max} = 0{,}0436$ rad = 2,5 derajat (maksimum ayunan beban)

$x_{3max} = 0{,}25$ m/s = 25 cm/s (kecepatan *crane* maksimum)

$x_{4max} = 0{,}15$ rad/s (maksimum kecepatan ayunan beban)

$u_{imax} = 1$

$\rho = 0 < \rho \leq 100$, dengan $\Delta\rho = 0{,}01$

Maka untuk matriks *Q type* diagonal diperoleh nilai matriks bobot *Q* dan *R* berdasarkan metode TEM sebesar:

$$Q_o = \begin{bmatrix} 0{,}1667 & 0 & 0 & 0 \\ 0 & 263{,}025 & 0 & 0 \\ 0 & 0 & 0{,}0008 & 0 \\ 0 & 0 & 0 & 22{,}2222 \end{bmatrix} \quad \text{dan} \quad R = [0{,}0340]$$

untuk matriks *state* A dan B berikut

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0{,}2525 & 0 & 0{,}00013 \\ 0 & -15{,}0421 & 0 & -0{,}0079 \end{bmatrix} \text{ dan } B = \begin{bmatrix} 0 \\ 0 \\ 0{,}8272 \\ -1{,}2396 \end{bmatrix}$$

Dengan menyelesaikan persamaan Ricatti (3.7) menggunakan perintah di Matlab K=LQR(A,B,Q_{opt},R_{opt}) , diperoleh besarnya *gain K* sebesar:

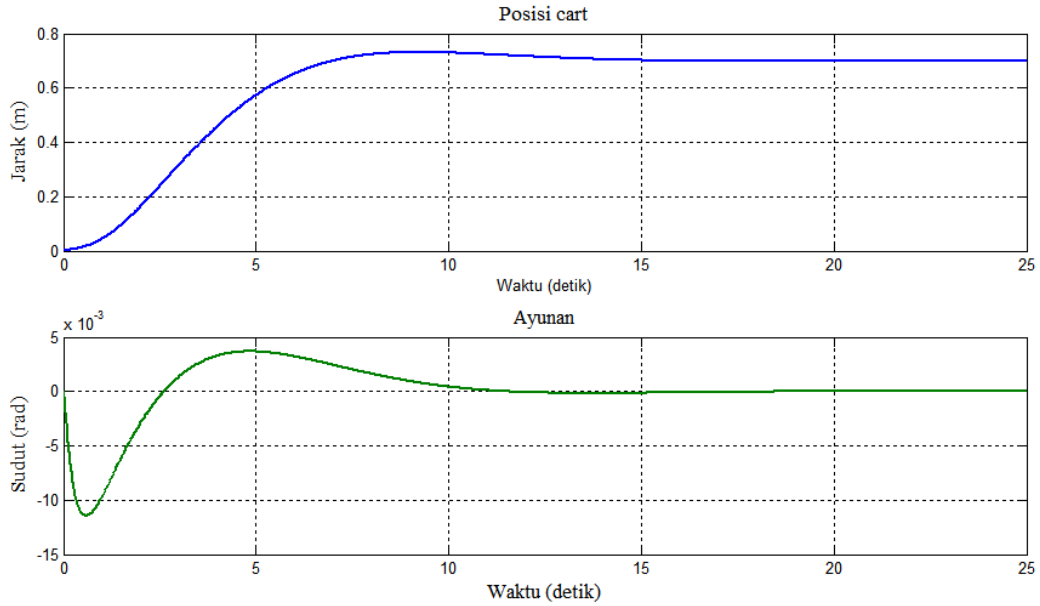
$K = [2{,}2143 \quad -94{,}5028 \quad 6{,}9440 \quad -24{,}4022]$

dimana untuk penerapan pada kontroler PD digunakan

$K_{px} = 2{,}2143 \qquad K_{dx} = 6{,}9440$

$K_{p\theta} = -94{,}5028 \qquad K_{p\theta} = -24{,}4022$

Nilai *gain K* ini selanjutnya disimulasikan dan dihasilkan *output* berikut ini:



Gambar 4.8. *Output* Sistem untuk Kontroler PD-LQR TEM dengan *Q* Diagonal

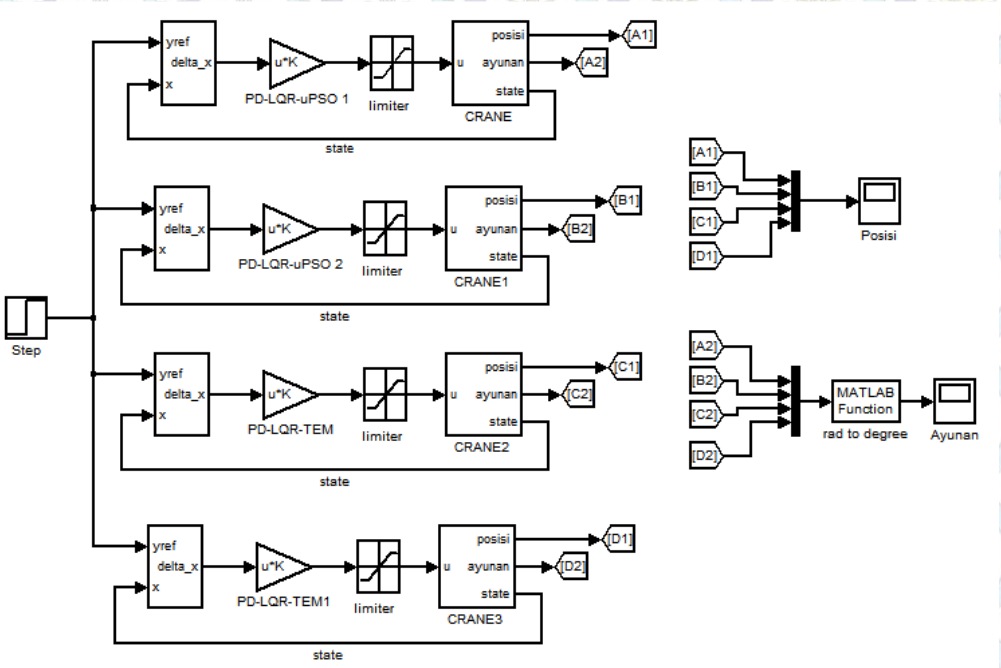
4.6 Pengujian dan Perbandingan Sistem Kontroler PD-LQR

Untuk mengetahui metode dalam pemilihan matriks pembobot *Q* dan *R* yang menghasilkan *output* yang paling optimal, dibuat suatu blok simulink dimana tiap-tiap kontroler PD-LQR dengan metode yang berbeda diuji keluarannya masing-masing. Adapun perbandingan *gain K* untuk tiap-tiap kontroler disajikan dalam Tabel 4.1.

Tabel 4.1. Nilai *Gain K* untuk Tiap Kontroler PD-LQR Berdasar *Type* Matriks *Q*

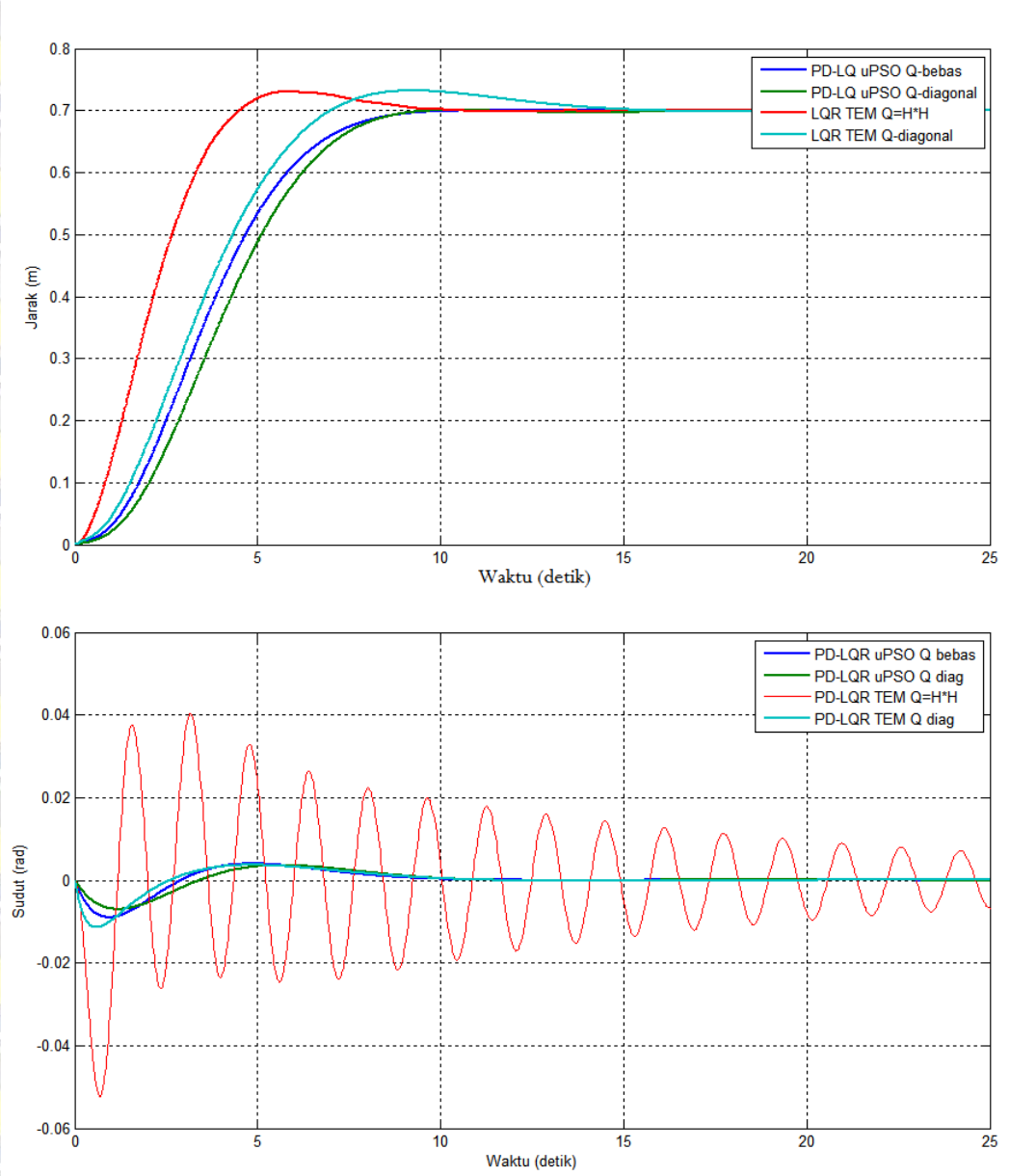
Konstanta *Gain*	Kontroler PD-LQR-uPSO		Kontroler PD-LQR-TEM	
	Matriks Q *type* bebas	Matriks Q *type* diagonal	Matriks Q *type* H^TH	Matriks Q *type* diagonal
Kpx (posisi)	5,5844	1,3523	0,6917	2,2143
Kdx (kecepatan)	20,8321	5,5375	1,3169	6,9440
Kp_θ (ayunan)	-271,5268	-72,7614	-0,1322	-94,5028
Kd_θ (kecepatan sudut)	-155,1819	-63,3140	-0,0784	-24,4022

Adapun blok simulink pengujian kontroler PD-LQR seperti tampak pada Gambar 4.9 berikut.



Gambar 4.9. Blok Simulink Pengujian Kontroler PD-LQR Berdasar Matriks *Q*

Adapun tampilan *output plant* dengan kontroler PD-LQR untuk tiap-tiap metode optimasi dan pemilihan jenis matriks *Q* yang berbeda dapat dilihat pada Gambar 4.10 berikut.



Gambar 4.10. *Output* Sistem untuk Kontroler PD-LQR Berdasar Pemilihan Matriks *Q*

Untuk perbandingan performansi sistem tiap-tiap pemilihan jenis matriks *Q* dan metode optimasinya dapat dilihat pada Tabel 4.2.

Tabel 4.2. Perbandingan Performansi Kontroler PD-LQR Berdasar Matriks *Q*

Parameter *Output*	Kontroler PD-LQR-uPSO		Kontroler PD-LQR-TEM	
	Matriks *Q* *type* bebas	Matriks *Q* *type* diagonal	Matriks *Q* *type* H^TH	Matriks *Q* *type* diagonal
Settling Time (detik)	7,8	8,0	7,4	11,5
Steady state error (m)	3,3398e-7	6,4231e-6	0,0009	0,0001
Persen *overshoot* (%)	0	0,0033	**4,2689**	**4,4926**
Maksimum ayunan (rad)	0,0091	0,0071	**0,0559**	0,0115
Performance Index	0,2010	0,0782	0,9331	0,2551
Fitnees values	0,1444	0,1305	54,4687	17,0464

Dari hasil tabel perbandingan terlihat bahwa kontroler PD-LQR metode optimasi uPSO dengan pemilihan matriks *Q* diagonal menghasilkan nilai *fitness* terkecil dibanding dengan kontroler PD-LQR dengan metode dan jenis pemilihan matriks *Q* lainnya.

Kontroler PD-LQR dengan matriks Q diagonal juga menghasilkan kontroler dengan indeks performansi terkecil dibanding jenis pemilihan kontroler lainnya.

Sedangkan untuk kontroler PD-LQR dengan metode TEM, terlihat bahwa kedua jenis kontroler TEM ini belum memenuhi persyaratan awal perancangan dimana untuk parameter persentase *overshoot* kontroler PD-LQR dengan TEM menghasilkan *overshoot* melebihi dari syarat batas 4% yang diinginkan.

4.7 Pengujian dan Perbandingan dengan Kontroler Lain

Pada bagian ini akan diuji dan dibandingkan kontroler PD-LQR optimasi dengan uPSO matriks *Q* diagonal dengan beberapa kontroler lain yang sudah pernah diteliti sebelumnya. Kontroler PD-LQR dengan uPSO *type* matriks *Q* diagonal dipilih karena merupakan kontroler PD-LQR dengan hasil paling optimal dibanding kontroler PD-LQR lainnya.

Sebagai pembanding dipilih hasil penelitian Ismail Rokhim [10]. Adapun kontroler yang dipakai sebagai pembanding adalah:

4.7.1 Kontroler PD-*Basic*

Dengan menyelesaikan Persamaan (3.30) dan (3.31), jika dipilih:

$K_{px} = 1$, maka

$K_{dx} = 2\sqrt{K_{px}} = 2\sqrt{1} = 2$

dan

$K_{p\theta} = 10$, maka

$$K_{dx} = 2\sqrt{0{,}442 * 1 * [10 + (1{,}12 + 0{,}12) * 9{,}8]}$$
$$= 2\sqrt{0{,}442 * 22{,}152}$$
$$= 2\sqrt{9{,}95} = 2 * 3{,}155$$
$$= 6{,}31$$

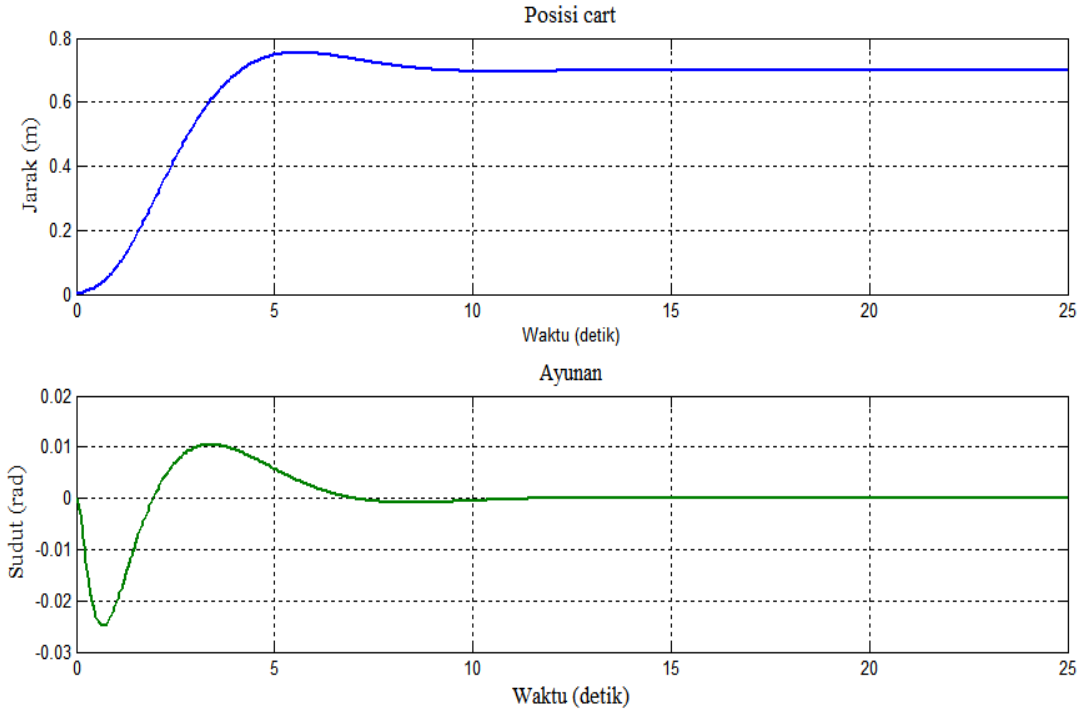
Sehingga diperoleh *gain K* sebesar:

$K = [1 \quad 10 \quad 2 \quad 6{,}31]$

dan besar sinyal kontrol *u* didapat:

$u = (1 + 2s)(x_{ref} - x) + (10 + 6{,}31s).\theta$

Selanjutnya disimulasikan dengan simulink dan akan menghasilkan *output* sistem *crane* seperti tampak pada Gambar 4.11. Berikut tampilan *output* sistem *crane* dengan kontroler PD-*Basic*.



Gambar 4.11. *Output Crane* untuk Jenis Kontroler PD-*Basic*

4.7.2 Kontroler SMC

Dengan menyelesaikan Persamaan (3.32) dan (3.33), jika dipilih

$K_{px} = 1$

$K_{dx} = 2\sqrt{K_{px}} = 2$

$K_{p\theta} = (0{,}12 + 1{,}12) * 9{,}8 = 12{,}15$

$K_{d\theta} = 1/1{,}2369 = 0{,}802$

Sehingga diperoleh *gain K* sebesar:

$K = [1 \quad -12{,}15 \quad 2 \quad 0{,}802]$

Untuk besarnya sinyal kontrol *u*, diperoleh:

$u = u_1 - u_{SMC}$

sinyal kontrol posisi :

$u_1 = (K_{px} + sK_{dx})(x_{ref} - x_1)$

$u_1 = (1 + 2s)(x_{ref} - x)$

sinyal kontrol ayunan :

$u_{SMC} = [-K_{p\theta} x_2 + K_{d\theta} x_4 - k.sat(S, \varepsilon)]$

$S = x_4 + \lambda x_2 = x_4 + 1.x_2$

$u_{SMC} = [-12{,}15x_2 + 0{,}802x_4 - 1.sat((x_4 + x_2), \varepsilon)]$

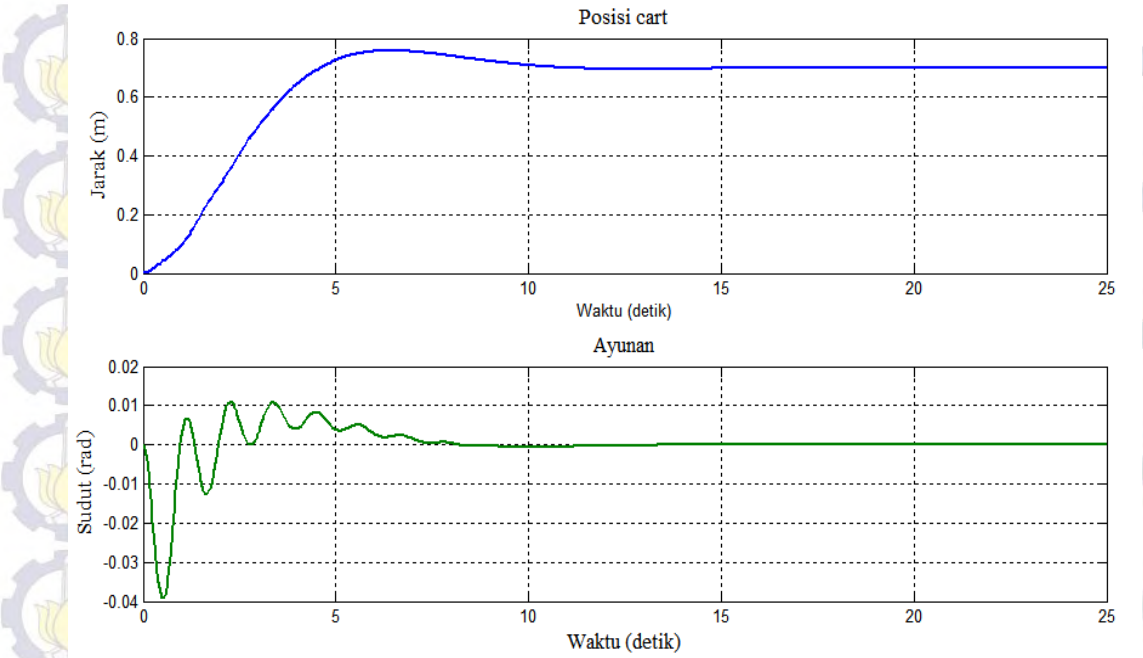
$u_{SMC} = [-12{,}15\theta + 0{,}802\dot{\theta} - 1.sat((\dot{\theta} + \theta), \varepsilon)]$

sehingga diperoleh :

$u = u_1 - u_{SMC}$

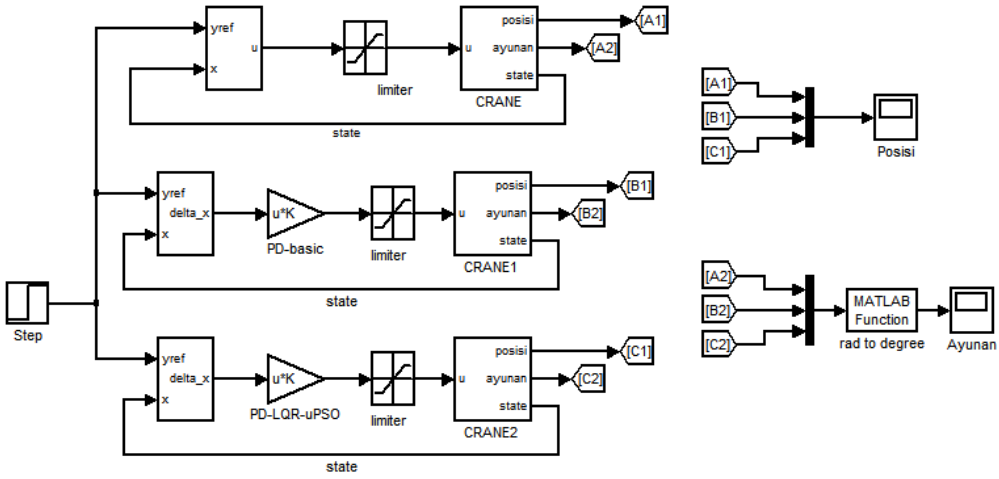
$u = [(1 + 2s)(x_{ref} - x)] - [-12{,}15.\theta + 0{,}802.\dot{\theta} - 1.sat((\dot{\theta} + \theta), \varepsilon)]$

Dari hasil ini selanjutnya disimulasikan dengan simulink, dan dihasilkan *output* sistem *crane* seperti Gambar 4.12.



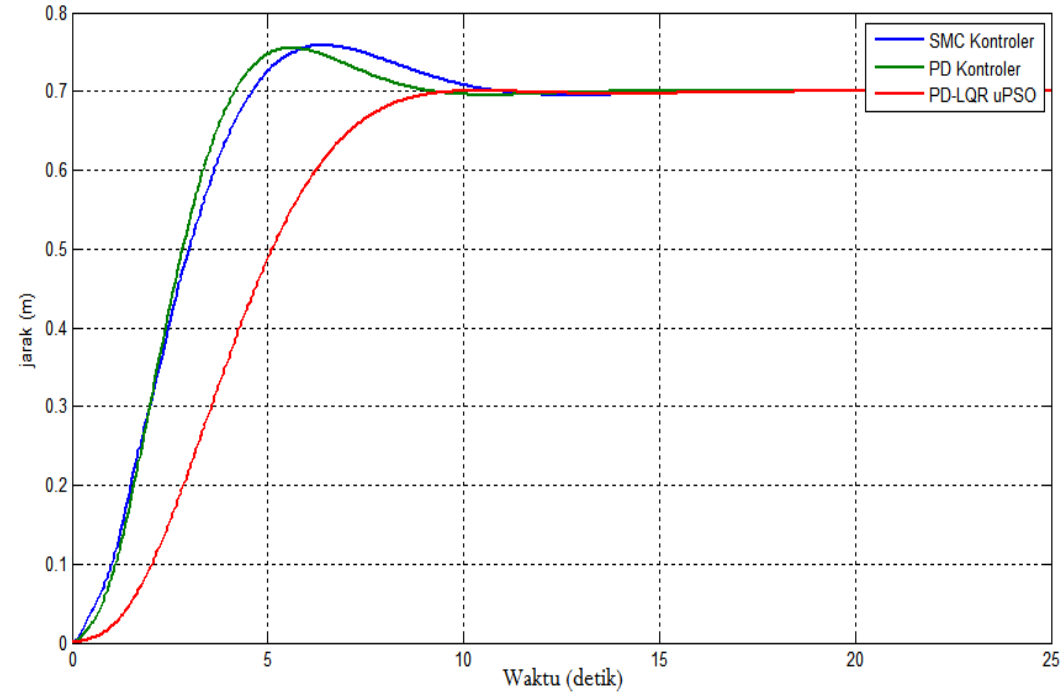
Gambar 4.12. *Output Crane* untuk Jenis Kontroler SMC

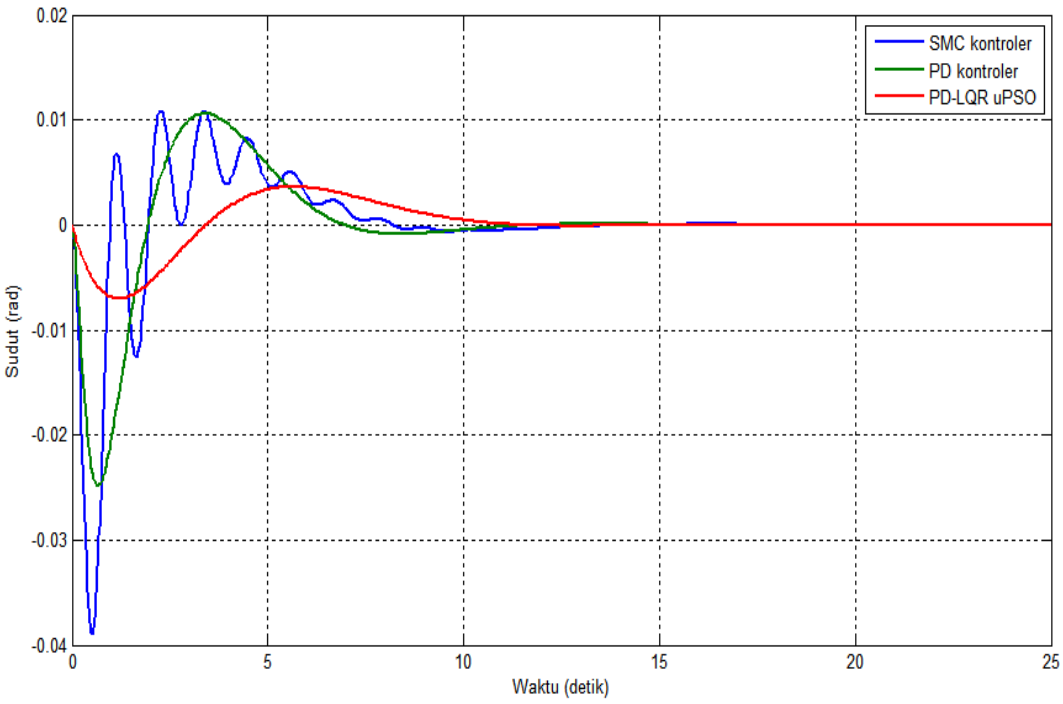
Pengujian dilakukan dengan mensimulasi tiap-tiap kontroler anti-ayun yang diuji dalam suatu blok simulink untuk dibandingkan keluaran *output* kontroler masing-masing. Adapun blok simulinknya seperti Gambar 4.13 berikut ini.



Gambar 4.13. Blok Simulink Pengujian Beberapa Jenis Kontroler Anti Ayun

Berikut tampilan *output plant* dengan kontroler PD-LQR untuk tiap-tiap metode optimasi dan pemilihan jenis matriks *Q* yang berbeda:





Gambar 4.14. *Output Crane* untuk Beberapa Jenis Kontroler yang Diuji

Hasil pengujian perbandingan karakteristik respon *output* dari tiap-tiap kontroler yang diujikan ini ditampilkan dalam bentuk Tabel 4.3.

Tabel 4.3. Perbandingan Karakteristik Respon Tiap Kontroler

Karakteristik respon	Kontroler PD	Kontroler SMC-PI	Kontroler PD-LQR-uPSO
Settling Time (Ts) (detik)	7,6499	9,1	7,9719
Steady state error (e_{ss}) (m)	8,4794e-7	1,5467e-5	2,7621e-6
Persen *overshoot* (%OS)	**7,8286**	**8,3224**	0,0765
Maksimum ayunan (rad)	0,025	0,0392	0,0071
Integral Square Error	0,9576	0,9387	1,5466
RMS energy drive	0,916	0,1054	0,0734

Dari hasil tabel perbandingan terlihat bahwa kontroler PD-LQR metode optimasi uPSO secara simulasi memiliki parameter *output* yang lebih baik dari kontroler jenis lain yang pernah diteliti. Kontroler PD-LQR optimasi uPSO ini lebih baik pada parameter *settling time*, *steady state error*, persentase *overshoot*, dan juga pemakaian energi rata-rata.

Namun untuk parameter indeks performansi dengan memakai perhitungan *Integral Square Error* (ISE) terlihat bahwa kontroler SMC memiliki jumlah *error* kuadrat yang lebih kecil dibanding kontroler lainnya.

4.8 Kontroler PD-LQR *Type* Integral (PD-LQRI) Optimasi UPSO

Untuk kontroler PD-LQR *type* I (PD-LQRI) dengan optimasi algoritma uPSO dikarenakan kontroler ini merupakan pengembangan dari kontroler PD-LQR dasar, maka dipilih nilai *Q* optimal yang membentuk PD-LQR optimal dalam hal ini dipilih matriks *Q* optimal bentuk diagonal yaitu:

$$Q_{opt} = \begin{bmatrix} 0{,}0393 & 0 & 0 & 0 \\ 0 & 0{,}0006 & 0 & 0 \\ 0 & 0 & 0{,}1560 & 0 \\ 0 & 0 & 0 & 92{,}4696 \end{bmatrix} \quad \text{dan} \quad R = [0{,}0215]$$

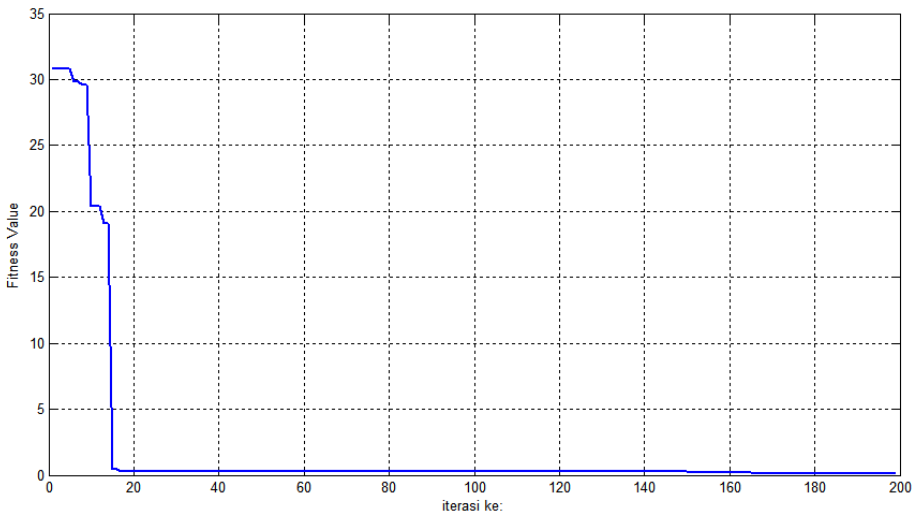
Selanjutnya akan dicari nilai *Q augmented* dengan:

$$Q_a = \begin{bmatrix} Q_{opt} & 0 \\ 0 & q_i \end{bmatrix}$$

Dengan nilai q_i dihasilkan dari proses *tuning* dengan uPSO.

Pada iterasi awal, nilai mula-mula q_i yang dibangkitkan secara acak, terlihat bahwa matriks Q_a yang terbentuk telah memenuhi syarat utama berupa matriks Q_a semidefinit positif, namun *output* yang dihasilkan masih belum memenuhi kriteria batasan yang ditentukan.

Adapun grafik yang menggambarkan progres pencapaian nilai *fitness* optimal dari kontroler PD-LQRI bisa dilihat pada Gambar 4.15 berikut.

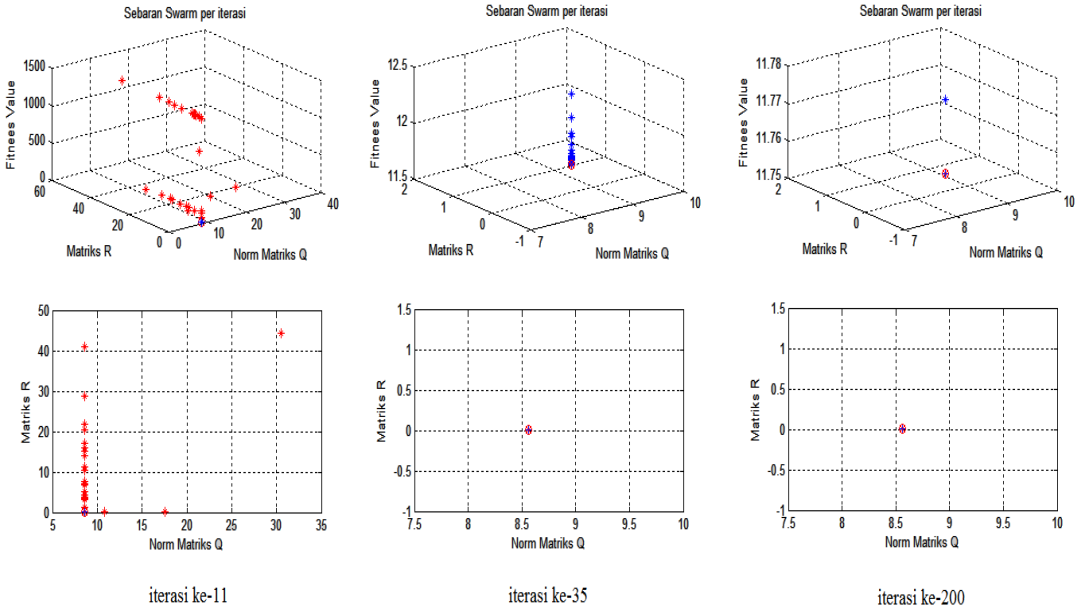


Gambar 4.15. Progres Pencapaian Nilai *Fitness* PD-LQRI dengan Optimasi UPSO

Pada saat iterasi ke-1, nilai *fitness* mula-mula adalah 30,81. Pada iterasi ke-15 nilai *fitness* turun menjadi 0,47 dan terus berkurang secara bertahap hingga pada iterasi ke-150 nilai *fitness* mencapai 0,2 dan turun menjadi 0,1467 pada saat iterasi ke-200.

Untuk proses sebaran *swarm* menuju daerah Q_a dan R yang optimal, pada PD-LQRI terlihat sebaran *swarm* fokus pada satu lokasi, dimana terlihat pada saat iterasi ke-11 ada sekitar 18 *swarm* yang belum memenuhi syarat batasan kriteria *output* sistem, dan saat iterasi ke-65 hingga iterasi ke-200, semua partikel tampak berkumpul pada daerah yang

sama dan telah memenuhi syarat batasan kriteria *output* yang diberikan semisal ayunan yang tidak lebih dari 0,0436 rad (2,5 derajat). Gambar 4.16 mengilustrasikan proses sebaran *swarm* dalam mencari daerah optimal.



Gambar 4.16. Sebaran *Swarm* Matriks Q_a dan R untuk PD-LQRI Optimasi UPSO

Sehingga untuk kontroler PD-LQR *type* integral dengan optimasi uPSO, didapatkan matriks Q_a optimal dan R optimal sebagai berikut:

$$Q_a = \begin{bmatrix} 0{,}0393 & 0 & 0 & 0 & 0 \\ 0 & 0{,}0006 & 0 & 0 & 0 \\ 0 & 0 & 0{,}7727 & 0 & 0 \\ 0 & 0 & 0 & 92{,}469 & 0 \\ 0 & 0 & 0 & 0 & 0{,}0068 \end{bmatrix} \text{ dan } R_a = \begin{bmatrix} 0{,}0129 \end{bmatrix}$$

Untuk matriks *state* berupa

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0{,}2525 & 0 & 0{,}00013 \\ 0 & -15{,}0421 & 0 & -0{,}0079 \end{bmatrix} \text{ dan } B = \begin{bmatrix} 0 \\ 0 \\ 0{,}8272 \\ -1{,}2396 \end{bmatrix}$$

Dibentuk suatu matriks *augmented state* sesuai Persamaan (3.23), sehingga diperoleh matriks *augmented state* berupa:

$$A_a = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0{,}2525 & 0 & 0{,}00013 & 0 \\ 0 & -15{,}0421 & 0 & -0{,}0079 & 0 \\ -1 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{dan} \quad B_a = \begin{bmatrix} 0 \\ 0 \\ 0{,}8272 \\ -1{,}2396 \\ 0 \end{bmatrix}$$

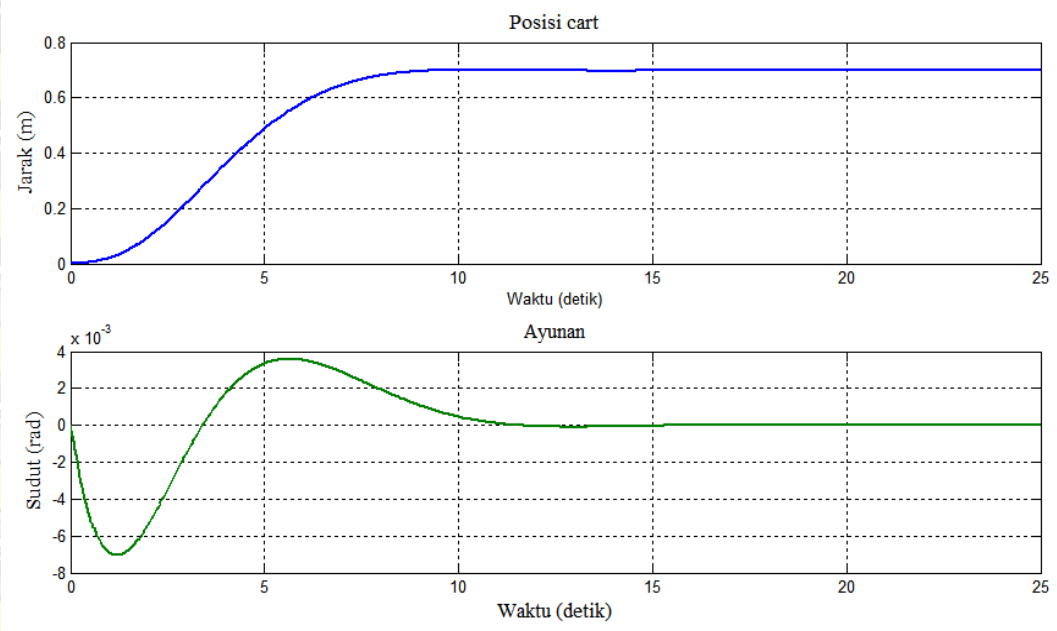
Dengan menyelesaikan Persamaan Ricatti (3.26) menggunakan perintah di Matlab K=LQR(A_a,B_a,Q_a,R), diperoleh besarnya *gain K* optimal sebesar:

$$K_{opt} = [4{,}3488 \;\; -120{,}0656 \;\; 10{,}8464 \;\; -78{,}6033 \;\; -0{,}7314]$$

dengan

$$K_{px} = 4{,}3488 \qquad K_{dx} = 10{,}8464 \qquad K_{ix} = -0{,}7314$$
$$K_{p\theta} = -120{,}0656 \qquad K_{d\theta} = -78{,}6033$$

Nilai-nilai *gain K* ini selanjutnya disimulasikan dengan simulink dan akan menghasilkan *output crane* seperti Gambar 4.17 berikut:



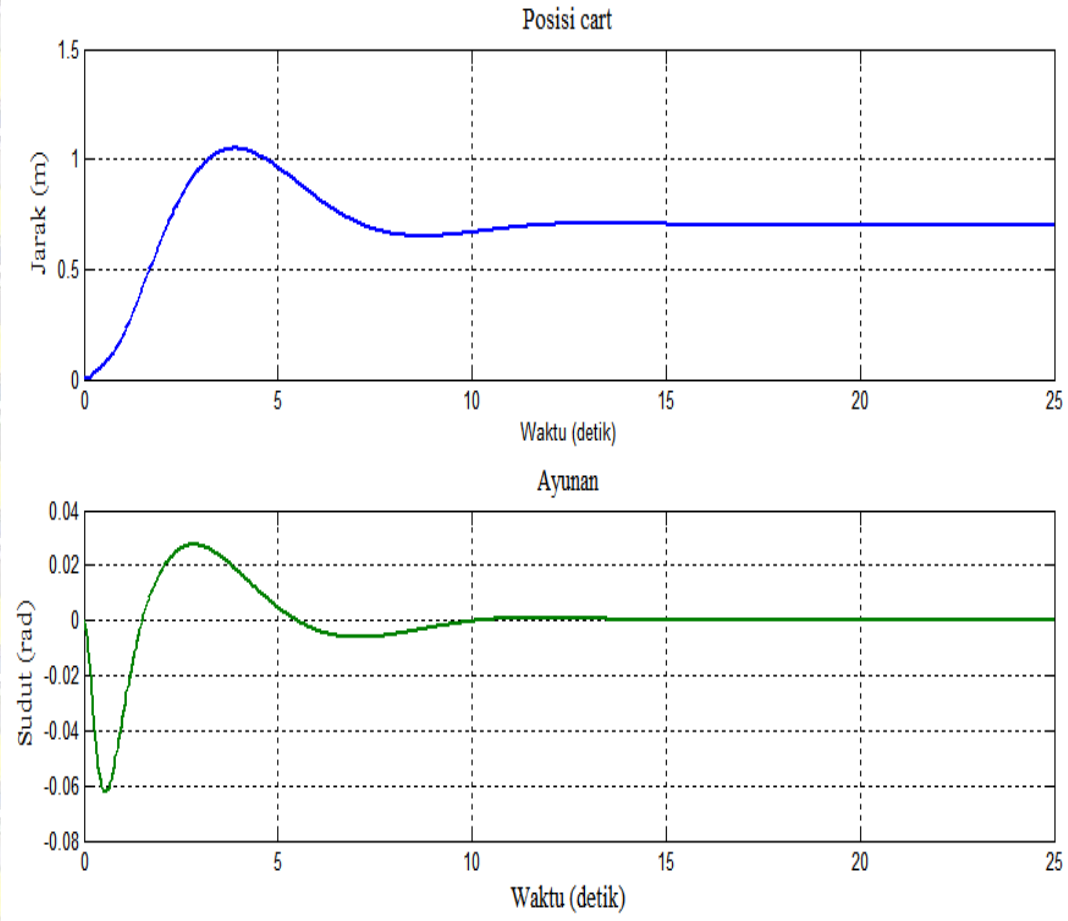
Gambar 4.17. *Output Crane* untuk Kontroler PD-LQRI Optimasi UPSO

Sebagai perbandingan, dipilih kontroler PD-*State Feedback* dengan integral kompensator berdasarkan penelitian Yong-Seok Kim, dkk [4] dari Persamaan (3.34) dan (3.35) dipilih nilai masing-masing sebagai berikut:

$K_{px} = 2{,}8 \qquad K_{dx} = 3{,}9 \qquad K_{ix} = 1$

$K_{p\theta} = -16 \qquad K_{d\theta} = -4{,}5$

Sehingga akan yang menghasilkan *output* sistem *crane* seperti Gambar 4.18 berikut:



Gambar 4.18. *Output Crane* untuk Jenis Kontroler PD-*State Feedback* Integral

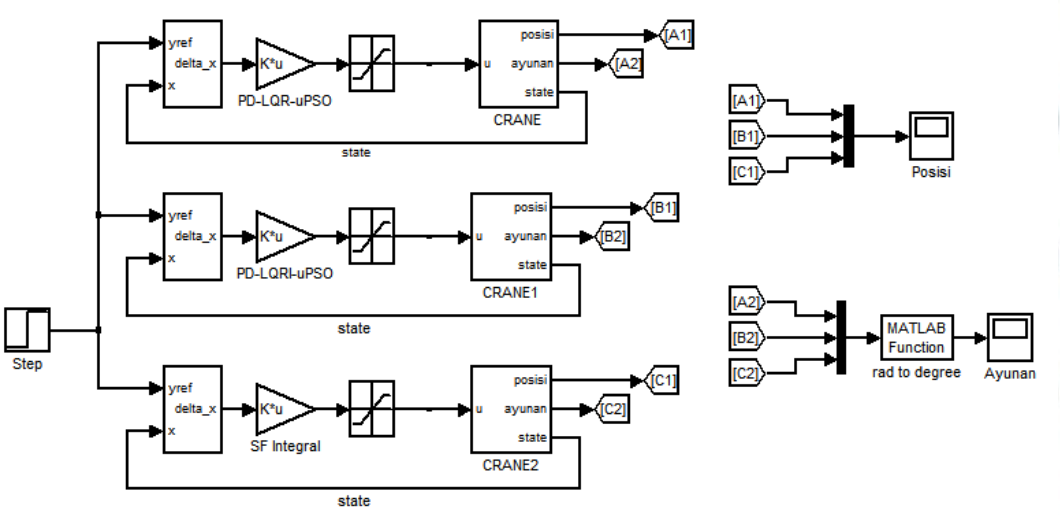
Pengujian dilakukan dengan mensimulasi tiap-tiap kontroler anti ayun yang diuji dalam suatu blok simulink untuk dibandingkan keluaran *output* kontroler masing-masing.

Untuk masing-masing kontroler diisikan dengan *gain K* sesuai Tabel 4.4 berikut.

Tabel 4.4. Nilai *Gain K* untuk Kontroler PD-LQR, PD-LQRI dan PD-SF-Integral

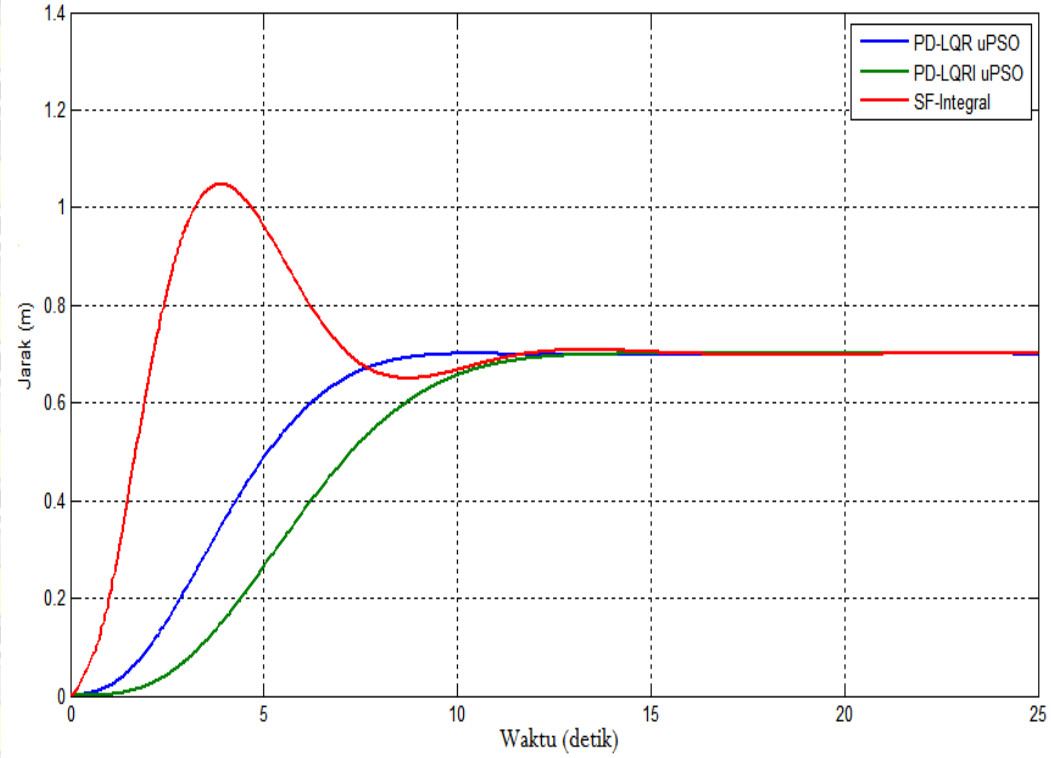
Konstanta *Gain*	Kontroler PD-LQR-uPSO	Kontroler PD-LQRI-uPSO	Kontroler PD-SF-Integral
Kpx (posisi)	1,3523	4,3488	2,8
Kdx (kecepatan)	5,5339	10,8464	3,9
Kp_θ (ayunan)	-72,7614	-120,0656	-16
Kd_θ (kecepatan sudut)	-63,3140	-18,6033	-4,5
Ki (*error* posisi)	-	-0,7314	1

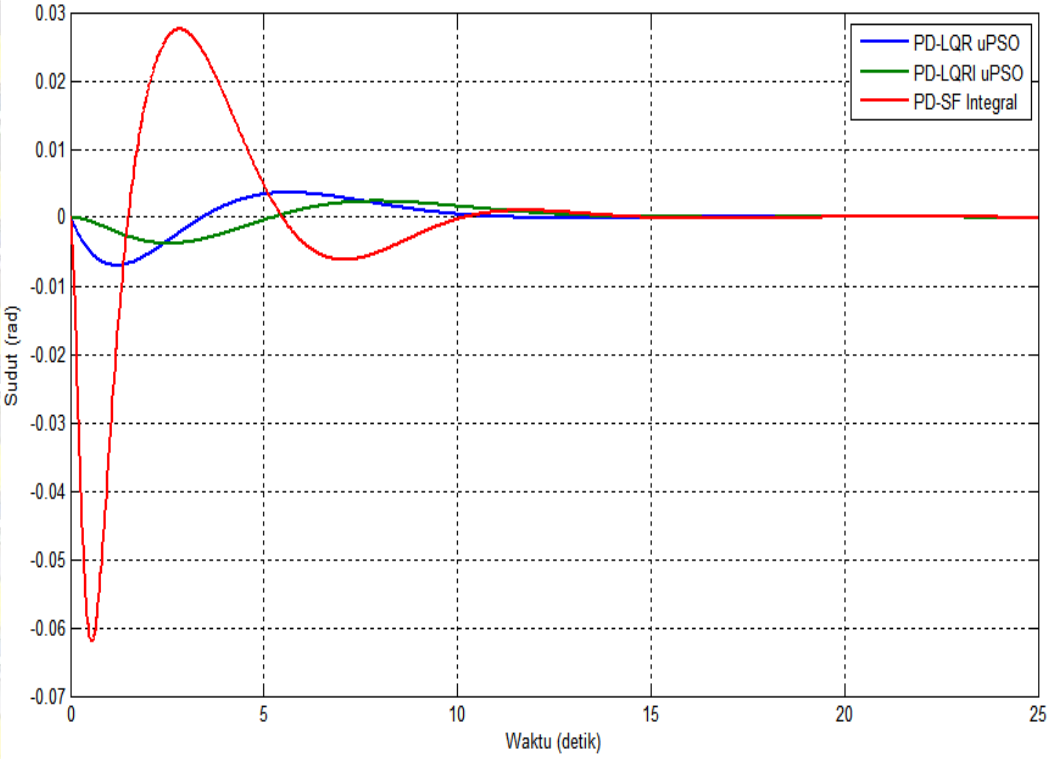
Adapun blok simulink pengujiannya dapat dilihat seperti Gambar 4.19 dibawah ini:



Gambar 4.19. Blok Simulink Pengujian Beberapa Jenis Kontroler Anti Ayun

Berikut tampilan *output plant* dengan kontroler PD-LQR untuk kontroler yang diujikan seperti tampak pada Gambar 4.20.





Gambar 4.20. *Output Crane* untuk Beberapa Jenis Kontroler yang Diuji

Hasil pengujian ini ditampilkan dalam Tabel 4.5, berikut tabel perbandingan karakteristik respon *output* setelah penambahan kompensator Integral.

Tabel 4.5. Perbandingan Kontroler PD-LQR Tanpa dan Dengan Kompensator Integral

Karakteristik respon	Kontroler PD-LQR-uPSO	Kontroler PD-LQRI-uPSO	Kontroler *State Feedback*-I
Settling time (detik)	7,9609	10,9754	10,5344
Steady state error (m)	2,7621e-6	0,0093	3,5528e-5
Max. Overshoot (%)	0,0766	0,0638	**49,7027**
Maksimum ayunan (rad)	0,0071	0,0038	**0,0622**
Integral Square Error	1,5467	2,3024	1,1905
RMS energy drive	0,0682	0,0175	0,0221

Untuk kontroler PD-LQR dengan tambahan kompensator Integral (disebut juga kontroler LQR *type* I), terlihat dapat menghasilkan ayunan beban maksimum yang lebih kecil dibandingkan dengan kontroler PD-LQR *type* 0. Selain itu juga memiliki jumlah pemakaian energi rata-rata (RMS) yang juga lebih kecil dibanding kontroler lain.

Namun pada beberapa parameter lain, kontroler PD-LQR *type* 0 justru lebih baik dibandingkan kontroler PD-LQR *type* I. Walaupun demikian parameter *output* kontroler PD-LQR *type* I tetap berada dalam batas yang disyaratkan saat awal perancangan sistem.

Untuk kontroler PD *State Feedback* dengan kompensator integrator, terdapat dua parameter yang tidak memenuhi persyaratan perancangan sistem, yaitu %OS yang lebih dari 4% dan ayunan maksimum yang lebih dari 0,0436 rad (2,5 derajat).

4.9 Implementasi *Real*

Untuk implementasi eksperimen *plant* di laboratorium, akan dipilih kontroler PD-LQR yang dioptimasi dengan algoritma uPSO dengan *type* matriks *Q* diagonal, dan PD-LQR *type* I yang dikembangkan dari PD-LQR sebelumnya. Sebagai perbandingan juga akan dipilih kontroler PD *Basic* dan kontroler SMC-PI.

Hasil pengukuran diperoleh panjang maksimum jalur *crane* sekitar 0,8324 meter, oleh karenanya untuk nilai x_{ref} dipilih 0,7 meter.

4.9.1 Kontroler PD-LQR UPSO Matriks Q Diagonal

Dari hasil proses optimasi kontroler PD-LQR dengan uPSO untuk implementasi *plant* di laboratorium dengan pemilihan matriks *Q* tipe diagonal diperoleh nilai matriks bobot *Q* dan *R* optimal sebesar:

$$Q_{opt} = \begin{bmatrix} 0{,}0393 & 0 & 0 & 0 \\ 0 & 0{,}0006 & 0 & 0 \\ 0 & 0 & 0{,}1560 & 0 \\ 0 & 0 & 0 & 92{,}4696 \end{bmatrix} \quad \text{dan} \quad R = [0{,}0215]$$

Selanjutnya, untuk matriks *state* A dan B berikut

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0{,}2525 & 0 & 0{,}00013 \\ 0 & -15{,}0421 & 0 & -0{,}0079 \end{bmatrix} \text{ dan } B = \begin{bmatrix} 0 \\ 0 \\ 0{,}8272 \\ -1{,}2396 \end{bmatrix}$$

Dengan menyelesaikan persamaan Ricatti (3.7), menggunakan perintah di Matlab K=LQR(A,B,Q_{opt},R_{opt}), diperoleh besarnya *gain K* optimal sebesar:

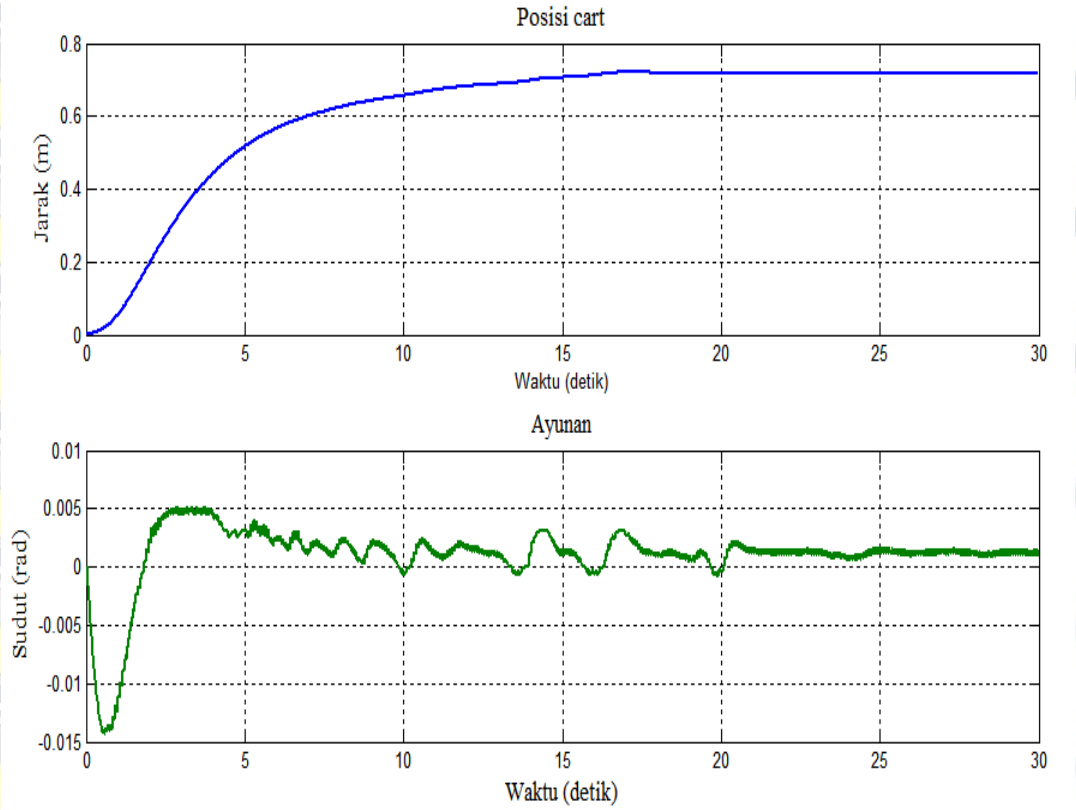
$$K = [1{,}3523 \;\; -72{,}7614 \;\; 5{,}5375 \;\; -64{,}3140]$$

dimana

$K_{px} = 1{,}3523$ $\qquad$ $K_{dx} = 5{,}5375$

$K_{p\theta} = 72{,}7614$ $\qquad$ $K_{d\theta} = 64{,}3140$

Adapun *output* kontroler PD-LQR optimasi uPSO untuk implementasi *real plant* di laboratorium dapat dilihat pada Gambar 4.21 dibawah:



Gambar 4.21. *Output* Kontroler PD-LQR UPSO untuk Implementasi *Plant Real*

4.9.2 Kontroler PD-LQR *Type* I Optimasi UPSO

Dari hasil proses optimasi kontroler PD-LQRI dengan uPSO untuk implementasi *plant* di laboratorium dengan pemilihan matriks *Q type* diagonal diperoleh nilai matriks bobot *Q* dan *R* optimal sebesar:

$$Q_a = \begin{bmatrix} 0{,}0393 & 0 & 0 & 0 & 0 \\ 0 & 0{,}0006 & 0 & 0 & 0 \\ 0 & 0 & 0{,}7727 & 0 & 0 \\ 0 & 0 & 0 & 92{,}469 & 0 \\ 0 & 0 & 0 & 0 & 0{,}0068 \end{bmatrix} \text{ dan}$$

$$R_a = [0{,}0129]$$

Dan matriks *state augmented* berupa:

$$A_a = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0{,}2525 & 0 & 0{,}00013 & 0 \\ 0 & -15{,}0421 & 0 & -0{,}0079 & 0 \\ -1 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{dan} \quad B_a = \begin{bmatrix} 0 \\ 0 \\ 0{,}8272 \\ -1{,}2396 \\ 0 \end{bmatrix}$$

Dengan menyelesaikan Persamaan Ricatti (3.26) menggunakan perintah di Matlab K=LQR(A_a,B_a,Q_a,R), diperoleh besarnya *gain K* optimal sebesar:

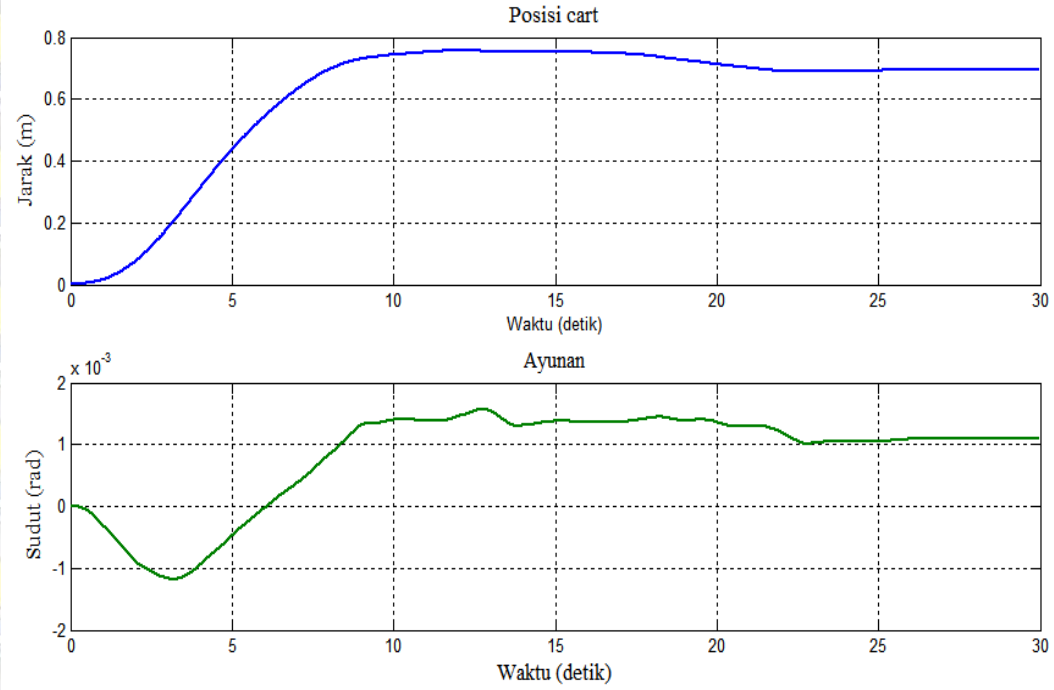
$$K_{opt} = [4{,}3488 \quad -120{,}0656 \quad 10{,}8464 \quad -78{,}6033 \quad -0{,}7314]$$

Dengan untuk implementasi pada kontroler PD-LQR

$$K_{px} = 4{,}3488 \qquad K_{dx} = 10{,}8464 \qquad K_{ix} = -0{,}7314$$

$$K_{p\theta} = -120{,}0656 \qquad K_{d\theta} = -78{,}6033$$

Adapun *output* kontroler PD-LQRI berdasarkan hasil implementasi pada *plant* di laboratorium dapat dilihat pada Gambar 4.22 berikut.



Gambar 4.22. *Output* Kontroler PD-LQRI UPSO untuk Implementasi *Plant Real*

4.9.3 Kontroler PD *Basic*

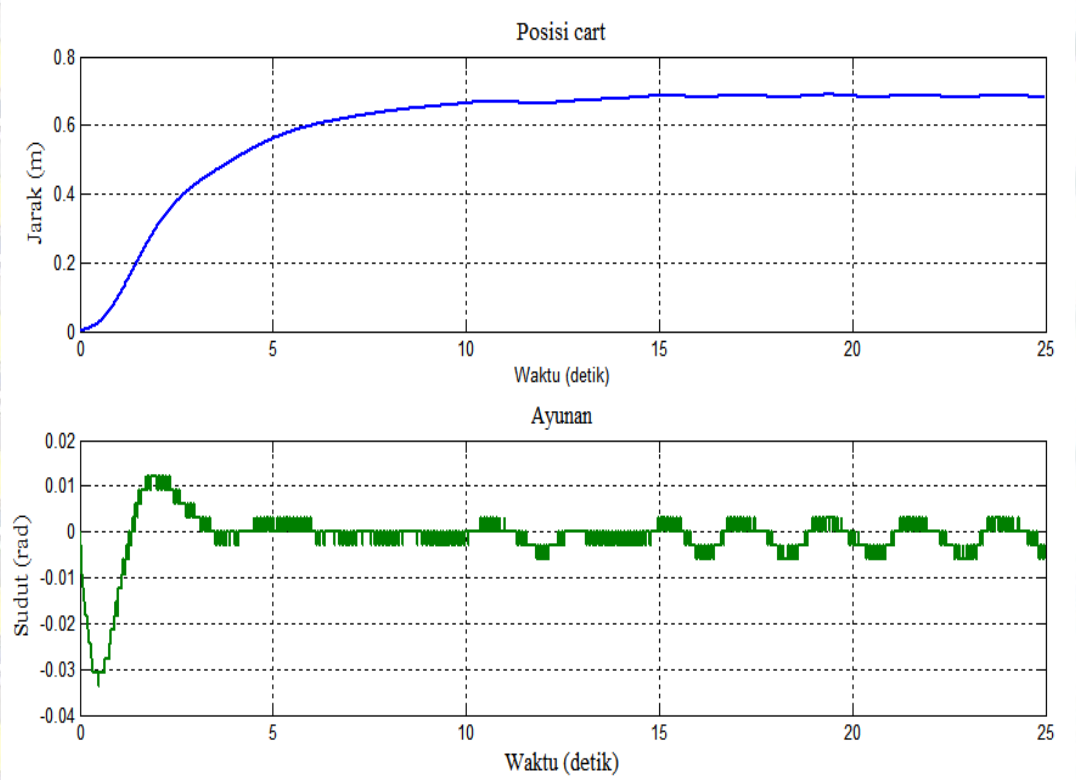
Dengan menyelesaikan Persamaan (3.30) dan (3.31), diperoleh *gain K* sebesar:

$K = [1 \quad 10 \quad 2 \quad 6{,}31]$

dan besar sinyal kontrol *u* didapat:

$u = (1+2s)(x_{ref} - x) + (10 + 6{,}31s).\theta$

Adapun *output* kontroler untuk *plant* implementasi *real* di laboratorium dapat dilihat pada Gambar 4.23 berikut.



Gambar 4.23. *Output* Kontroler PD-*Basic* untuk Implementasi *Plant Real*

4.9.4 Kontroler SMC-PI

Dengan menyelesaikan Persamaan (3.32) dan (3.33), diperoleh *gain K* sebesar

$K = [1 \quad -12{,}15 \quad 2 \quad 0{,}802]$

Sehingga besarnya sinyal kontrol u adalah:

$u = u_1 - u_{SMC}$

sinyal kontrol posisi :

$u_1 = (K_{px} + sK_{dx})(x_{ref} - x_1)$

$u_1 = (1+2s)(x_{ref} - x)$

sinyal kontrol ayunan :

$$u_{SMC} = [-K_{p\theta} x_2 + K_{d\theta} x_4 - k.sat(S, \varepsilon)]$$

$$S = x_4 + \lambda x_2 = x_4 + 1x_2$$

$$u_{SMC} = [-12{,}15x_2 + 0{,}802x_4 - 1.sat((x_4 + x_2), \varepsilon)]$$

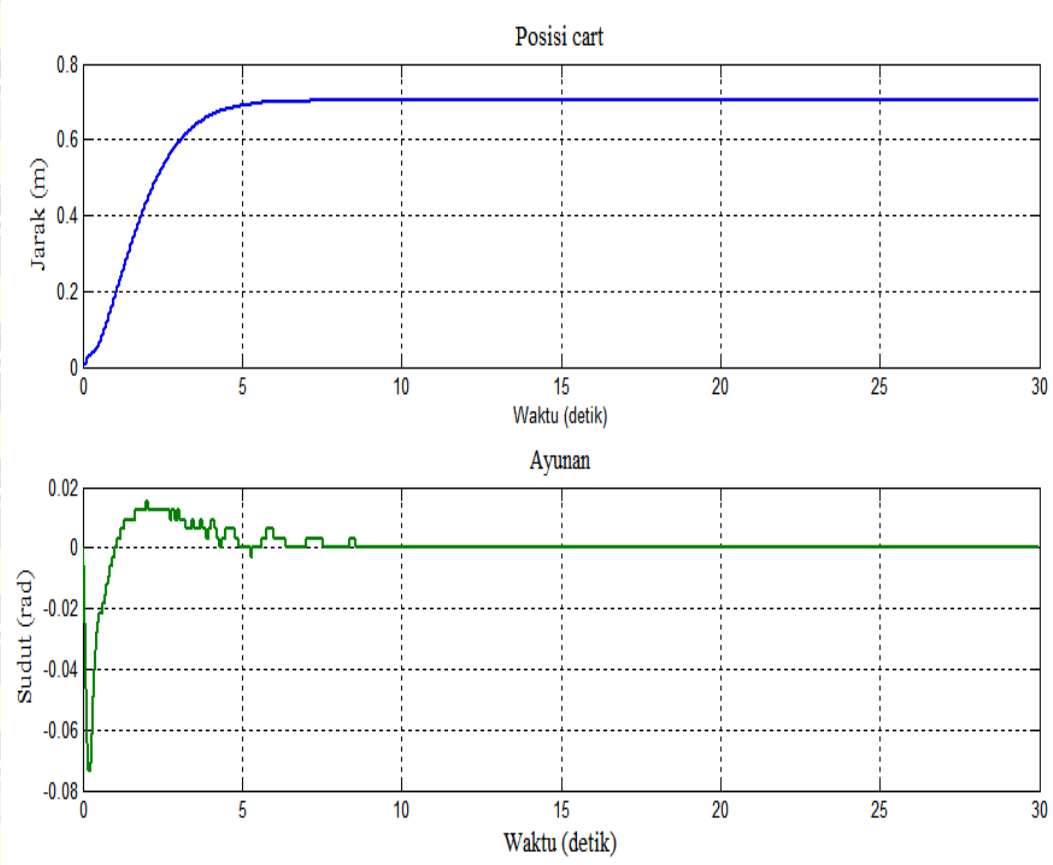
$$u_{SMC} = [-12{,}15\theta + 0{,}802\dot{\theta} - 1.sat((\dot{\theta} + \theta), \varepsilon)]$$

sehingga diperoleh :

$$u = u_1 - u_{SMC}$$

$$u = [(1+2s)(x_{ref} - x)] - [-12{,}15.\theta + 0{,}802.\dot{\theta} - 1.sat((\dot{\theta} + \theta), \varepsilon)]$$

Adapun *output plant* implementasi *real* kontroler SMC di laboratorium dapat dilihat pada Gambar 4.24.



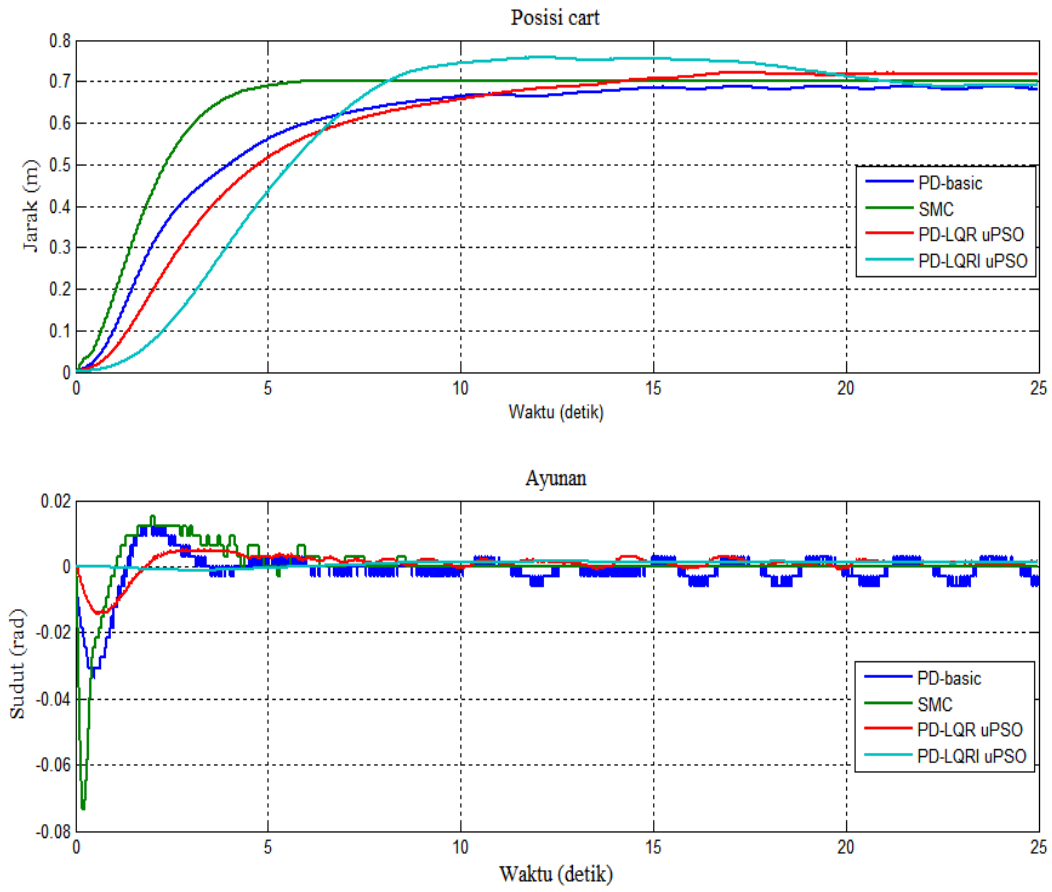
Gambar 4.24. *Output* Kontroler SMC untuk Implementasi *Plant Real*

Hasil implementasi ini ditampilkan dalam bentuk tabel dan juga gambar yang memperlihatkan karakteristik *output* tiap-tiap kontroler, baik secara simulasi nonlinear maupun hasil implementasi langsung.

Berikut karakteristik respon *output* dari tiap-tiap kontroler yang diujikan seperti yang terdapat pada Tabel 4.6 dan Tabel 4.7 dan *output* sistem pada Gambar 4.25.

Tabel 4.6. Perbandingan *Output* Kontroler untuk Simulasi *Plant* Nonlinear

Karakteristik respon	Kontroler PD *basic*	Kontroler SMC	PD-LQR-uPSO	PD-LQRI-uPSO
Settling Time (Ts) (detik)	7,6499	9,1	7,9719	10,9754
ESS (e_{ss}) (m)	8,4794e-7	1,5467e-5	2,7621e-6	0,0093
Persen *overshoot* (%OS)	**7,8286**	**8,3224**	0,0765	0,0638
Maksimum ayunan (rad)	0,0250	0,0392	0,0071	0,0038
Integral Square Error	0,9576	0,9387	1,5466	2,3024
RMS energy drive	0,916	0,1054	0,0734	0,0175



Gambar 4.25. Perbandingan *Output* Kontroler untuk Implementasi *Plant Real*

Tabel 4.7. Perbandingan *Output* Kontroler untuk Implementasi *Plant Real*

Karakteristik respon	Kontroler PD *basic*	Kontroler SMC	PD-LQR-uPSO	PD-LQRI-uPSO
Settling Time (Ts) (detik)	10,3370	12,3960	17,8300	19,4410
ESS (meter)	0,0242	0,0070	0,0145	0,0019
Persen *overshoot* (%OS)	0,5686	0,3075	3,0823	**8,2729**
Maksimum ayunan (rad)	0,0338	**0,0738**	0,0145	0,0014
Integral Square Error	5,3719	16,3643	5,3719	5,4378
RMS energy drive	0,3505	0,1523	0,5963	0,6327

Walaupun hasil simulasi pada *plant* nonlinear kontroler PD-LQR type I (PD-LQRI) memiliki parameter *output* yang cukup baik, namun dalam implementasi real ternyata menghasilkan persen *overshoot* yang paling besar dibanding kontroler lainnya.

Dari hasil simulasi, diperoleh jumlah pemakaian energi rata-rata (RMS) pada kontroler PD-LQRI lebih kecil dibanding kontroler lain, namun hasil dari implementasi ternyata kontroler PD-LQRI memiliki pemakaian energi yang lebih besar dibanding kontroler yang lain.

Untuk parameter besar ayunan beban, terlihat bahwa baik kontroler PD-LQR maupun kontroler PD-LQRI menghasilkan ayunan paling kecil dibanding kontroler PD *basic* maupun SMC.

Dari hasil implementasi, kontroler PD-LQR maupun PD-LQRI memiliki pemakaian energi rata-rata yang lebih besar dibanding kontroler yang lain, ini karena pada kontroler PD-LQR maupun PD-LQRI, saat titik referensi *crane* cenderung bergetar untuk mencapai posisi referensi dan untuk menjaga ayunan beban tetap kecil. Sedangkan kontroler SMC lebih stabil dalam mencapai posisi referensinya.

Hasil implementasi kontroler PD-LQR memiliki parameter *output* sesuai yang diharapkan yaitu masih dalam batas yang disyaratkan saat perancangan sistem, sedangkan kontroler PD-LQRI terdapat parameter persen *overshoot* yang melebihi batas persyaratan perancangan awal sebesar 4%.

BAB V
PENUTUP

5.1 Kesimpulan

Untuk menghasilkan kontroler PD-LQR yang lebih optimal, optimasi dengan algoritma uPSO terbukti dapat dipakai untuk menentukan besaran matriks pembobot *Q* dan *R* yang sesuai dengan kriteria *output* sistem yang diinginkan.

Adapun dalam memilih matriks pembobot *Q* dan *R* pada kontroler PD-LQR, pemilihan matriks *Q* berbentuk diagonal dapat dijadikan pilihan awal sebagai pilihan yang paling mudah.

Penambahan kompensator integral pada kontroler PD-LQR *type* 0 sehingga dihasilkan kontroler PD-LQR *type* I, terbukti mampu mengurangi ayunan beban maksimum dan mengurangi pemakaian rata-rata energi kontrol, sehingga kontroler PD-LQR *type* I layak dipilih untuk hasil yang lebih optimal.

Ketika diimplementasikan pada *plant* SPK, kontroler PD-LQR dan PD-LQRI cenderung menggunakan energi kontrol yang lebih besar dibanding kontroler SMC. Walaupun demikian kontroler SMC memiliki ayunan awal yang cenderung besar dibandingkan kontroler lain

5.2 Saran

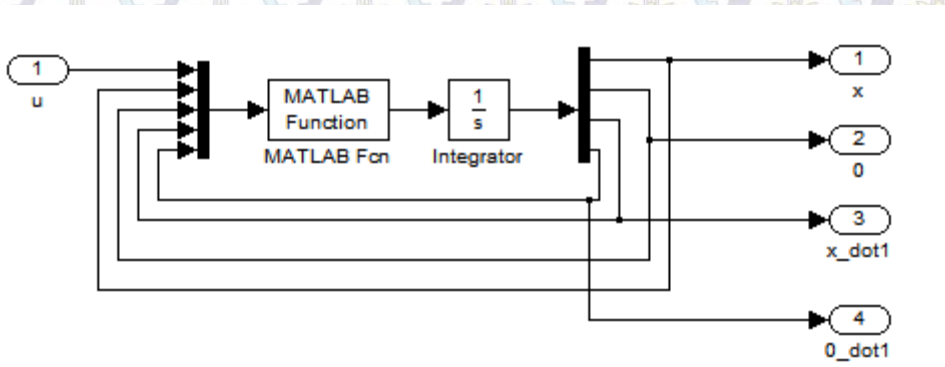
Untuk implementasi *real* kiranya dapat diterapkan pada *plant* yang memiliki panjang lintasan lebih dari 1 meter.

Perlunya ada tambahan mekanisme gangguan (*perturbation*) saat proses optimasi mencapai keadaan tenang (*steady*) pada daerah sekitar nilai optimalnya.

-----Halaman ini sengaja dikosongkan-----

LAMPIRAN

Model Simulink Sistem *Crane* untuk proses simulasi



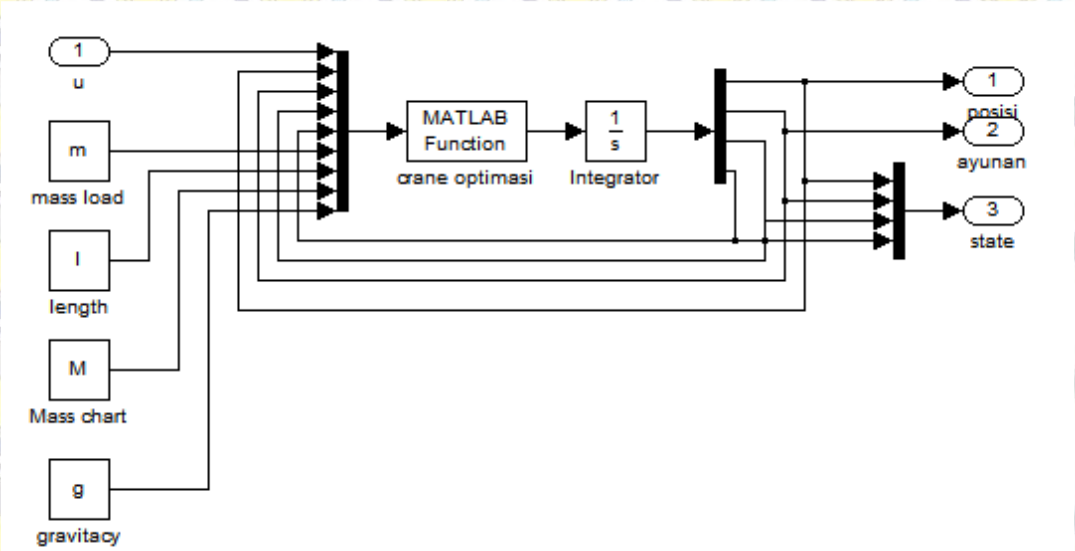
Gambar A1: Model Simulink Sistem *Crane*

Kode M-File fungsi Matlab Sistem_crane.m

```
function out=modelSScrane(in);
%parameters
mc=1.12 % mass of crane
l=0.0167903;  %
mp=0.12; % mass of pendulum
g=9.8; %gravity constant
J=0.0135735;
fp=0.000107;
T=2.5316;
miu=(mc+mp)*l;
a=l^2+J/(mc+mp);

%input
u=in(1);

%state
x1=in(2); %X
x2=in(3)-pi; %Theta
x3=in(4); %X_dot
x4=in(5); %Theta_dot

%state-space equation
x1_dot=x3;
x2_dot=x4;
x3_dot=(a*(u-miu*x4^2*sin(x2))+(l*cos(x2)*(miu*g*sin(x2)-
fp*x4)))/(J+miu*l*sin(x2)^2);
x4_dot=((l*cos(x2)*(u-miu*x4^2*sin(x2)))+(miu*g*sin(x2)-
fp*x4))/(J+miu*l*sin(x2)^2);
```

```
%output
out(1)=x1_dot;
out(2)=x2_dot;
out(3)=x3_dot;
out(4)=x4_dot;
```

Model Simulink Sistem *Crane* untuk proses optimasi



Gambar A2: Model Simulink Sistem *Crane* untuk optimasi