

TUGAS AKHIR - EE184701

KONTROL KECEPATAN MOTOR DC DENGAN ESTIMATOR KECEPATAN EXTENDED KALMAN FILTER DAN KONTROLER PROPORTIONAL INTEGRAL BERBASIS PARTICLE SWARM OPTIMIZATION

KIET PASCAL ASMARA

NRP 07111940000110

Dosen Pembimbing

Eka Iskandar, S.T., M.T.

NIP 198005282008121001

Yusuf Bilfaqih, S.T., M.T.

NIP 197203251999031001

Program Studi Sarjana Teknik Elektro

Departemen Teknik Elektro

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2023



TUGAS AKHIR - EE184701

KONTROL KECEPATAN MOTOR DC DENGAN ESTIMATOR KECEPATAN EXTENDED KALMAN FILTER DAN KONTROLER PROPORTIONAL INTEGRAL BERBASIS PARTICLE SWARM OPTIMIZATION

KIET PASCAL ASMARA

NRP 07111940000110

Dosen Pembimbing

Eka Iskandar, S.T., M.T.

NIP 198005282008121001

Yusuf Bilfaqih, S.T., M.T.

NIP 197203251999031001

Program Studi Sarjana Teknik Elektro

Departemen Teknik Elektro

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

Tahun 2023



FINAL PROJECT - EE184701

SPEED CONTROL OF DC MOTOR USING EXTENDED KALMAN FILTER ESTIMATOR AND PARTICLE SWARM OPTIMIZATION BASED PROPORTIONAL INTEGRAL CONTROLLER

KIET PASCAL ASMARA

NRP 07111940000110

Advisor

Eka Iskandar, S.T., M.T.

NIP 198005282008121001

Yusuf Bilfaqih, S.T., M.T.

NIP 197203251999031001

Bachelor Program of Electrical Engineering

Department of Electrical Engineering

Faculty of Intelligent Electrical and Information Technology

Sepuluh Nopember Institute of Technology

Surabaya

Year 2023

LEMBAR PENGESAHAN

KONTROL KECEPATAN MOTOR DC DENGAN ESTIMATOR KECEPATAN EXTENDED KALMAN FILTER DAN KONTROLER PROPORTIONAL INTEGRAL BERBASIS PARTICLE SWARM OPTIMIZATION

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat

memperoleh gelar Sarjana pada

Program Studi S-1 Teknik Elektro

Departemen Teknik Elektro

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Oleh : **KIET PASCAL ASMARA**

NRP. 07111940000110

Disetujui oleh Tim Penguji Tugas Akhir :

1. Eka Iskandar, S.T., M.T.

Pembimbing

2. Yusuf Bilfaqih, S.T., M.T.

Ko-pembimbing

3. Dr. Trihastuti Agustinah, ST., MT.

Penguji

4. Ir. Ali Fatoni, MT.

Penguji

5. Mohamad Abdul Hady S.T., M.T.

Penguji

SURABAYA

Juni, 2023

PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini:


Nama mahasiswa / NRP : Kiet Pascal Asmara / 07111940000110
Program studi : Teknik Elektro
Dosen Pembimbing / NIP : Eka Iskandar, S.T., M.T. / NIP 198005282008121001

dengan ini menyatakan bahwa Tugas Akhir dengan judul “Kontrol Kecepatan Motor DC Dengan Estimator Kecepatan Extended Kalman Filter Dan Kontroler Proportional Integral Berbasis Particle Swarm Optimization” adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.


Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, 13 Juni 2023

Mengetahui
Dosen Pembimbing


Eka Iskandar, S.T., M.T.
NIP. 198005282008121001

Mahasiswa


Kiet Pascal Asmara
NRP. 07111940000110

ABSTRAK

KONTROL KECEPATAN MOTOR DC DENGAN ESTIMATOR KECEPATAN EXTENDED KALMAN FILTER DAN KONTROLER PROPORTIONAL INTEGRAL BERBASIS PARTICLE SWARM OPTIMIZATION

Nama Mahasiswa / NRP : Kiet Pascal Asmara / 07111940000110
Departemen : Teknik Elektro FTEIC - ITS
Dosen Pembimbing : 1. Eka Iskandar, S.T., M.T.
2. Yusuf Bilfaqih, S.T., M.T.

Abstrak

Motor DC merupakan salah satu motor yang paling banyak digunakan dalam industri maupun keseharian. Untuk berbagai aplikasi dari motor DC, pengendalian kecepatan dan torsi secara presisi diperlukan, menggunakan sensor kecepatan seperti tachogenerator untuk mengetahui nilai kecepatan dari motor. Tetapi, khususnya dalam aplikasi industri, sensor tersebut cukup mahal dan juga meningkatkan kemungkinan munculnya gangguan dari kegagalan sensor. Maka, dikembangkan teknik kendali *sensorless* yang dapat mengestimasi kecepatan motor dengan sinyal listrik, tanpa sensor kecepatan. Pada penelitian ini, akan digunakan estimator *Extended Kalman Filter* (EKF) dan kontroler *Proportional Integral* (PI) berbasis *Particle Swarm Optimization* (PSO). EKF akan mengestimasi kecepatan rotor menggunakan pengukuran tegangan dan arus dari sensor listrik. Lalu, input arus jangkar akan dikendalikan oleh kontroler PI yang dioptimisasikan dengan algoritma PSO. Algoritma PSO akan mencari nilai parameter PI optimal dengan mengoptimisasikan nilai *error* dan respon transien dalam fungsi objektif. Algoritma PSO tersebut menghasilkan parameter PI optimal bernilai (3.9406, 20.6850) yang menghasilkan *overshoot* sebesar 8.83%, *settling time* sebesar 0.678s, *rise time* sebesar 0.0312s, dan ITAE sebesar 1.1667. EKF menghasilkan kecepatan estimasi dengan RMSE sebesar 1.538, yang lebih kecil dibandingkan kecepatan aktual yang memiliki RMSE sebesar 2.045.

Kata kunci: *Motor DC, PSO, EKF, Fungsi objektif*

ABSTRACT

SPEED CONTROL OF DC MOTOR USING EXTENDED KALMAN FILTER ESTIMATOR AND PARTICLE SWARM OPTIMIZATION BASED PROPORTIONAL INTEGRAL CONTROLLER

Student Name / NRP : Kiet Pascal Asmara / 07111940000110
Department : Electrical Engineering FTEIC - ITS
Advisor : 1. Eka Iskandar, S.T., M.T.
2. Yusuf Bilfaqih, S.T., M.T.

Abstract

DC motors are one of the most widely used motors in industry and everyday life. For many applications of DC motors, precise control of speed and torque is required, using a speed sensor such as a tachogenerator to determine the speed value of the motor. However, especially in industrial applications, these sensors are quite expensive and also increase the possibility of interference from sensor failures. Thus, a sensorless control technique was developed which can estimate the motor speed with an electrical signal, without a speed sensor. In this study, the Extended Kalman Filter (EKF) estimator and Proportional Integral (PI) controller based on Particle Swarm Optimization (PSO) will be used. EKF will estimate the rotor speed using voltage and current measurements from electrical sensors. Then, the armature current input will be controlled by the PI controller which is optimized with the PSO algorithm. The PSO algorithm will find the optimal PI parameter value by optimizing the error value and transient response in the objective function. The PSO algorithm produces optimal PI parameters worth (3.9406, 20.6850) which results in an overshoot of 8.82%, a settling time of 0.6783s, a rise time of 0.0312s, and an ITAE of 1.1667. EKF produces an estimated speed with an RMSE of 1.538, which is smaller than the actual speed which has an RMSE of 2.045.

Keywords: DC Motor, PSO, EKF, Objective Function

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena atas rahmat-Nya penulis dapat menyelesaikan penelitian dan tugas akhir dengan judul “Kontrol Kecepatan Motor Dc dengan Estimator Kecepatan Extended Kalman Filter dan Kontroler Proportional Integral Berbasis Particle Swarm Optimization” dengan baik. Tugas akhir ini diajukan untuk memenuhi salah satu syarat kelulusan untuk memperoleh gelar Sarjana Teknik dan merupakan mata kuliah wajib di Departemen Teknik Elektro Institut Teknologi Sepuluh Nopember.

Dalam pengerjaan dan pelaksanaan penelitian ini, penulis telah dibantu dan didukung oleh banyak pihak. Oleh karena itu penulis menyampaikan banyak terima kasih kepada :

1. Orang tua dan keluarga penulis yang selalu mendukung dan mendoakan penulis selama masa pengerjaan tugas akhir.
2. Bapak Eka Iskandar S.T., M.T. dan Bapak Yusuf Bilfaqih, S.T., M.T. selaku dosen pembimbing tugas akhir penulis yang selalu memberikan saran, arahan, dan toleransi kepada penulis dalam masa pengerjaan tugas akhir.
3. Ibu Dr. Trihastuti Agustinah, ST., MT. selaku dosen wali penulis selama berkuliah di Departemen Teknik Elektro ITS.
4. Bapak dan Ibu Dosen serta tenaga pendidik di Departemen Teknik Elektro ITS, khususnya bidang studi Teknik Sistem Pengaturan
5. Teman-teman ITS yang tergabung dalam grup KC terutama Daniel, Goldian, Sastro, Altaira, dan Gilbert yang selalu menemani dan membantu penulis dalam masa perkuliahan.
6. Teman-teman anggota grup “DOGCIX” yaitu Ananta, Rommero, Kevin, Deo, Timmy, Haryo, Nicodemus, Arya, dan Jovan yang selalu memberi semangat kepada penulis untuk menyelesaikan tugas akhir ini.
7. Teman-teman dekat penulis, Ryan dan Farrel yang selalu menyemangati penulis selama masa perkuliahan berlangsung.

Dalam menyusun tugas akhir penulis menyadari masih terdapat banyak kekurangan dikarenakan keterbatasan pengetahuan serta pengalaman penulis. Saran dan kritik penulis harapkan untuk dijadikan perbaikan dan pengembangan sehingga tugas akhir ini dapat memberikan manfaat bagi pembaca.

Surabaya, 19 Juli 2023



Kiet Pascal Asmara

Halaman ini sengaja dikosongkan

DAFTAR ISI

LEMBAR PENGESAHAN	i
PERNYATAAN ORISINALITAS	ii
ABSTRAK	iii
ABSTRACT	iv
DAFTAR ISI	viii
DAFTAR GAMBAR	viii
DAFTAR TABEL	xi
DAFTAR SIMBOL	xii
BAB 1 PENDAHULUAN	14
1.1 Latar Belakang	14
1.2 Rumusan Masalah	14
1.3 Tujuan	15
1.4 Batasan Masalah	10
1.5 Manfaat	15
BAB 2 TINJAUAN PUSTAKA	16
2.1 Hasil Penelitian Terdahulu	16
2.2 Dasar Teori	16
BAB 3 METODOLOGI	22
3.1 Metode yang digunakan	22
3.2 Bahan dan peralatan yang digunakan	22
3.3 Desain Eksperimen	20
3.4 Urutan pelaksanaan penelitian	20
BAB 4 Hasil dan Pembahasan	32
4.1 Pengujian PI-PSO	26
4.2 Pengujian Estimator EKF	30
BAB 5 Kesimpulan dan Saran	41
5.1 Kesimpulan	41
5.2 Saran	41
DAFTAR PUSTAKA	42
LAMPIRAN	44
BIODATA PENULIS	47

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Gambar 2.1 Skematik motor DC separately excited	(12)
Gambar 2.2 Model beban nonlinear motor DC	(13)
Gambar 3.1 Diagram Alur Penelitian.....	(17)
Gambar 3.2 Diagram blok sistem estimasi dan kontrol kecepatan motor DC dengan EKF dan PI-PSO.....	(18)
Gambar 3.3 Model sistem estimasi dan kontrol kecepatan motor DC dengan EKF dan PI-PSO di Simulink.....	(18)
Gambar 3.4 Model motor DC di Simulink.....	(20)
Gambar 3.5 Model beban pada Simulink.....	(20)
Gambar 3.6 Subsistem blok EKF pada Simulink.....	(24)
Gambar 4.1 Grafik perubahan fitness pada skenario 5.....	(26)
Gambar 4.2a Respon kecepatan dengan solusi skenario 1.....	(27)
Gambar 4.2b Respon kecepatan dengan solusi skenario 5.....	(28)
Gambar 4.3a Respon kecepatan PI $K_p=100$ dan $K_i=50$	(29)
Gambar 4.3b Respon kecepatan PI $K_p=100$ dan $K_i=50$ diperbesar.....	(29)
Gambar 4.4 RMSE kecepatan estimasi dan aktual dengan setpoint konstan $\omega_{ref}=100$ rad/s.....	(30)
Gambar 4.4a Respon kecepatan dengan setpoint konstan $\omega_{ref}=100$ rad/s 30.....	(31)
Gambar 4.5 Arus estimasi dan aktual dengan setpoint konstan $\omega_{ref}=100$ rad/s.....	(31)
Gambar 4.6 Pengujian setpoint berubah-ubah.....	(32)
Gambar 4.7 RMSE estimasi dan aktual dengan setpoint berubah-ubah.....	(33)
Gambar 4.8 Arus estimasi dan aktual dengan setpoint berubah-ubah.....	(33)

DAFTAR TABEL

Tabel 2.1 Karakteristik kontroler PID.....	(15)
Tabel 3.1 Desain Eksperimen.....	(21)
Tabel 3.2 Parameter model motor DC.....	(21)
Tabel 3.3 Parameter algoritma PSO.....	(21)
Tabel 4.1 Hasil pengujian PSO.....	(26)
Tabel 4.2 Respon transien kecepatan motor untuk skenario 2 dan 7.....	(28)
Tabel 4.3 Pembobotan respon transien pada nilai fitness skenario 7.....	(29)
Tabel 4.4 Hasil pengujian PSO-ITAE.....	(30)
Tabel 4.5 Respon transien sistem dengan <i>setpoint</i> konstan $\omega_{ref} = 100$	(31)

DAFTAR SIMBOL

$v(t)$: tegangan suplai armature

$e_a(t)$: back emf

R_a : resistansi armature

$i_a(t)$: arus armature

L_a : induktansi armature

K_e : konstan back emf

ω_m : kecepatan motor

$T(t)$: torsi motor

K_t : konstan torsi

J : momen inersia motor

$T_L(t)$: torsi beban

$T_f(t)$: torsi gesekan coulomb

D : koefisien damping viscous

g : konstan gravitasi

L : panjang lengan

θ : percepatan sudut

T_s : waktu sampling

x_k : variabel state input

y_k : vektor output

u_k : vektor input

P_k : matriks kovarians prediksi

Q : matriks kovarians pengukuran

R : matriks kovarians proses

F_k : matriks *jacobian* transisi state

S_k : kovarians inovasi

K_k : *Kalman Gain*

H_k : matriks *jacobian* observasi

Halaman ini sengaja dikosongkan

BAB 1 PENDAHULUAN

Pada bab ini akan dibahas mengenai latar belakang dilakukannya penelitian ini, rumusan masalah yang dibahas pada penelitian ini, batasan masalah dari penelitian ini, tujuan dan manfaat penelitian.

1.1 Latar Belakang

Motor DC merupakan salah satu mesin listrik yang paling banyak digunakan dalam industri, seperti dalam rolling mill angin, kipas, lengan robot dan aplikasi lainnya. Motor tersebut memerlukan pengendalian kecepatan atau posisi secara presisi agar bisa diaplikasikan pada berbagai proses dalam industri. Dengan menggunakan sistem pengendalian *feedback*, kecepatan motor DC dapat dikendalikan dan performa motor dapat meningkat secara drastis dibandingkan dengan sistem *open loop*. Umumnya, beberapa sensor seperti *tachogenerator*, *encoder*, dan *resolver*, dipasang pada poros motor untuk mengukur kecepatan motor sebagai *feedback*. (Aydogmus et al., 2014) Tetapi, terutama dalam aplikasi industri, sensor tersebut cukup mahal dan juga meningkatkan kemungkinan munculnya gangguan dari kegagalan sensor. Oleh karena itu, dikembangkan metode pengendalian *sensorless* yang tidak menggunakan sensor kecepatan, hanya sensor listrik.

Dalam pengendalian *sensorless*, kecepatan motor diestimasi menggunakan pengukuran sensor listrik menggunakan sebuah *observer*. *Observer* tersebut melakukan estimasi kecepatan sudut motor menggunakan model matematika motor dan sinyal listrik, umumnya dengan nilai arus dan tegangan pada armature (Rigatos, 2008). Sinyal estimasi ini kemudian di *feedback* dan dibandingkan dengan sinyal referensi untuk menghasilkan *error*. *Error* tersebut menjadi input sebuah kontroler yang akan menghasilkan sinyal input bagi motor.

Dalam penelitian ini observer yang akan digunakan adalah *Extended Kalman Filter* atau EKF yang mampu melakukan estimasi state pada sistem nonlinier. EKF digunakan karena model motor DC dalam penelitian ini berbeban nonlinier. dan kontroler yang digunakan adalah kontroler *Proportional Integral* atau PI, berbasis algoritma *Particle Swarm Optimization* atau PSO. PSO merupakan teknik optimisasi yang berdasarkan konsep pergerakan dari kumpulan organisme seperti burung atau ikan dalam mencari makanan. Algoritma PSO akan digunakan untuk mencari parameter terbaik dari kontroler PI.

1.2 Rumusan Masalah

Untuk mengendalikan kecepatan motor DC, umumnya digunakan kontroler PI untuk mengendalikan tegangan input ke motor. Tetapi, metode *tuning* PI konvensional tidak menghasilkan respon kecepatan yang terkendali dalam sistem motor DC nonlinier. Maka, permasalahan di sini adalah bagaimana cara *tuning* kontroler PI dengan baik pada sistem motor DC non-linier.

Pengendalian kecepatan motor DC memerlukan sinyal *feedback* kecepatan itu sendiri untuk dibandingkan dengan kecepatan referensi dan menghitung nilai *error*. Kecepatan *feedback* tersebut umumnya menggunakan kecepatan sensor sebagai pengukur. Tetapi, khususnya pada aplikasi industri, sensor kecepatan dapat memiliki biaya tinggi dan menyebabkan *error* pengukuran. Maka, permasalahan di sini adalah bagaimana cara mengestimasi atau menghitung kecepatan *feedback* tanpa menggunakan sensor kecepatan.

1.3 Batasan Masalah

Batasan masalah yang perlu diperhatikan dalam penelitian ini adalah:

1. Sistem disimulasikan dalam MATLAB dan Simulink
2. Diasumsikan noise proses dan pengukuran memiliki distribusi Gaussian.
3. Spesifikasi motor DC yang digunakan
4. *Setpoint* motor DC yang digunakan

1.4 Tujuan

Tujuan dari penelitian ini adalah menerapkan algoritma PSO dengan fungsi objektif termodifikasi dan EKF untuk mengendalikan dan mengestimasi kecepatan motor DC dalam simulasi. Performa PSO-PI dan EKF akan diukur dengan melihat nilai overshoot, rise time, settling time, *Integral of Time Multiplied by Absolute Error* (ITAE) dan *root mean squared error* (RMSE) antara kecepatan estimasi dan kecepatan aktual. Diharapkan hasil simulasi memiliki *overshoot*, *rise time*, dan *RMSE steady state* yang kecil.

1.5 Manfaat

Hasil dari penelitian ini diharapkan dapat menunjang perkembangan dalam pengendalian kecepatan motor DC, serta sebagai referensi untuk penelitian selanjutnya mengenai topik tersebut.

BAB 2 TINJAUAN PUSTAKA

Pada bab ini akan dibahas mengenai hasil penelitian terdahulu yang melatarbelakangi penelitian ini dan tinjauan pustaka yang diperlukan pada penelitian ini.

2.1 Hasil Penelitian Terdahulu

Penelitian mengenai pengendalian kecepatan sensorless motor DC sudah mencakupi banyak metode. (Tripathi et al., 2022) mengendalikan kecepatan motor DC secara *sensorless* menggunakan EKF sebagai estimator dan sebuah fuzzy logic controller atau FLC, dan membandingkannya dengan kontroler PID yang di-*tuning* secara konvensional dan menemukan bahwa penggunaan FLC dapat menurunkan overshoot sebesar 51%, rise time sebesar 36%, dan settling time 64%. (Gundogdu et al., 2020) membandingkan penggunaan algoritma *system identification artificial neural network* atau SI-ANN dengan EKF. SI-ANN memiliki performa sekitar tiga kali lebih bagus walau memerlukan tenaga komputasi yang sedikit lebih tinggi. Umumnya, kontrol dan estimasi kecepatan motor DC tidak digabung menjadi satu sistem.

Penelitian mengenai pengendalian motor DC dengan PSO-PID sudah banyak dilakukan. (Wang et al., 2015) mengendalikan kecepatan motor DC dengan PID yang dioptimisasikan dengan algoritma PSO dan ABC. Model motor DC yang digunakan memiliki fungsi transfer orde 1. Ditemukan bahwa algoritma PSO memiliki respon transien lebih baik dibandingkan algoritma ABC, kecuali untuk *steady state error* yang lebih besar. (Solihin et al., 2011) membandingkan kontrol kecepatan motor DC menggunakan PSO-PID dengan PID yang di-*tuning* dengan metode *Ziegler-Nichols* (ZN). Fungsi transfer dari motor DC adalah orde 2. Ditemukan bahwa PSO-PID menurunkan *overshoot* sebesar 94%, settling time 51%, dan steady state error 21%.

(Zahir et al., 2018) menggunakan GA-PID untuk mengendalikan brushed DC motor. Pada penelitian ini digunakan fungsi objektif yang disesuaikan untuk PID, dimana rise time, settling time, overshoot, dan steady state error digabung dengan *Integral of Time Multiplied by Absolute Error* atau ITAE. Fungsi istimewa ini menurunkan rise time sebesar 76.63%, settling time 78.29%, dan steady state error 41.46%. Modifikasi fungsi objektif seperti ini belum pernah diaplikasikan pada PSO-PID.

Umumnya PSO ditingkatkan performanya dengan memodifikasi parameter pada algoritma PSO-nya sendiri seperti pada (Feng et al., 2021). Pada penelitian tersebut, PSO-PID diaplikasikan pada sistem servo yang mengendalikan *tracking* trayektori. Koefisien inersia dan percepatan dimodifikasi dengan diberi faktor mutasi yang dapat beradaptasi pada tiap iterasi. Ditemukan bahwa dibandingkan PSO-PID konvensional dengan fungsi fitness ITAE, performa tracking PSO-PID termodifikasi 13.18% lebih baik.

2.2 Dasar Teori

Tugas akhir ini mengimplementasikan teori-teori dasar dari motor DC, *Extended Kalman Filter*, PID, dan PSO.

2.2.1 Motor DC

Motor DC adalah motor listrik yang merupakan perangkat elektromekanis yang menggunakan interaksi medan magnet dan konduktor untuk mengubah energi listrik menjadi energi mekanik putar, dimana motor DC dirancang untuk dijalankan dari sumber daya arus

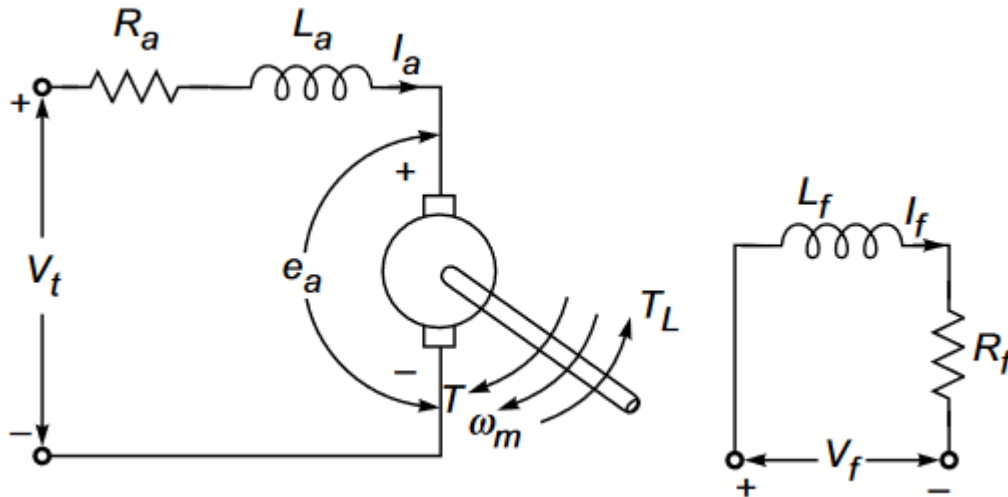
searah (Nise, 2010). Motor DC memiliki stator dengan kutub *salient* yang dieksitasi oleh satu atau lebih lilitan medan. Kumparan *armature* mesin DC berada di rotor, dengan arus dialirkan dari atau ke sana melalui sikat karbon yang bersentuhan dengan segmen komutator tembaga (El-Hawary, 2002). Motor DC jenis sumber daya terpisah atau *separately excited* yang digunakan pada penelitian ini, menggunakan sumber arus listrik untuk kumparan medan terpisah dengan sumber arus listrik untuk kumparan *armature* pada rotor.

2.2.1.1 Model Matematika Motor DC

Model mesin tersebut terdiri dari persamaan listrik dan mekanis. Persamaan listrik motor DC diperoleh dari Hukum Kirchoff kedua yang diaplikasikan pada rangkaian motor DC *separately excited* yang dapat dilihat pada Gambar 1. Persamaan listriknya sebagai berikut

$$v(t) = e_a(t) + R_a i_a(t) + L_a \frac{d}{dt} i_a(t) \quad (2.1)$$

dimana $v(t)$ adalah tegangan suplai armature, $e_a(t)$ adalah back emf, R_a adalah resistansi armature, $i_a(t)$ adalah arus armature, dan L_a adalah induktansi armature.

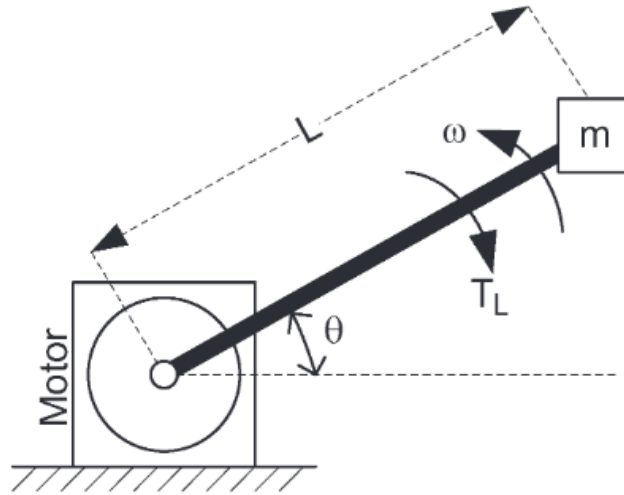


Gambar 2.1. Skematik motor DC *separately excited*

Motor DC *separately excited* memerlukan sumber tegangan DC tambahan V_f untuk mensuplai medan magnetiknya. L_f adalah induktansi kumparan medan dan R_f adalah resistansi kumparan medan. Ketika tegangan kumparan medan V_f konstan hingga tercapai steady-state, i_f yaitu arus kumparan medan juga konstan. Maka, back emf dapat diturunkan untuk mengubah persamaan (1) menjadi

$$v(t) = K_e i_f(t) \omega_m + R_a i_a(t) + L_a \frac{d}{dt} i_a(t) \quad (2.2)$$

dimana K_e adalah konstan back emf dan ω_m adalah kecepatan motor.



Gambar 2.2. Model beban nonlinear motor DC

Persamaan dinamis sistem mekanis motor DC *separately excited* sebagai berikut

$$T(t) = K_t i_a(t) \quad (2.3)$$

$$T(t) = J \frac{d}{dt} \omega_m(t) + T_L(t) + T_f(t) + D \omega_m(t) \quad (2.4)$$

dimana $T(t)$ adalah torsi motor, K_t adalah konstan torsi, J adalah momen inersia motor dan beban, $T_L(t)$ adalah torsi beban, $T_f(t)$ adalah torsi gesekan coulomb, dan D adalah koefisien damping viscous (Tripathi et al., 2022). Berdasarkan Gambar 2, torsi beban T_L dapat diturunkan menjadi

$$T_L = mgL \cos \theta + mL^2 \frac{d}{dt} \omega_m(t) \quad (2.5)$$

dimana m adalah massa beban, g adalah konstan gravitasi, L adalah panjang lengan, dan θ adalah percepatan sudut. θ diperoleh dari integral ω_m . Model state space dari motor DC dapat dirumuskan dari persamaan (2), (3), (4) dan (5) dengan mencari persamaan untuk dua variabel state yaitu arus armature i_a dan kecepatan motor ω_m .

$$\begin{aligned} \dot{\omega}_m &= \frac{T(t) - T_L(t) - T_f(t) - D \omega_m(t)}{J} \\ i_a &= \frac{v(t) - K_e i_f(t) \omega_m - R_a i_a(t)}{L_a} \end{aligned} \quad (2.6)$$

Dari persamaan (2.6) dapat diperoleh persamaan state space motor DC sebagai berikut:

$$\begin{bmatrix} \dot{\omega}_m \\ \dot{i}_a \end{bmatrix} = \begin{bmatrix} -\frac{D}{mL^2 + J} & \frac{K_t}{mL^2 + J} \\ -\frac{K_e}{L_a} & -\frac{R_a}{L_a} \end{bmatrix} \begin{bmatrix} \omega_m \\ i_a \end{bmatrix} + \begin{bmatrix} \frac{-mgL \cos \theta - T_f}{mL^2 + J} & 0 \\ 0 & \frac{1}{L_a} \end{bmatrix} \begin{bmatrix} 1 \\ v_t \end{bmatrix} \quad (2.7)$$

dimana perhitungan θ menggunakan ω_m menyebabkan sistem menjadi nonlinier. Model diskrit dari persamaan state space dari sistem dapat ditulis sebagai berikut:

$$x_{k+1} = A_d x_k + B_d u_k + v_k \quad (2.8)$$

$$y_k = C_d x_k + w_k \quad (2.9)$$

dimana $x = [\omega, i_a]^T$ adalah vektor state, $y = [0, i_a]^T$ adalah vektor output, $u = [1, v_t]^T$ adalah vektor input. v dan w adalah noise proses dan pengukuran dengan kovarians Q dan R . A_d adalah matriks sistem A dalam bentuk diskrit, B_d adalah matriks input B dalam bentuk diskrit, dan C_d adalah matriks transformasi C dalam bentuk diskrit. Ketiga matriks tersebut didiskritisasi pada waktu sampling T_s dalam persamaan berikut:

$$A_d = I + AT_s \quad (2.10)$$

$$B_d \approx BT_s \quad (2.11)$$

$$C_d = C \quad (2.12)$$

(Aydogmus et al., 2012).

2.2.2 Extended Kalman Filter

Extended Kalman Filter atau EKF adalah algoritma matematika yang dapat digunakan untuk mengestimasi variabel state yang umumnya tidak bisa diukur dengan menggunakan variabel lain yang terukur dan statistik *noise*. EKF merupakan observer stokastik untuk estimasi state sistem dinamik nonlinier secara real-time menggunakan sinyal *noise* yang terukur. *Noise* tersebut berkorelasi dengan ketidakpastian pengukuran dan pembuatan model.

Algoritma EKF terdiri dari dua tahap utama, yaitu *prediction* dan *update*. Pada tahap *prediction*, model matematis dari sistem yang mengandung estimasi sebelumnya digunakan dan pada tahap *update*, sebuah skema koreksi *feedback* diaplikasikan pada state yang telah diprediksi. Skema koreksi *feedback* tersebut memiliki variabel tambahan yang mengandung perbedaan *weighted* antara sinyal pengukuran dan estimasi. Variabel ini disebut *Kalman Gain*. *Kalman Gain* akan menentukan seberapa signifikan pengaruh sinyal pada perhitungan estimasi. (Aydogmus et al., 2012).

Dengan persamaan state space (2.8) dan (2.9), langkah prediksi EKF adalah sebagai berikut:

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_k) \quad (2.13)$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q \quad (2.14)$$

dimana \hat{x} adalah state estimasi dari variabel state input x_k , u_k adalah vektor input, P_k dan Q merupakan matriks kovarians prediksi dan pengukuran, dan F_k adalah matriks *jacobian* transisi state. Langkah update EKF adalah sebagai berikut:

$$S_k = H_k P_{k|k-1} H_k^T + R \quad (2.15)$$

$$K_k = P_{k|k-1} H_k^T S_k^{-1} \quad (2.16)$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (y_{k+1} - H_k \hat{x}_{k|k-1}) \quad (2.17)$$

$$P_{k|k} = P_{k|k-1} (I - K_k H_k) \quad (2.18)$$

dimana S_{k+1} adalah kovarians inovasi, K_{k+1} adalah *Kalman Gain*, I adalah matriks identitas, H_k adalah matriks *jacobian* observasi, R merupakan matriks kovarians proses, dan y_{k+1} adalah vektor output. Matriks *jacobian* F_k dan H_k didefinisikan sebagai turunan parsial seperti berikut:

$$F_k = \frac{\partial f(\hat{x}_k, u_k)}{\partial x} \quad (2.19)$$

$$H_k = \frac{\partial h(\hat{x}_{k|k-1})}{\partial x} \quad (2.20)$$

Persamaan (2.13) – (2.18) merupakan satu siklus algoritma EKF (Terzic et al., 2001).

2.2.3 PID

Kontroler *Proportional-Integral-Derivative* atau PID merupakan salah satu kontroler yang paling banyak digunakan dalam industri. PID memiliki performa kontrol yang baik dengan konfigurasi yang mudah dilakukan. Persamaan PID dapat ditulis sebagai berikut

$$u(t) = K_p e(t) + K_i \int_0^t e(t) dt + K_d \frac{de(t)}{dt} \quad (2.21)$$

dimana $u(t)$ adalah output kontroler, $e(t)$ adalah nilai error, K_p adalah gain *proportional*, K_i adalah gain *integral*, dan K_d adalah gain *derivative*. Untuk menentukan nilai gain terbaik diperlukan penyetelan atau *tuning* terlebih dahulu. Terdapat berbagai metode *tuning* PID, seperti *Ziegler-Nichols* atau *Cohen-Coon* (El-Deen et al., 2015). Berikut ini adalah karakteristik dari masing-masing nilai kontroler:

Tabel 2.1 Karakteristik kontroler PID

Gain	Rise Time	Settling Time	Overshoot	Error Steady State
K_p	Berkurang	Sedikit perubahan	Bertambah	Berkurang
K_i	Berkurang	Bertambah	Bertambah	Berkurang
K_d	Sedikit berkurang	Berkurang	Berkurang	Sedikit perubahan

2.2.4 Particle Swarm Optimization

Particle Swarm Optimization atau PSO adalah algoritma optimasi stokastik berdasarkan model simulasi sosial yang dikembangkan dari konsep dan aturan gerakan bebas, tanpa halangan yang mengatur populasi binatang yang terorganisir secara sosial, seperti kawanan burung dan sekolah ikan. Populasi tersebut disebut sebagai *swarm*, yang didefinisikan sebagai sebuah koleksi partikel yang cenderung berkumpul bersama, dimana setiap partikel bergerak secara acak (Štimac et al., 2015).

Partikel memiliki dua parameter yaitu posisi X dan kecepatan V . Posisi merepresentasikan sebuah solusi sementara kecepatan menunjukkan jarak dan arah dari partikel dalam sistem. Kualitas dari posisi akan ditentukan oleh sebuah fungsi objektif yang menandakan *fitness*,

umumnya dipakai kriteria error. Semakin kecil nilai *fitness*, semakin bagus kualitas posisinya. Posisi terbaik dari setiap partikel tersimpan dalam variabel bernama *best position* (p_{best}) dan posisi terbaik dari semua partikel tersimpan dalam variabel bernama *global best* (g_{best}). Kedua ini ditunjukkan pada persamaan berikut:

$$p_{best}(i, t) = \arg \min [f(X_i(k))], \quad i \in \{1, 2, \dots, N_p\} \quad (2.22)$$

$$g_{best}(t) = \arg \min [f(X_i(k))] \quad (2.23)$$

dimana i adalah index partikel, t adalah nomor iterasi, N_p adalah jumlah total partikel, P adalah posisi dan f adalah fungsi objektif. Jumlah partikel dan iterasi perlu ditentukan sebelum simulasi.

Posisi dan kecepatan di update sesuai dengan persamaan berikut:

$$V_i(t + 1) = wV_i(t) + c_1r_1(p_{best}(i, t) - X_i(t)) + c_2r_2(g_{best}(t) - X_i(t)) \quad (2.24)$$

$$X_i(t + 1) = X_i(t) + V_i(t + 1) \quad (2.25)$$

dimana V dan X menunjukkan kecepatan dan posisi, w adalah koefisien inersia yang digunakan untuk mengendalikan eksplorasi global dan lokal, c adalah koefisien percepatan yang positif, dan r adalah faktor acak yang terdistribusi pada *range* $[0, 1]$ (Zhang et al., 2015). Koefisien percepatan akan ditetapkan sebagai konstan. Nilai w berkurang setiap iterasi agar partikel dapat memindahkan fokus pencarian dari global menjadi lokal. Nilai maksimum dan minimum w perlu ditentukan sebelum simulasi dimulai. Pengurangan w ditunjukkan oleh persamaan berikut:

$$w = w_{max} - t \left(\frac{w_{max} - w_{min}}{t_{max}} \right) \quad (2.26)$$

Nilai *upper bound* dan *lower bound* yaitu nilai maksimum dan minimum dari variabel yang dioptimisasi juga dipilih. *Bound* yang terpilih akan digunakan membatasi kecepatan V dan posisi X , serta menginisialisasi posisi X (Wang et al., 2015). Ini ditunjukkan oleh persamaan berikut:

$$V_{max} = 0.2(ub - lb) \quad (2.27)$$

$$V_{min} = -V_{max} \quad (2.28)$$

Sebagai fungsi objektif, umumnya *Integral of Time multiplied by Absolute Error* (ITAE) digunakan sebagai kriteria error. Rumus dari ITAE sebagai berikut:

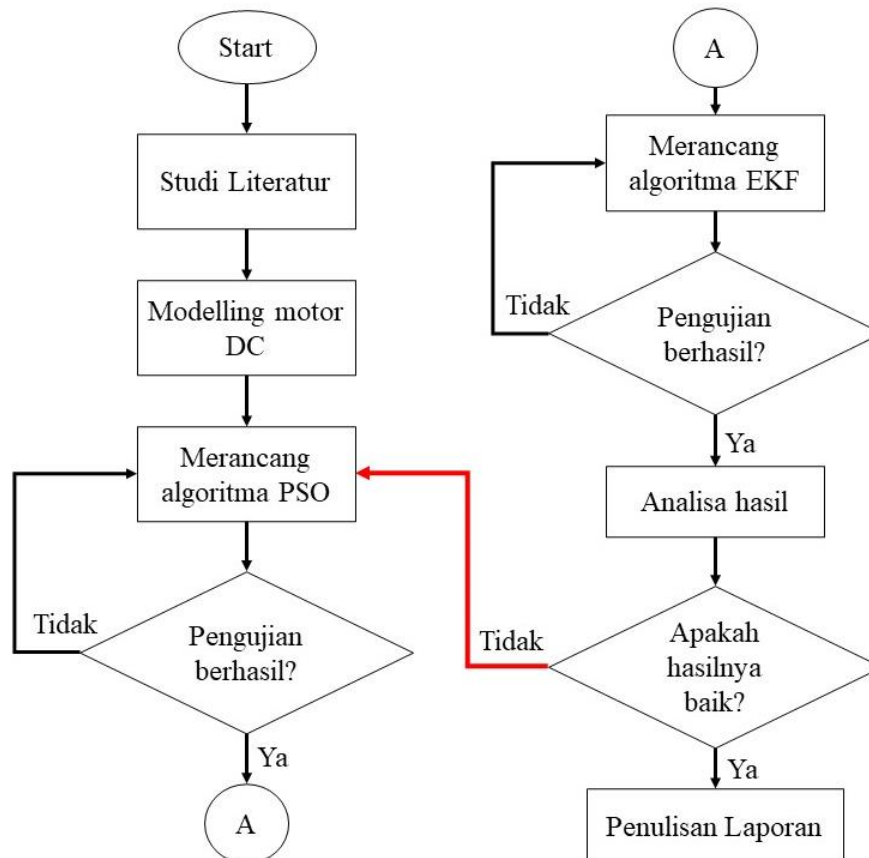
$$ITAE = \int_0^{\tau} t |e(t)| dt \quad (2.29)$$

dimana $e(t)$ adalah error dan t adalah waktu.

BAB 3 METODOLOGI

Pada bab ini akan dibahas mengenai urutan atau serangkaian proses dalam melakukan penelitian ini, bahan dan alat yang diperlukan pada penelitian ini dan metode yang digunakan pada penelitian ini.

3.1 Metode yang digunakan



Gambar 3.1 Diagram Alur Penelitian

Pada gambar 3.1 dapat dilihat alur penelitian untuk tugas akhir ini. Pertama, dimulai dengan mempelajari literatur yang terkait dengan kontrol motor menggunakan kontroler PID yang telah dimodifikasi, dan berbagai teknik estimasi kecepatan motor. Selain itu, dipelajari lebih lanjut metode PSO dan EKF. Lalu, sistem akan dirancang mulai dari pembuatan model motor DC sampai perancangan algoritma PSO dan EKF yang langsung diuji pada model motor DC tersebut. Jika hasil pengujian tidak memuaskan, maka algoritma PSO dan EKF akan dirancang kembali, khususnya pada fungsi objektif PSO dan matriks kovarians EKF. Pada tahap terakhir laporan tugas akhir akan disusun.

3.2 Bahan dan peralatan yang digunakan

Pada tugas akhir ini alat dan bahan yang digunakan adalah:

1. Komputer untuk menjalankan aplikasi dan menulis laporan.
2. Aplikasi MATLAB dan Simulink untuk melakukan simulasi

3.3 Desain Eksperimen

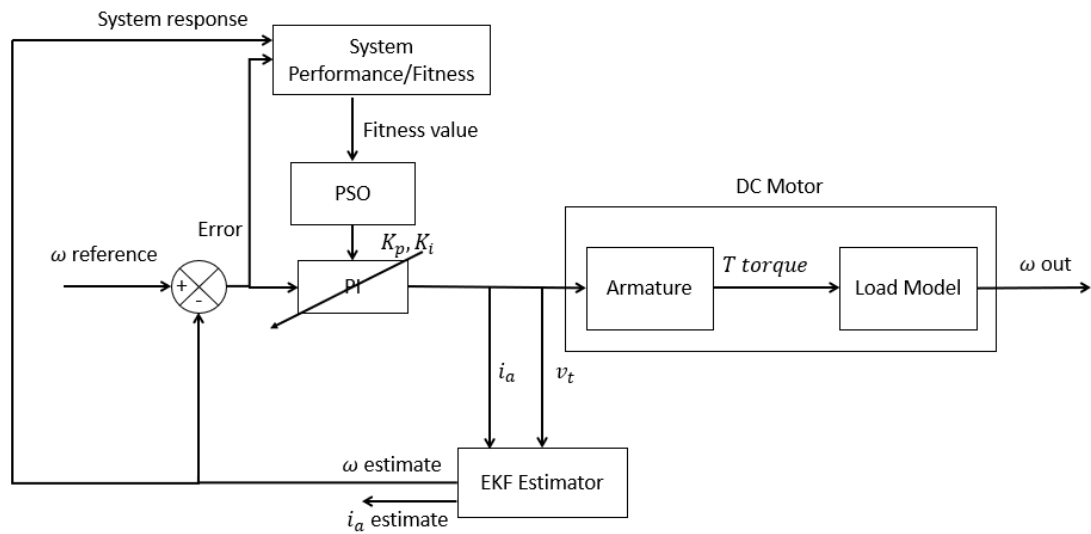
Pada penelitian tugas akhir ini akan melakukan serangkaian percobaan untuk mengetahui bagaimana performa dari sistem yang telah dibuat. Dalam mengetahui performa tersebut maka dirancang sebuah desain eksperimen dalam melakukan penelitian tugas akhir ini seperti pada Tabel 3.1.

Tabel 3.1 Desain Eksperimen

Parameter	Keterangan
<i>Research Gap</i>	<ul style="list-style-type: none"> - Pada penelitian sebelumnya telah digunakan EKF untuk mengestimasi kecepatan motor DC dan PI-PSO untuk mengendalikannya secara terpisah, tetapi belum ada yang menggabungkan keduanya dalam satu sistem. Selain itu, algoritma PI-PSO yang menggunakan fungsi objektif termodifikasi belum pernah diaplikasikan pada motor DC.
Hipotesis	<ul style="list-style-type: none"> - Dengan menggunakan EKF diharapkan kecepatan motor dapat diestimasi dengan nilai <i>error</i> dengan kecepatan aktual yang kecil. Selain itu, diharapkan penggunaan algoritma PSO dengan fungsi objektif yang mempertimbangkan respon transien akan memiliki hasil yang lebih baik dibandingkan PSO yang hanya menggunakan kriteria <i>error</i>.
Variabel Dependen	<ul style="list-style-type: none"> - Parameter kontroler PI - Nilai <i>fitness</i> partikel - Waktu komputasi - Kriteria <i>error</i> (RMSE dan ITAE) - Respon transien
Variabel Independen	<ul style="list-style-type: none"> - Jumlah populasi partikel - Batas tertinggi pencarian (<i>upper bound</i>) - Nilai dan jenis <i>setpoint</i> kecepatan referensi
Variabel Kontrol	<p>Untuk algoritma PSO:</p> <ul style="list-style-type: none"> - Jumlah iterasi - Jumlah variabel - Nilai <i>lower bound</i> - Inersia partikel - Koefisien percepatan - Bobot variabel respon transien pada fungsi objektif <p>Untuk algoritma EKF:</p> <ul style="list-style-type: none"> - Kovarians pengukuran - Kovarians proses

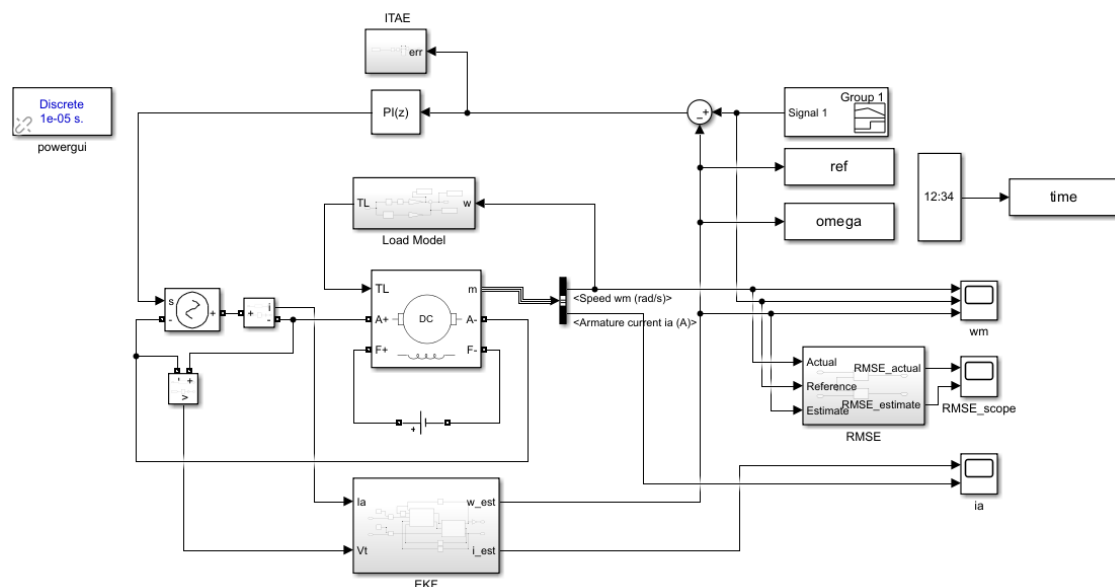
3.4 Urutan pelaksanaan penelitian

Urutan pelaksanaan penelitian tugas akhir ini dimulai dari membuat model motor DC, perancangan PSO-PI, dan perancangan estimator EKF. Secara keseluruhan, sistem dapat direpresentasikan oleh diagram blok berikut:



Gambar 3.2 Diagram blok sistem estimasi dan kontrol kecepatan motor DC dengan EKF dan PI-PSO

Sistem merupakan *close-loop* dengan input kecepatan referensi dan output kecepatan motor. Kecepatan motor diestimasi dengan pengukuran tegangan dan arus *armature* motor DC, yang menjadi input EKF. Kemudian EKF akan menghasilkan output estimasi arus, estimasi kecepatan yang akan digunakan sebagai sinyal *feedback* bagi kontroler PI, serta karakteristik respon sistem yang akan digunakan untuk fungsi objektif dari PSO. Kontroler PI akan di-tuning oleh algoritma PSO yang mempertimbangkan respon sistem dan error antara kecepatan referensi dan estimasi. Kontroler PI akan mengeluarkan sinyal tegangan yang akan menjadi input bagi *plant* motor DC yang berbeban nonlinier.



Gambar 3.3 Model sistem estimasi dan kontrol kecepatan motor DC dengan EKF dan PI-PSO di Simulink

3.4.1 Modelling motor DC

Pada tugas akhir ini motor DC yang digunakan adalah motor DC *separately excited* dengan parameter yang dapat dilihat pada tabel 3.1. Parameter tersebut diperoleh dari referensi (Tripathi et al., 2022). Parameter dari beban yang digunakan pada simulasi juga ditambahkan pada tabel.

Tabel 3.2 Parameter model motor DC

Parameter	Simbol	Nilai
Tegangan <i>armature</i> (V)	V_{arm}	240
Tegangan <i>field</i> (V)	V_{field}	300
Resistansi <i>armature</i> (Ω)	R_a	2.581
Induktansi <i>armature</i> (H)	L_a	0.028
Induktansi <i>mutual armature-field</i> (H)	L_{af}	0.9483
Inersia <i>shaft</i> (kg/m^2)	J	0.02215
Koefisien gesek (Nms)	D	0.002953
Torsi gesekan coulomb (Nm)	T_f	0.5161
Konstanta torsi motor (Nm/A)	K_t	1.79
Konstanta <i>back emf</i> (Vs/r)	K_e	1.79
Massa beban (kg)	m	5
Panjang <i>armature</i> (m)	L	0.05
Konstanta gravitasi (m/s^2)	g	9.81

3.4.1.1 Model matematika motor DC

Dengan menggunakan nilai parameter diatas dan memasukkannya ke persamaan (2.7), dapat diperoleh persamaan state space sebagai berikut:

$$\begin{bmatrix} \dot{\omega}_m \\ \dot{i}_a \end{bmatrix} = \begin{bmatrix} -\frac{D}{mL^2+J} & \frac{K_t}{mL^2+J} \\ -\frac{K_e}{L_a} & -\frac{R_a}{L_a} \end{bmatrix} \begin{bmatrix} \omega_m \\ i_a \end{bmatrix} + \begin{bmatrix} \frac{-mgL \cos \theta - T_f}{mL^2+J} & 0 \\ 0 & \frac{1}{L_a} \end{bmatrix} \begin{bmatrix} 1 \\ v_t \end{bmatrix}$$

$$\begin{bmatrix} \dot{\omega}_m \\ \dot{i}_a \end{bmatrix} = \begin{bmatrix} -0.0852 & 51.6595 \\ -63.9286 & -92.1786 \end{bmatrix} \begin{bmatrix} \omega_m \\ i_a \end{bmatrix} + \begin{bmatrix} -70.677 \cos \theta - 14.873 & 0 \\ 0 & 35.7143 \end{bmatrix} \begin{bmatrix} 1 \\ v_t \end{bmatrix} \quad (3.1)$$

Untuk mengimplementasikan model secara digital, persamaan (3.1) perlu diubah menjadi diskrit. Matriks koefisien A, B, dan C perlu diubah menjadi diskrit menggunakan persamaan (2.8) – (2.12). Dengan memilih waktu sampling $T_s = 10^{-5}$ bentuk diskrit matriks dapat dihitung menjadi:

$$A_d = I + AT_s = \begin{bmatrix} 1 & 0.0005 \\ -0.0006 & -0.9991 \end{bmatrix} \quad (3.2)$$

$$B_d \approx B(k)T_s = \begin{bmatrix} -7.0677e^{-4} \cos \theta - 1.4873e^{-4} & 0 \\ 0 & 3.57143e^{-4} \end{bmatrix} \quad (3.3)$$

$$C_d = C(k) = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \quad (3.4)$$

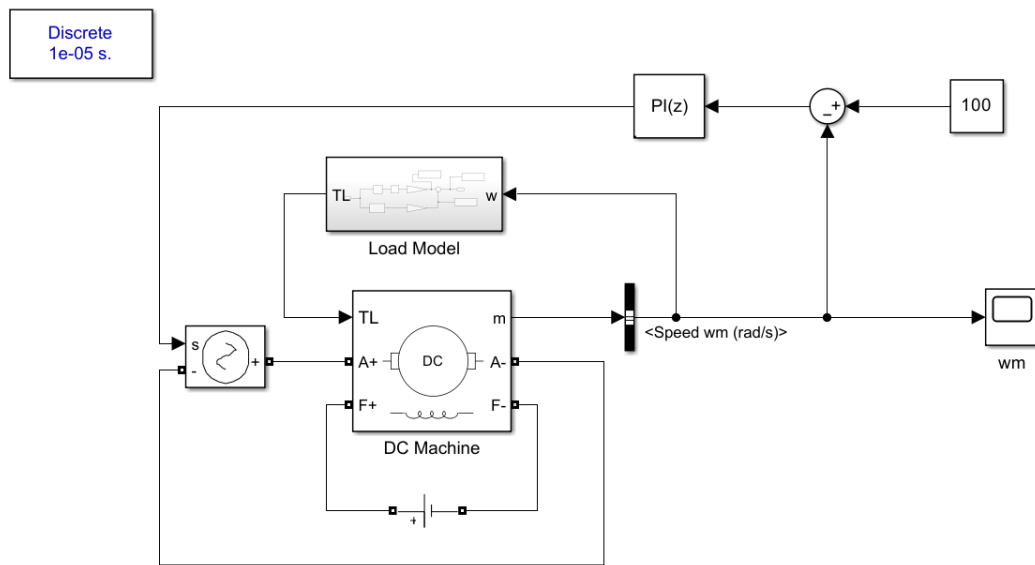
dengan persamaan state space diskrit:

$$x_{k+1} = A_d x_k + B_d u_k + v_k \quad (3.5)$$

$$y_k = C_d x_k + w_k \quad (3.6)$$

3.4.1.1 Pemodelan di Simulink

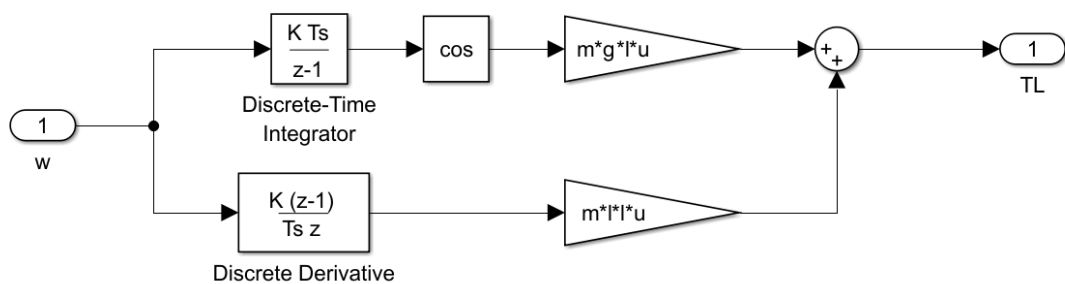
Setelah diketahui parameter motor DC dapat dibuat simulasi pada Simulink seperti pada gambar 3.1. Parameter tersebut dimasukkan pada blok *DC machine*.



Gambar 3.4 Model motor DC di Simulink

Motor DC dikendalikan dengan kontroler PI yang menerima error antara referensi dan output, yang kemudian mengeluarkan output tegangan menuju *armature*. Model dari beban berada pada subsistem *load model* yang dapat dilihat pada gambar 3.2. Model beban yang digunakan berasal dari persamaan (3.7), dimana nilai θ diperoleh dari integral ω_m . Perhitungan nilai $\cos \theta$ menyebabkan beban bersifat nonlinier.

$$T_L = mgL \cos \theta + mL^2 \frac{d}{dt} \omega_m(t) \quad (3.7)$$



Gambar 3.5 Model beban pada Simulink

3.4.2 Perancangan kontroler PI-PSO

Tabel 3.3 Parameter algoritma PSO

Parameter	Nilai
Jumlah iterasi t	30
Jumlah variabel N_{var}	2
<i>Lower bound</i>	[0 0]
w_{max}, w_{min}	1, 0.1
c_1, c_2	2, 2

Parameter yang terpilih untuk algoritma PSO dapat dilihat pada tabel 3.1. Terdapat 2 jumlah variabel sesuai dengan variabel kontroler PI. Jumlah iterasi ditentukan berdasarkan kecepatan ditemukan g_{best} yang umumnya memerlukan 20-30 iterasi. Nilai *upper bound* dan populasi akan divariasikan untuk menguji algoritma PSO.

3.4.2.1 Perancangan Algoritma PSO

Algoritma PSO yang dibuat pada MATLAB terdiri dari inisialisasi dan update. Dalam inisialisasi dicari nilai awal posisi, kecepatan, p_{best} , g_{best} , V_{min} , dan V_{max} . Posisi diinisialisasikan sebagai persamaan (3.8) yang membuat distribusi uniform, dimana $rand(1, N_{var})$ mengenerasi nilai acak untuk variabel PI. Nilai p_{best} dan g_{best} diinisialisasikan sebagai *inf* atau tak terhingga, dimana nilai p_{best} dan g_{best} awal dihitung pada *loop* update. V_{min} dan V_{max} dihitung sesuai dengan persamaan (3.9) dan (3.10).

$$X_1 = (ub - lb)rand(1, N_{var}) + lb \quad (3.8)$$

$$V_{max} = 0.2(ub - lb) \quad (3.9)$$

$$V_{min} = -V_{max} \quad (3.10)$$

Langkah update terdiri dari perhitungan *fitness* posisi X setiap partikel, update p_{best} dan g_{best} , update X, update V, dan output g_{best} . Pada kode di MATLAB, update p_{best} dan g_{best} dilakukan terlebih dahulu sebagai lanjutan proses inisialisasi. Lalu, sebelum X dan V dapat diupdate nilai w juga perlu diperbarui setiap iterasi sesuai dengan persamaan (3.11). Untuk menghitung X dan V digunakan persamaan (3.12) dan (3.13).

$$w = w_{max} - t \left(\frac{w_{max} - w_{min}}{t_{max}} \right) \quad (3.11)$$

$$V_i(t + 1) = wV_i(t) + c_1r_1(p_{best}(i, t) - X_i(t)) + c_2r_2(g_{best}(t) - X_i(t)) \quad (2.24)$$

$$X_i(t + 1) = X_i(t) + V_i(t + 1) \quad (2.25)$$

Algoritma PSO dapat ditulis sebagai berikut:

1. Inisialisasi

Untuk setiap partikel $i = 1, 2, \dots, N_p$

- Inisialisasi posisi dengan distribusi *uniform* $X_i(0) \sim U(lb, ub)$
- Inisialisasi p_{best} pada posisi awal
- Inisialisasi g_{best} pada populasi awal
- Inisialisasi $V_i(0)$

2. Update sampai iterasi maksimum atau solusi terbaik

Untuk setiap partikel $i = 1, 2, \dots, N_p$

- Pilih nilai r acak $r_1 r_2 \sim U(0, 1)$
- Update kecepatan V dengan persamaan (2.24)
- Update posisi X dengan persamaan (2.25)
- Jika $f[X_i(t)] < f[p_{best}(i, t)]$
 - Update $p_{best} = X_i(t)$
 - Jika $f[X_i(t)] < f[g_{best}(i, t)]$, update $g_{best} = X_i(t)$
- $t = t + 1$

3. Output g_{best} sebagai solusi terbaik

3.4.2.1 Perancangan Fungsi Objektif

Umumnya PSO menggunakan kriteria error seperti ITAE, IAE, atau ISE sebagai fungsi objektif. Tujuannya adalah meminimalkan error untuk memperoleh hasil terbaik. Tetapi, fungsi objektif tersebut kurang cocok untuk mendesain kontroler PI dengan baik. Maka, pada penelitian ini akan digunakan fungsi objektif yang mempertimbangkan respon sistem yaitu *rise time*, *overshoot*, *settling time*, *steady state error*, serta ITAE. Fungsi objektif tersebut dapat meningkatkan kualitas respon sistem dibanding fungsi ITAE saja. Fungsi objektif tersebut adalah:

$$f(K_p, K_i) = \mu \int_0^{\tau} t |e(t)| dt + \alpha O_v + \beta SS_e + \delta t_s + \gamma t_r$$

dimana O_v, SS_e, t_s, t_r adalah *overshoot*, *steady state error*, *settling time*, dan *rise time*. $\mu, \alpha, \beta, \delta, \gamma$ merupakan bobot dari masing-masing variabel yang bernilai 5, 0.8, 1, 5, 50. Pembobotan yang terpilih berdasarkan respon yang diinginkan, yaitu *overshoot* kecil, *error* minimum, *rise time* cepat, dan *settling time* yang lebih pelan (Zahir et al., 2018). Nilai pembobotannya dicari dengan tuning manual. Besar dari bobot tersebut tidak sepenuhnya mencerminkan kepentingan variabel tersebut. *Rise time* memiliki bobot sangat besar karena nilai *rise time* sangat kecil. *Overshoot* memiliki bobot yang kecil karena nilai *overshoot* sangat besar. Pada sistem nonlinier ini, nilai *steady state error* tidak akurat karena sinyal respon terus berosilasi. Maka, untuk meminimalkan error, nilai ITAE lebih berbobot dibandingkan *steady state error*. Pengaruh tiap variabel akan diuji pada bab 4 dengan melihat komposisi nilai *fitness*.

3.4.3 Perancangan Estimator EKF

Algoritma *extended kalman filter* ditunjukkan oleh persamaan (3.8) – (3.13) yang terdiri dari langkah prediksi dan update, dengan perhitungan matriks *jacobian* pada persamaan (3.14) dan (3.15) untuk melakukan linierisasi. Persamaan yang digunakan adalah:

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_k) = A_d x_{k-1|k-1} + B_d u_k \quad (3.8)$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q \quad (3.9)$$

$$S_k = H_k P_{k|k-1} H_k^T + R \quad (3.10)$$

$$K_k = P_{k|k-1} H_k^T S_k^{-1} \quad (3.11)$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (y_{k+1} - H_k \hat{x}_{k|k-1}) \quad (3.12)$$

$$P_{k|k} = P_{k|k-1} (I - K_k H_k) \quad (3.13)$$

$$F_k = \frac{\partial f(\hat{x}_k, u_k)}{\partial x} \quad (3.14)$$

$$H_k = \frac{\partial h(\hat{x}_{k|k-1})}{\partial x} \quad (3.15)$$

Untuk mengaplikasikan algoritma EKF, matriks *jacobian* perlu di evaluasi terlebih dahulu. Pada sistem ini Dari persamaan state space (3.5) dan (3.6) ditemukan matriks *jacobian* sebagai berikut:

$$F_k = \frac{\partial}{\partial x} \begin{bmatrix} 1 & 0.0005 \\ -0.0006 & -0.9991 \end{bmatrix} \begin{bmatrix} \omega_m \\ i_a \end{bmatrix} + \begin{bmatrix} -7.0677e^{-4} \cos \theta - 1.4873e^{-4} & 0 \\ 0 & 3.57143e^{-4} \end{bmatrix} \begin{bmatrix} 1 \\ v_t \end{bmatrix}$$

$$F_k = \frac{\partial}{\partial x} \begin{bmatrix} \omega_m + 0.0005 i_a & -7.0677e^{-4} \cos \theta - 1.4873e^{-4} \\ -0.0006 \omega_m - 0.9991 i_a & 3.57143e^{-4} v_t \end{bmatrix}$$

$$F_k = \begin{bmatrix} 1 & 0.0005 \\ -0.0006 & -0.9991 \end{bmatrix} = A_d \quad (3.16)$$

$$H_k = \frac{\partial}{\partial x} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_m \\ i_a \end{bmatrix}$$

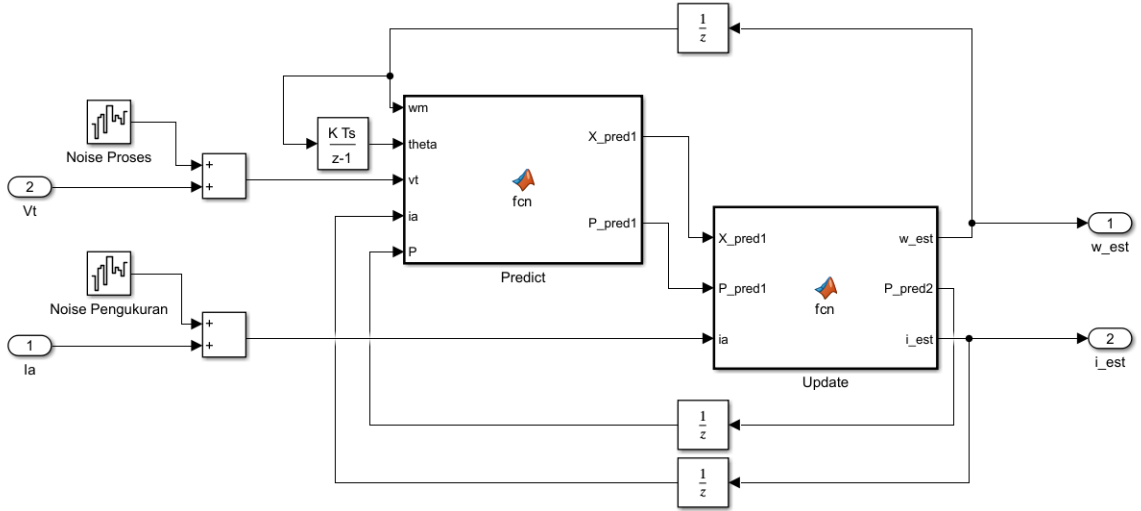
$$H_k = \frac{\partial}{\partial x} \begin{bmatrix} 0 \\ i_a \end{bmatrix}$$

$$H_k = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} = C_d \quad (3.17)$$

dimana F_k bernilai sama dengan A_d , dan H_k bernilai sama dengan C_d . Kedua *jacobian* yang terhitung bersifat konstan, maka pembaruan nilai *jacobian* tidak perlu dilakukan pada setiap state baru. Setelah matriks tersebut ditemukan, nilai dari kovarians Q dan R perlu ditentukan bersama dengan nilai awal kovarians P. Nilai ketiga kovarians akan ditentukan secara manual pada simulasi dengan melihat respon terbaik yang dihasilkan.

3.4.3.1 Perancangan EKF di Simulink

Pada Simulink, algoritma EKF dibangun sebagai subsistem dengan dua blok fungsi yaitu prediksi dan update, sesuai dengan dua langkah EKF yang ada. Kedua input v_t dan i_a diberi noise gaussian zero mean diluar blok fungsi.



Gambar 3.6 Subsistem blok EKF pada Simulink

Pada blok prediksi, dengan persamaan (3.8) nilai prediksi state $\hat{x}_{k|k-1} = [\omega_m, i_a]^T$ dapat dihitung dengan nilai estimasi ω_m dan i_a , input v_t , dan θ yang diperoleh dari melakukan integrasi pada estimasi ω_m . Lalu, kovarians prediksi $P_{k|k-1}$ dihitung sesuai dengan persamaan (3.9) menggunakan nilai estimasi i_a , *jacobian* F_k , kovarians pengukuran Q dan estimasi kovarians $P_{k-1|k-1}$. Matriks $P_{k-1|k-1}$ terinisialisasi dengan fitur *callback function* pada *properties* blok prediksi. Setelah melakukan tuning, nilai awal kovarians $P_{k-1|k-1}$ yang digunakan adalah:

$$P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (3.18)$$

Pada blok update, persamaan (3.10) sampai (3.13) diimplementasikan. Pertama, nilai i_a dan *jacobian* H_k digunakan untuk vektor observasi y_k . Lalu, kovarians inovasi S_k dicari dengan menggunakan kovarians prediksi $P_{k|k-1}$, kovarians proses R , dan *jacobian* H_k . Kemudian nilai gain Kalman K_k dihitung dan digunakan untuk mencari nilai estimasi state $\hat{x}_{k|k}$ dan estimasi kovarians prediksi $P_{k|k}$. Kedua elemen estimasi state, yaitu estimasi kecepatan ω_m dan arus i_a , dikeluarkan sebagai output blok EKF.

Setelah EKF diintegrasikan dalam sistem, dilakukan tuning nilai kovarians pengukuran Q dan kovarians proses R . Nilai Q menunjukkan penyimpangan nilai observasi dari model state-space yang dipakai, dan juga noise dari pengukuran sensor tegangan. Nilai R menunjukkan noise dari pengukuran sensor arus. Dari nilai awal matriks identitas, ditemukan bahwa respon sistem terbaik didapatkan dengan nilai Q dan R sebagai berikut:

$$Q = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \quad (3.19)$$

$$R = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \quad (3.20)$$

Kovarians Q di tuning terlebih dahulu sebelum R . Nilai yang paling stabil untuk Q dan R berada pada range 0.1 sampai 1. Nilai yang terbaik adalah 0.5, sama untuk kedua kovarians. Ini

mungkin disebabkan oleh keakuratan model state space yang baik, maka hanya error pengukuran arus dan tegangan yang berpengaruh secara signifikan.

Hasil estimasi kecepatan dan arus dari EKF akan dibandingkan dengan nilai output motor sebenarnya. Nilai *root mean square error* (RMSE) antara kecepatan estimasi dan kecepatan asli dihitung untuk mengukur performa EKF. RMSE digunakan untuk mengukur perbedaan antara hasil estimasi dan observasi (Gundogdu et al., 2020). Rumus mencari RMSE adalah:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2} \quad (3.21)$$

BAB 4 Hasil dan Pembahasan

Pada bab ini akan dibahas mengenai hasil, pembahasan serta dokumentasi dari serangkaian pengujian yang telah dilakukan pada penelitian ini.

4.1 Pengujian PI-PSO

Algoritma PSO yang menggunakan fungsi objektif termodifikasi dijalankan menggunakan parameter yang tertera di tabel 3.2. Parameter besar populasi dan *upper bound* akan divariasikan untuk menguji algoritma PSO. Nilai *upper bound* dipilih dengan melihat solusi sebelumnya dan mendekati nilainya kepada solusi optimal. Pada pengujian, set point referensi kecepatan yang terpilih adalah 100. Simulasi akan memiliki 30 iterasi dan respon tiap simulasi dijalankan selama 1 detik.

Solusi dengan nilai *fitness* tertinggi dan terkecil akan dibandingkan dengan melihat nilai ITAE dan respon transien. Karakteristik respon yang diinginkan adalah *overshoot* < 10%, *settling time* < 1s, dan *rise time* < 0.1s. Komposisi nilai *fitness* akan dilihat untuk melihat efektivitas pembobotan fungsi objektif termodifikasi. Algoritma PSO yang menggunakan fungsi objektif ITAE juga akan dilihat responnya untuk dibandingkan dengan PSO yang menggunakan fungsi objektif termodifikasi.

4.1.1 Pengujian Fungsi Objektif Termodifikasi

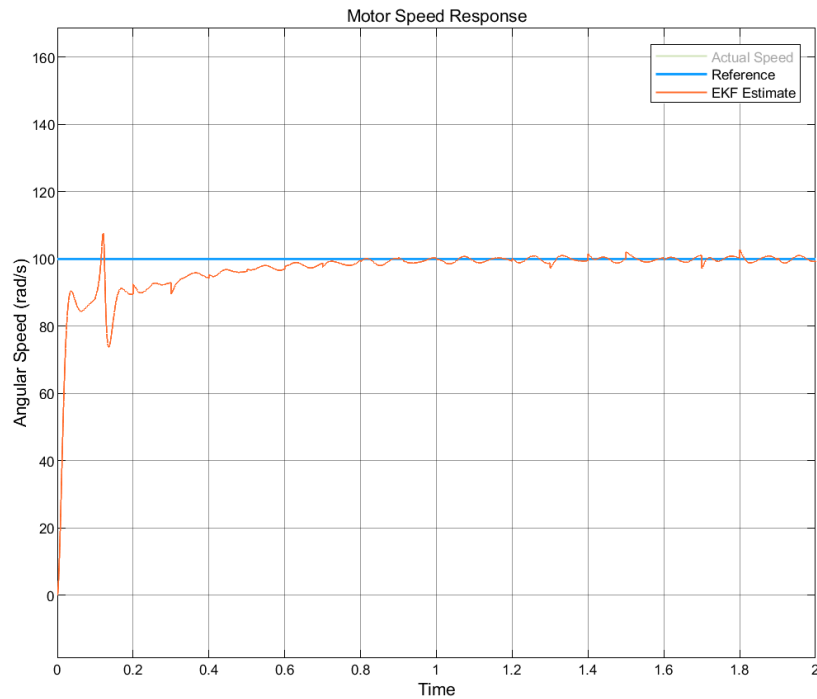
Tabel 4.1 Hasil pengujian PSO dengan Fungsi Objektif Termodifikasi

Skenario	Populasi	Max K_p	Max K_i	Solusi _(K_p, K_i)	<i>Fitness</i> optimal	Waktu komputasi (s)
1	25	100	50	(4.1062, 19.2321)	18.5565	2304.7
2	20	100	50	(4.1474, 18.5197)	18.6863	1858.3
3	25	50	40	(4.0551, 19.3428)	18.1014	2281.2
4	20	50	40	(3.9860, 20.2721)	18.1603	1852.1
5	25	20	30	(3.9432, 20.5340)	18.0107	2290.5
6	20	20	30	(4.0168, 20.0083)	18.0261	1861.0
7	25	15	25	(3.9406, 20.6850)	17.8497	2287.8
8	20	15	25	(3.9437, 20.5848)	17.8568	1846.0
9	25	10	25	(3.9603, 20.3812)	17.9072	2293.4
10	20	10	25	(3.9740, 20.2136)	17.9243	1855.8

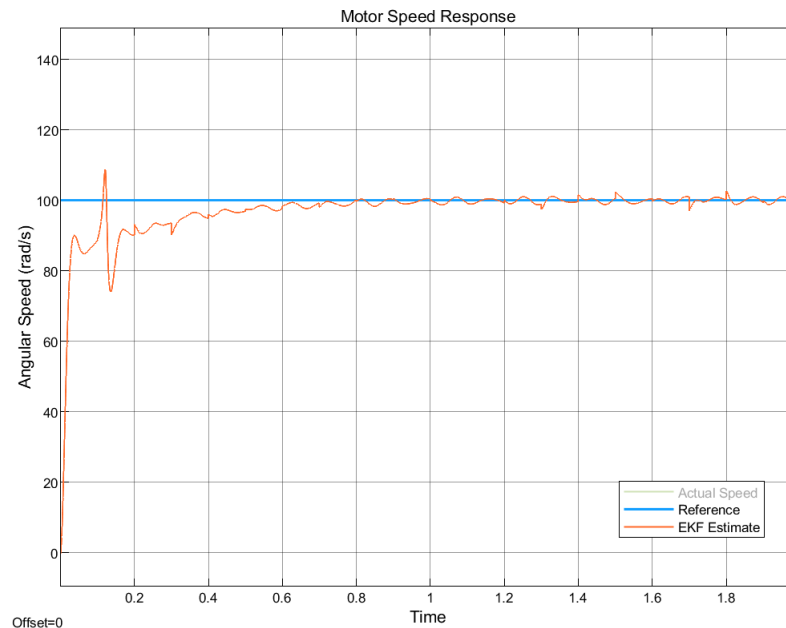
Dari hasil pengujian di tabel 4.1, dapat dilihat bahwa waktu simulasi berkurang seiring dengan jumlah populasi. Terdapat perbedaan 400s antara skenario dengan populasi 25 dan 20. Jumlah populasi yang lebih besar dapat meningkatkan kualitas solusi. Pada setiap skenario dimana parameter ditetapkan dan populasi divariasikan, skenario dengan populasi lebih besar memiliki nilai *fitness* yang lebih kecil.

Ditemukan bahwa nilai *fitness* optimal cenderung lebih baik ketika nilai *upper bound* lebih kecil, terutama pada skenario 3 sampai 10 ketika *upper bound* $K_p \leq 50$. Terdapat perbedaan nilai *fitness* sebesar 0.3246 antara skenario 1 dan 3, dengan $K_p = 100$ dan $K_p = 50$. Sedangkan, perbedaan skenario 3 dengan 7 yang memiliki solusi terbaik hanya sebesar 0.2446. Ini berarti range pencarian $K_p = 100$ terlalu besar. Perbedaan *fitness* tertinggi dan terkecil bernilai 0.8366. Untuk mengetahui seberapa signifikan perbedaan ini, respon transien solusi terbaik dan terburuk akan dibandingkan. Skenario 7 memiliki hasil terbaik dengan populasi 25, *upper*

bound (15, 25), *fitness* optimal 17.8497, dan waktu simulasi 2287.8s. Skenario 2 memiliki hasil terburuk dengan populasi 20, *upper bound* (100, 50), *fitness* optimal 18.6863, dan waktu simulasi 1858.3s.



Gambar 4.1a Respon kecepatan dengan solusi skenario 2



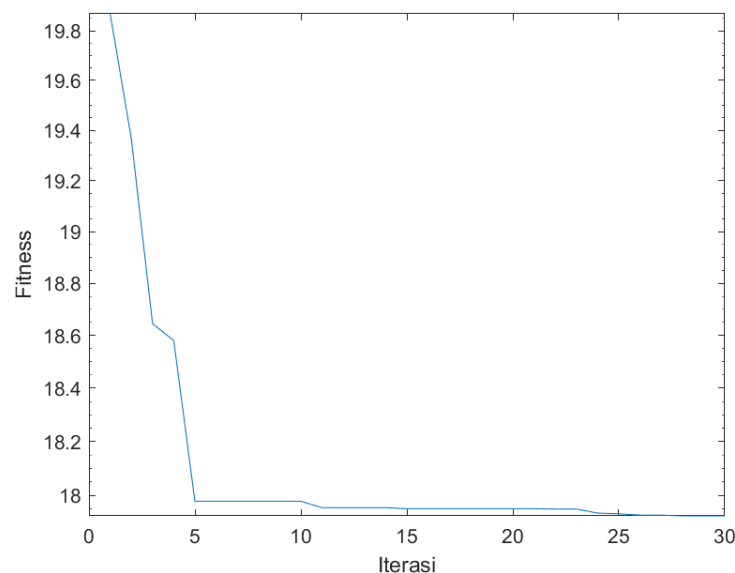
Gambar 4.2b Respon kecepatan dengan solusi skenario 7

Pada gambar 4.1a dan 4.1b ditunjukkan respon kecepatan motor menggunakan solusi skenario 2 dengan fitness terburuk dan skenario 7 dengan fitness terbaik. Dapat dilihat bahwa bentuk sinyal respon kedua skenario sangat mirip. Nilai ITAE pada skenario 2 adalah 1.4237 dan pada skenario 7 adalah 1.1667. Terdapat perbedaan error sebesar 0.2570 yang tidak terlalu signifikan.

Tabel 4.2 menunjukkan respon transien untuk kedua skenario. Skenario 2 memiliki *rise time* yang sedikit lebih kecil dibandingkan skenario 7 dengan perbedaan 0.0029s, dan *overshoot* yang sedikit lebih besar dengan perbedaan 0.0091. *Rise time* lebih kecil dan *overshoot* lebih besar disebabkan oleh parameter K_p yang lebih besar. *Settling time* skenario 2 dan 7 memiliki perbedaan sebesar 0.1806s, dimana skenario 2 memiliki nilai yang lebih besar. Hasil respon transien untuk kedua skenario memenuhi nilai yang diinginkan. Maka dapat disimpulkan bahwa perubahan parameter populasi dan upper bound dalam algoritma PSO dapat meningkatkan kualitas nilai fitness optimal.

Tabel 4.2 Respon transien kecepatan motor untuk skenario 2 dan 7

Parameter	Skenario 2	Skenario 7
Overshoot	8.8374	8.8283
Settling time	0.8589	0.6783
Rise time	0.0283	0.0312



Gambar 4.2 Grafik perubahan *fitness* pada skenario 7

Gambar 4.2 menunjukkan nilai *fitness* terbaik pada tiap iterasi dalam skenario 7. Dapat dilihat bahwa nilai *fitness* memiliki penurunan signifikan sebesar 2.1 dari iterasi pertama sampai kelima. Nilai *fitness* tidak memiliki perubahan signifikan untuk iterasi selanjutnya, terutama pada iterasi 25 sampai 30. Perubahan *fitness* dari iterasi 5 sampai 30 sebesar 0.0677. Solusi terbaik menghasilkan nilai $K_p = 3.9406$ dan $K_i = 20.6850$. Parameter tersebut akan digunakan untuk menguji performa estimator EKF pada bagian berikutnya.

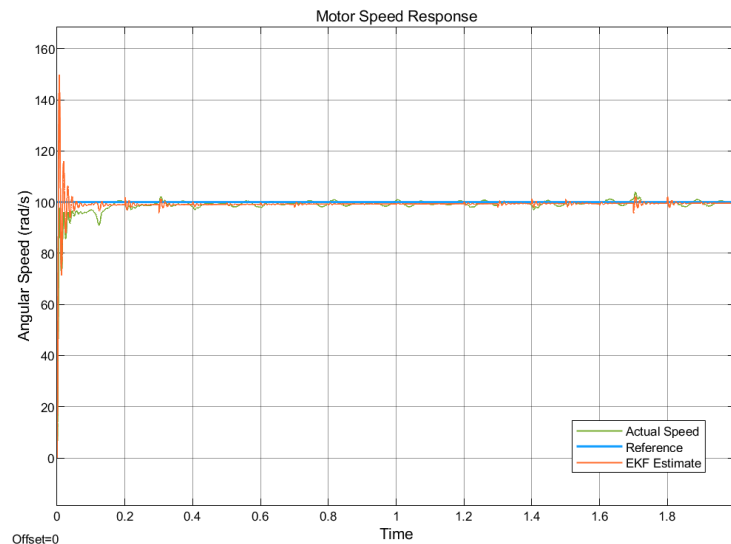
Pada bagian ini akan dilihat pembobotan respon transien pada nilai *fitness* skenario 7. Pada fungsi objektif, ITAE, *overshoot*, *steady state error*, *settling time*, dan *rise time* memiliki bobot yang bernilai 5, 0.8, 1, 5, 50. Tabel 4.3 menunjukkan pembobotan respon transien dan ITAE pada nilai *fitness* skenario terbaik. Dapat dilihat bahwa pengaruh terbesar berada pada *overshoot* dan ITAE, diikuti dengan *settling time* dan *rise time*. *Error steady state* memiliki

pengaruh sangat kecil karena ITAE menggantikannya sebagai kriteria error. Pembobotan tersebut berhasil dalam mengendalikan respon transien dari kecepatan motor.

Tabel 4.3 Pembobotan respon transien pada nilai *fitness* skenario 7

Parameter	Nilai asli	Nilai*bobot	Persentase
ITAE	1.1667	5.8337	32.68%
Overshoot	8.8283	7.0627	39.57%
Error <i>steady state</i>	0.0029	0.0029	0.02%
Settling time	0.6783	3.3913	19.00%
Rise time	0.0312	1.5591	8.73%
Total		17.8497	100%

4.1.2 Pengujian Fungsi Objektif ITAE



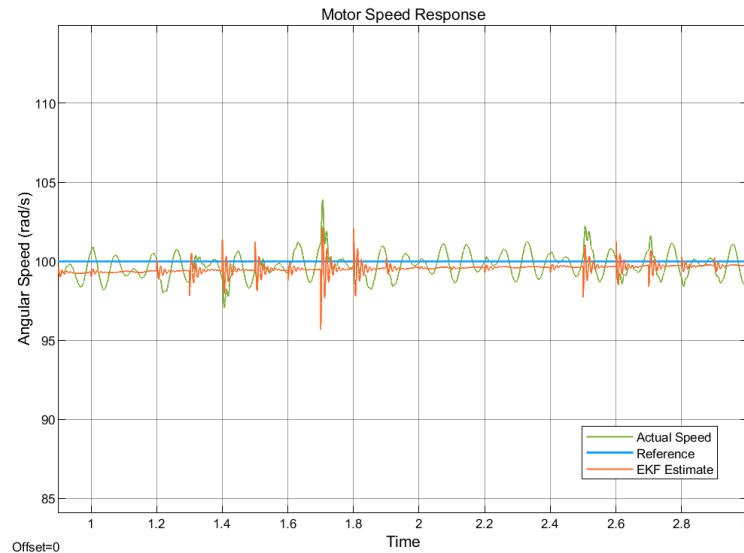
Gambar 4.3a Respon kecepatan PI $K_p = 100$ dan $K_i = 50$

Dengan menggunakan ITAE sebagai fungsi objektif, algoritma PSO yang memiliki *upper bound* maksimum pada pengujian sebelumnya yaitu (100, 50), memiliki solusi yang terbaik. Nilai tiap solusi selalu bernilai sama dengan *upper bound* karena paramter K_p akan selalu dimaksimalkan untuk mengurangi error. Maka, solusi terbaik akan memiliki K_p tertinggi yaitu skenario 1. Mengubah populasi tidak mempengaruhi solusi, seperti pada skenario 3 dan 4.

Tabel 4.4 Hasil pengujian PSO-ITAE

Skenario	Populasi	Max K_p	Max K_i	Solusi _(K_p, K_i)	<i>Fitness</i> optimal (ITAE)
1	25	100	50	(100, 50)	0.4315
2	25	50	40	(50, 40)	0.6096
3	25	20	30	(20, 30)	1.1059
4	20	20	30	(20, 30)	1.1059
5	20	15	25	(15, 25)	1.1941
6	20	10	25	(10, 25)	1.1979

Hasil ITAE dan respon transien pada gambar 4.3a sangat baik dengan ITAE= 0.4315, *rise time* = 0.0027s, dan *settling time* = 0.4057s. Tetapi, *overshoot* dari sistem ketika motor dimulai sangat besar mencapai 49%. Dengan ITAE saja sebagai fungsi objektif, respon transien tidak dapat dikendalikan. Dengan membandingkan respon transien tersebut dengan respon transien pada tabel 4.2, dapat dilihat bahwa respon transien dapat dikendalikan dengan menambahkan respon transien pada fungsi objektif.



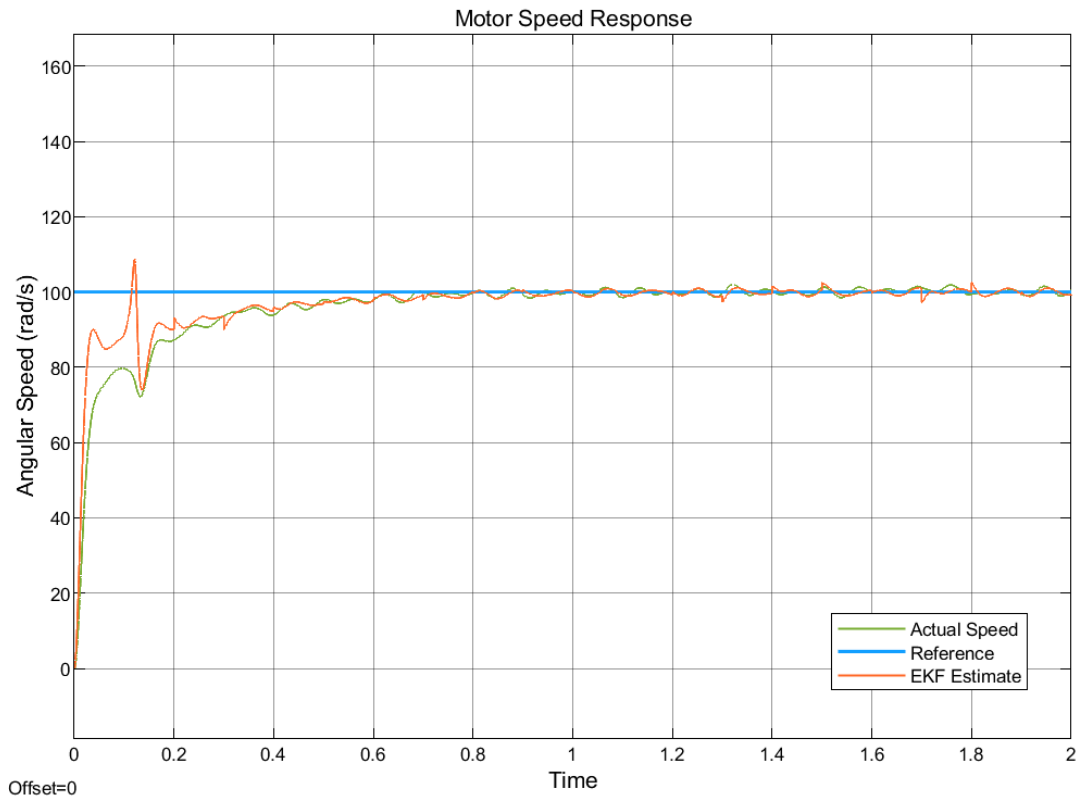
Gambar 4.3b Respon kecepatan PI $K_p = 100$ dan $K_i = 50$ diperbesar

Gambar 4.3b menunjukkan sinyal respon yang diperbesar. Dapat dilihat bahwa bentuk sinyal estimasi tidak sesuai dengan sinyal aktual, terdapat beberapa spike yang berfrekuensi tinggi. Sinyal estimasi juga tidak beresilasi tepat pada kecepatan referensi yaitu 100 rad/s.

4.2 Pengujian Estimator EKF

Estimator EKF akan diuji dengan membandingkan kecepatan estimasi dan aktual. Nilai RMSE kedua output dengan kecepatan referensi juga akan dibandingkan untuk mengukur performa estimator. Nilai arus estimasi dan aktual juga akan dibandingkan. Sistem juga akan diuji dengan kecepatan referensi yang berubah-ubah. Nilai K_p dan K_i optimal dipilih sesuai dengan hasil pengujian PSO yaitu $K_p = 3.9406$ dan $K_i = 20.6850$. RMSE akan menggantikan ITAE sebagai kriteria error bagi sistem untuk mengukur performa estimator.

4.2.1 Pengujian *Setpoint* Konstan



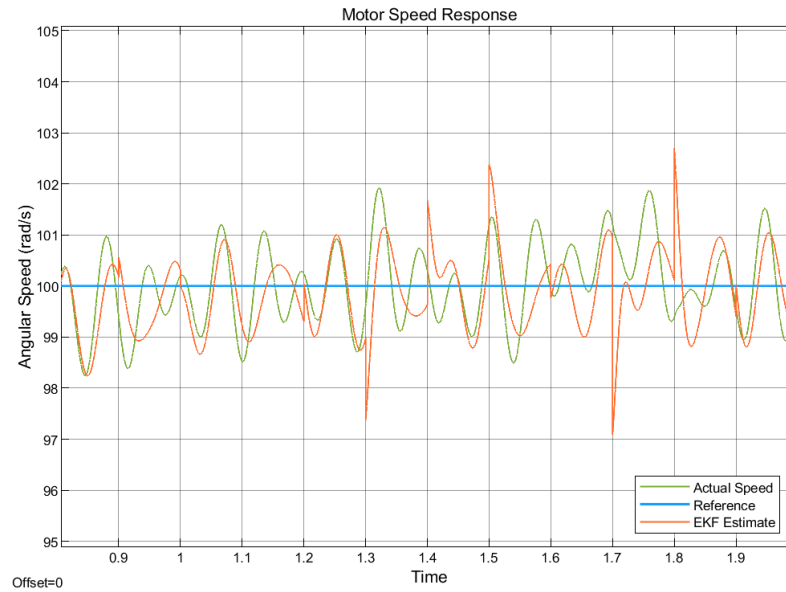
Gambar 4.4a Respon kecepatan dengan *setpoint* konstan $\omega_{ref} = 100 \text{ rad/s}$

Pengujian pertama adalah dengan referensi sinyal konstan 100 rad/s . Simulasi dijalankan selama 2s. Karakteristik respon yang diinginkan adalah *overshoot* $< 10\%$, *settling time* $< 1.5\text{s}$, *rise time* $< 0.1\text{s}$, dan *steady state* yang berosilasi pada 100 rad/s . Gambar 4.4a menunjukkan respon kecepatan dari sistem. Dapat dilihat bahwa bentuk sinyal estimasi menyerupai sinyal aktual. Perbedaan paling signifikan berada pada awal, sinyal estimasi lebih cepat dan tidak stabil dibandingkan sinyal aktual.

Tabel 4.2 menunjukkan karakteristik respon sinyal estimasi dan aktual. Dapat dilihat bahwa kedua sinyal memiliki *overshoot* dibawah 10% , *settling time* dibawah 1.5s , dan *rise time* dibawah 0.1s yang memenuhi karakteristik respon yang diinginkan. Kecepatan aktual memiliki *overshoot* dan *settling time* lebih kecil dibandingkan kecepatan estimasi, dimana perbedaan *overshoot* cukup signifikan sebesar 6.9087 dan *settling time* sebesar 0.2682 . *Rise time* kecepatan aktual lebih lambat dengan perbedaan sebesar 0.1869s . Selain *overshoot*, respon transien kecepatan estimasi cukup mendekati kecepatan aktual.

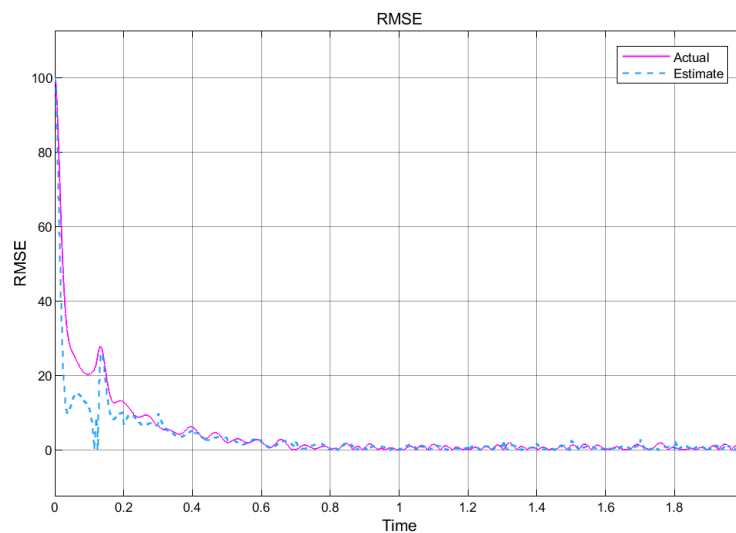
Tabel 4.3 Respon transien sistem dengan *setpoint* konstan $\omega_{ref} = 100$

Parameter	ω Estimasi	ω Aktual
Overshoot	8.8283	1.9196
Settling time	0.6783	0.9465
Rise time	0.0312	0.2181



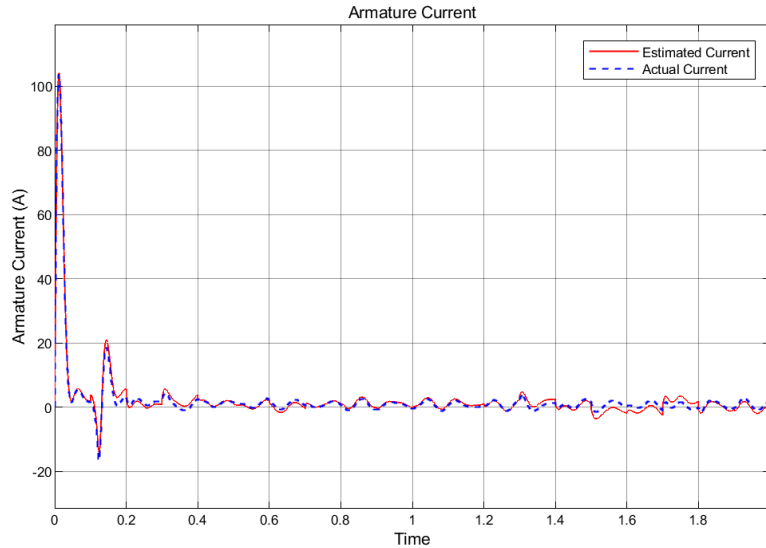
Gambar 4.3b Respon kecepatan dengan $setpoint$ konstan $\omega_{ref} = 100 \text{ rad/s}$ diperbesar

Respon kecepatan diperbesar pada gambar 4.3b dan dapat dilihat bahwa bentuk dari sinyal estimasi menyerupai sinyal aktual. Keduanya berosilasi pada range 98 rad/s sampai 102 rad/s . Dengan melihat *steady state* dari $t = 1 \text{ s}$ sampai $t = 2 \text{ s}$, diperoleh error RMSE bernilai 1.538 untuk sinyal estimasi dan 2.045 untuk sinyal aktual. Ini menunjukkan bahwa kecepatan estimasi memiliki *error* dengan $setpoint$ yang lebih kecil dibandingkan kecepatan aktual.



Gambar 4.4 RMSE kecepatan estimasi dan aktual dengan $setpoint$ konstan $\omega_{ref} = 100 \text{ rad/s}$

Dari grafik RMSE kecepatan di gambar 4.4, dapat dilihat juga bahwa kecepatan estimasi memiliki RMSE yang lebih kecil pada bagian awal dari $t = 0$ sampai $t = 0.2$, dimana respon masih belum stabil.

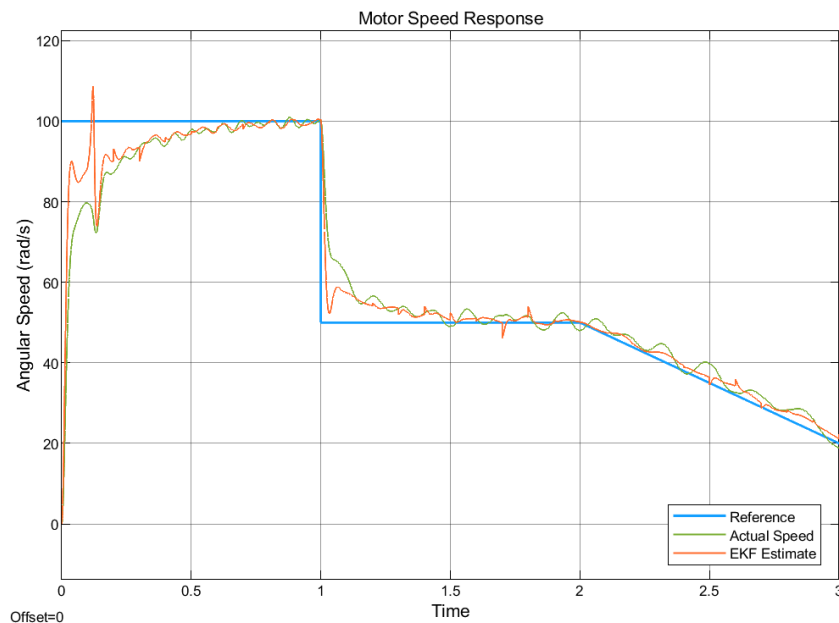


Gambar 4.5 Arus estimasi dan aktual dengan $setpoint$ konstan $\omega_{ref} = 100 \text{ rad/s}$

Pada gambar 4.5 menunjukkan grafik arus estimasi dan aktual. Dapat dilihat bahwa arus estimasi menyerupai bentuk arus aktual, dengan rata-rata $RMSE = 1.086$. Ini menunjukkan bahwa estimasi arus lebih baik dibandingkan estimasi kecepatan.

4.2.2 Pengujian *Setpoint* Berubah-ubah

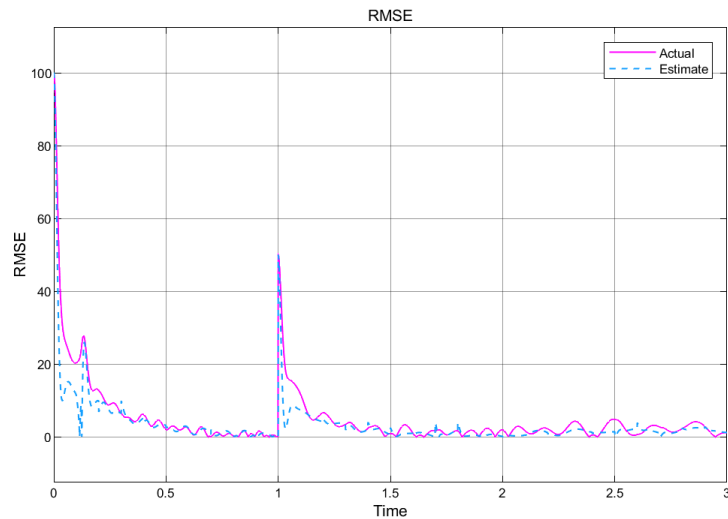
Pada sub-bab ini dilakukan pengujian dengan *setpoint* kecepatan referensi yang berubah-ubah dengan beban nonlinier. Simulasi akan dijalankan selama 3s. Bentuk sinyal referensi yang diuji adalah sinyal step dan sinyal ramp untuk melihat efek penurunan kecepatan referensi secara mendadak dan perlahan.



Gambar 4.6 Pengujian *setpoint* berubah-ubah

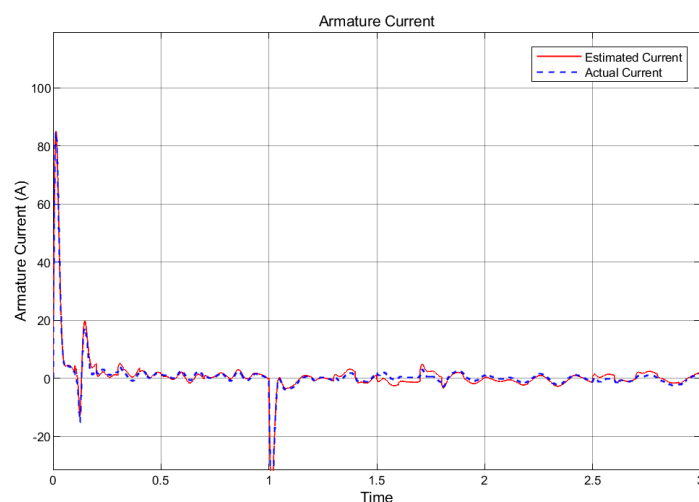
Pada gambar 4.6 terdapat sinyal step yang turun dari $\omega_{ref} = 100$ sampai $\omega_{ref} = 50$ pada $t = 1\text{s}$, dan sinyal ramp yang turun dari $\omega_{ref} = 50$ sampai $\omega_{ref} = 20$ pada $t = 2\text{s}$ sampai $t = 3\text{s}$. Dapat dilihat bahwa pada penurunan step dan ramp, sinyal estimasi dapat memiliki

respon yang lebih cepat dengan *error* yang lebih kecil. Dari $t = 1\text{s}$ sampai $t = 2\text{s}$ pada sinyal step, sinyal estimasi memiliki nilai $RMSE = 9.244$ dan sinyal aktual memiliki nilai $RMSE = 15.436$. Dari $t = 2\text{s}$ sampai $t = 3\text{s}$ pada sinyal ramp, sinyal estimasi memiliki nilai $RMSE = 3.252$ dan sinyal aktual memiliki nilai $RMSE = 3.940$. $RMSE$ ini ditunjukkan dalam bentuk grafik pada gambar 4.7. Pada kedua jenis sinyal referensi, nilai $RMSE$ lebih besar dibandingkan input step konstan yang bernilai 1.538. Maka, estimasi kecepatan pada referensi berubah-ubah memiliki performa yang lebih buruk dibandingkan input konstan.



Gambar 4.7 $RMSE$ estimasi dan aktual dengan *setpoint* berubah-ubah

Gambar 4.8 menunjukkan grafik arus estimasi dan aktual pada pengujian *setpoint* berubah-ubah. Dapat dilihat bahwa arus estimasi menyerupai bentuk arus aktual, sama seperti pada pengujian *setpoint* konstan, dengan rata-rata $RMSE = 1.601$. Dibandingkan dengan pengujian *setpoint* tetap, $RMSE$ pada pengujian ini lebih tinggi dengan perbedaan 0.515. Ini menunjukkan bahwa secara keseluruhan, estimasi dengan *setpoint* berubah-ubah lebih buruk dibandingkan *setpoint* tetap.



Gambar 4.8 Arus estimasi dan aktual dengan *setpoint* berubah-ubah

BAB 5 Kesimpulan dan Saran

Pada bab ini akan dibahas mengenai kesimpulan yang didapatkan dari penelitian ini dan saran yang bisa digunakan untuk penelitian selanjutnya.

5.1 Kesimpulan

Berdasarkan analisa data dan pembahasan yang telah dijelaskan pada bab sebelumnya, didapatkan kesimpulan sebagai berikut:

1. Sistem kontrol dan estimasi menggunakan estimator EKF dan kontroler PI-PSO berhasil dalam mengendalikan dan mengestimasi kecepatan putaran motor DC *separately excited*.
2. Kualitas nilai *fitness* optimal dapat ditingkatkan dengan membesarkan populasi dan mengurangi nilai *upper bound* dalam algoritma PSO.
3. Fungsi objektif yang mempertimbangkan respon transien berhasil dalam mengendalikan respon transien kecepatan motor DC.
4. EKF menghasilkan kecepatan estimasi dengan *error setpoint* yang lebih kecil dibandingkan kecepatan aktual.
5. Estimasi kecepatan motor DC menggunakan EKF dengan *setpoint* berubah-ubah memiliki *error* lebih tinggi dibandingkan *setpoint* konstan.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan maka diperoleh saran untuk pengembangan kedepan, yaitu sebagai berikut:

1. Memvariasikan parameter algoritma PSO lainnya seperti koefisien inersia w dan koefisien percepatan c sehingga hasil *fitness* lebih optimal.
2. Mengembangkan sistem adaptif untuk menyesuaikan kovarians pengukuran Q dan kovarians proses R dengan perubahan dalam sistem agar hasil estimasi EKF lebih baik.
3. Mengembangkan metode tuning bobot fungsi objektif yang adaptif.

DAFTAR PUSTAKA

- Tripathi, R., Singh, A., Gangwar, P., & Verma, R. (2022). Sensorless speed control of DC motor using EKF estimator and TSK fuzzy logic controller, *Automatika*, 63:2, 338-348, DOI: 10.1080/00051144.2022.2039990
- Rigatos, G.G. (2009). Particle and Kalman filtering for state estimation and control of DC motors. *ISA transactions*, 48 1, 62-72 .
- Zhang, Yudong., Wang, Shuihua., Ji, Genlin. (2015) "A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications", *Mathematical Problems in Engineering*, vol. 2015, Article ID 931256, 38 pages. <https://doi.org/10.1155/2015/931256>
- Aydogmus, O., Talu, M. F.. (2012). Comparison of Extended-Kalman-Filter and Particle-Filter-Based Sensorless Speed Control. *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 2, pp. 402-410, Feb. 2012, doi: 10.1109/TIM.2011.2164851.
- B. Terzic and M. Jadric, (2001). "Design and implementation of the extended Kalman filter for the speed and rotor position estimation of brushless DC motor," in *IEEE Transactions on Industrial Electronics*, vol. 48, no. 6, pp. 1065-1073, Dec. 2001, doi: 10.1109/41.969385
- Gundogdu, A., Celikel, R. & Aydogmus, O. (2020). Comparison of SI-ANN and Extended Kalman Filter-Based Sensorless Speed Controls of a DC Motor. *Arab J Sci Eng* 46, 1241–1256 (2021). <https://doi.org/10.1007/s13369-020-05014-3>
- Zafer Aydogmus, Omur Aydogmus. (2014). A comparison of artificial neural network and extended Kalman filter based sensorless speed estimation, *Measurement*, Volume 63, 2015, Pages 152-158, ISSN 0263-2241, <https://doi.org/10.1016/j.measurement.2014.12.010>.
- Štimac, G., Braut, S. & Žigulić, R. (2014). Comparative analysis of PSO algorithms for PID controller tuning. *Chin. J. Mech. Eng.* **27**, 928–936. <https://doi.org/10.3901/CJME.2014.0527.302>
- Wang, H., Hu, Y., Liao, W. and Yan, T. (2015) ‘Optimal PID control of DC motor with ABC and PSO algorithms’, *Int. J. Advanced Mechatronic Systems*, Vol. 6, No. 5, pp.193–200.
- Yudong Zhang, Shuihua Wang, Genlin Ji. (2015) "A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications", *Mathematical Problems in Engineering*, vol. 2015, Article ID 931256, 38 pages. <https://doi.org/10.1155/2015/931256>
- Hao Feng, Wei Ma, Chenbo Yin, Donghui Cao, (2021). Trajectory control of electro-hydraulic position servo system using improved PSO-PID controller, *Automation in Construction*, Volume 127, 2021, 103722, ISSN 0926-5805, <https://doi.org/10.1016/j.autcon.2021.103722>.
- Solihin, Mahmud & Tack, Lee & Moey, Lip Kean. (2011). Tuning of PID Controller Using Particle Swarm Optimization (PSO). *Proceeding of the International Conference on Advanced Science, Engineering and Information Technology*. 1. 10.18517/ijaseit.1.4.93.
- Nise, S. N., (2010) *Control Systems Engineering*. Wiley, p. 79-82.

- El-Hawary, Mohamed, E. (2002). Principles of Electric Machines with Power Electronic Applications. *Wiley-IEEE Press*. p. 183-184.
- El-Deen, A., Mahmoud, A., El-Sawi, A. (2015). Optimal PID Tuning for DC Motor Speed Controller Based on Genetic Algorithm. *International Review of Automatic Control* vol. 8 N. 1, pp 80-85.
- Bindu, R., Namboothiripad, M. K. (2012) Tuning of PID Controller for DC Servo Motor using Genetic Algorithm. *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, pp.310-314, 2012.
- Zahir, A.A. M., Alhady, S.S.N., Othman, W.A.F.W., Ahmad, M.F. (2018). Genetic Algorithm Optimization of PID Controller for Brushed DC Motor. *Intelligent Manufacturing & Mechatronics. Lecture Notes in Mechanical Engineering*. Springer, Singapore. https://doi.org/10.1007/978-981-10-8788-2_38
- Bernard, Albinus. (2013). Speed control of separately excited dc motor using artificial intelligent approach. Masters thesis, Universiti Tun Hussein Onn Malaysia.

LAMPIRAN

Source code MATLAB

a. Parameter motor DC

```
% Parameters
Ts = 10^-5; % [s] sampling time

% DC motor model
Va = 240; % [V] arm voltage
Vf = 300; % [V] arm field
Ra = 2.581; % [Ohm] arm resistance
La = 0.028; % [H] arm inductance
Laf = 0.9483; % [H]
J = 0.02215; % [kg/m^2] inertia
D = 0.002953; % [Nms] damping coeff
Tf = 0.5161; % [Nm]
K = 1.79; % [Nm/A]

% Load Parameters
m = 5; % [kg] mass
l = 0.05; % [m] armature length
g = 9.81; % [m/s^2] gravity
```

b. Algoritma PSO

```
tstart = tic;
% PSO parameters
nVar = 2;
ub = [15 25];
lb = [0 0];
fobj = @PSO_tuning;

noP = 25;
maxIter = 30;
wMax = 1;
wMin = 0.1;
c1 = 2;
c2 = 2;

% Initialize
for k = 1 : noP
    Swarm.Particles(k).X = (ub-lb) .* rand(1,nVar) + lb; % posisi awal
    Swarm.Particles(k).V = zeros(1, nVar); % kecepatan awal
    Swarm.Particles(k).PBEST.X = zeros(1,nVar);
    Swarm.Particles(k).PBEST.O = inf; % Pbest awal

    Swarm.GBEST.X = zeros(1,nVar);
    Swarm.GBEST.O = inf; % Gbest awal
end
vMax = (ub - lb) .* 0.2;
vMin = -vMax;
```

```

% Main loop
for t = 1 : maxIter
    % Fitness value
    for k = 1 : noP
        currentX = Swarm.Particles(k).X;
        Swarm.Particles(k).O = fobj(currentX);
        % Update Best Position
        if Swarm.Particles(k).O < Swarm.Particles(k).PBEST.O
            Swarm.Particles(k).PBEST.X = currentX;
            Swarm.Particles(k).PBEST.O = Swarm.Particles(k).O;
        end
        % Update Global Best
        if Swarm.Particles(k).O < Swarm.GBEST.O
            Swarm.GBEST.X = currentX;
            Swarm.GBEST.O = Swarm.Particles(k).O;
        end
    end
    % Update X and V
    w = wMax - t .* ((wMax - wMin) / maxIter);
    for k = 1 : noP
        % update V
        Swarm.Particles(k).V = w .* Swarm.Particles(k).V ...
            + c1 .* rand(1,nVar) .* (Swarm.Particles(k).PBEST.X -
Swarm.Particles(k).X) ...
            + c2 .* rand(1,nVar) .* (Swarm.GBEST.X - Swarm.Particles(k).X);
        % Check V
        index1 = find(Swarm.Particles(k).V > vMax);
        index2 = find(Swarm.Particles(k).V < vMin);

        Swarm.Particles(k).V(index1) = vMax(index1);
        Swarm.Particles(k).V(index2) = vMin(index2);
        % update X
        Swarm.Particles(k).X = Swarm.Particles(k).X + Swarm.Particles(k).V;
        % Check X
        index1 = find(Swarm.Particles(k).X > ub);
        index2 = find(Swarm.Particles(k).X < lb);

        Swarm.Particles(k).X(index1) = ub(index1);
        Swarm.Particles(k).X(index2) = lb(index2);
    end

    outmsg = ['Iterasi#', num2str(t) , ' Global Best = ' ,
num2str(Swarm.GBEST.O), ' X = ', num2str(Swarm.GBEST.X)];
    disp(outmsg);

    cgCurve(t) = Swarm.GBEST.O;
end
% Plot Iterations
semilogy(cgCurve);
xlabel('Iterasi')
ylabel('Weight')
tend = toc(tstart);
disp(tend);

```

c. Perhitungan Fungsi Objektif

```

function fitness = PSO_tuning(kk) % init
assignin('base', 'kk', kk);

```



```

sim('DC_motor_disc.slx');
int_error = ITAE(length(ITAE));

setp = 100; % input

result = stepinfo(omega, time, setp);

risetime = result.RiseTime;
overshoot = result.Overshoot;
settlingtime = result.SettlingTime;
err_ss = abs(setp-omega(end));

Fitness1 = (risetime)*50;
Fitness2 = (err_ss)*1;
Fitness3 = (overshoot)*0.8;
Fitness4 = int_error*5;
Fitness5 = settlingtime*5;
% 50 1 0.8 5 5

fitness = Fitness1 + Fitness2 + Fitness3 + Fitness4 + Fitness5;

end

```

d. Blok EKF simulink

```

function [X_pred1, P_pred1] = fcn(wm,theta,vt,ia,P)

w_est = 1*wm + 0.0005*ia + (-7.0677e-04*cos(theta)-1.4873e-04);
i_est = -0.0006*wm + 0.9991*ia + 3.5714e-04*vt;
X_pred1 = [w_est; i_est]; % prediksi state

Fk= [1.0000 0.0005; % jacobian
      -0.0006 0.9991];
Q =[0.5 0; 0 0.5]; % kovarians proses

P_pred1 = Fk*P*Fk' + Q; % kovarians prediksi

function [w_est, P_pred2, i_est] = fcn(X_pred1, P_pred1, ia)

Hk = [0 0; 0 1]; % jacobian
R = [0.5 0; 0 0.5]; % kovarians proses

yk = Hk*[0;ia]; % vektor observasi
Sk = Hk*P_pred1*Hk' + R; % kovarians inovasi

K_gain = P_pred1*Hk'*(Sk^-1); % kalman gain

X_pred2 = X_pred1 + K_gain*(yk - Hk*X_pred1); % estimasi state
P_pred2 = P_pred1*(eye(2)-K_gain*Hk); % estimasi kovarians prediksi

w_est = X_pred2(1); % estimasi kecepatan
i_est = X_pred2(2); % estimasi arus

```

BIODATA PENULIS



Penulis dilahirkan di Jakarta, 13 April 2001, merupakan anak kedua dari 2 bersaudara. Penulis telah menempuh pendidikan formal yaitu di SMP Santa Ursula BSD dan SMA Binus Serpong. Setelah lulus dari SMA tahun 2019, Penulis mengikuti SBMPTN dan diterima di Departemen Teknik Elektro FTEIC - ITS pada tahun 2019 dan terdaftar dengan NRP 07111940000110.

Di Departemen Teknik Elektro Penulis sempat aktif di beberapa kegiatan Seminar yang diselenggarakan oleh Departemen, event kepanitiaan seperti ELECTRA, dan sebagai asisten praktikum LPDO.