Kontrol Kecepatan Motor DC dengan Estimator Kecepatan Extended Kalman Filter dan Kontroler Proportional Integral Berbasis Particle Swarm Optimization

Kiet Pascal Asmara, Eka Iskandar, dan Yusuf Bilfaqih
Departemen Teknik Elektro, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember (ITS)
e-mail: kietpascal@gmail.com, iskandar@elect-eng.its.ac.id, bilfaqih@ee.its.ac.id

***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*— EKF, Fungsi objektif, Motor DC, PSO**

I. PENDAHULUAN

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* [1]. 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.

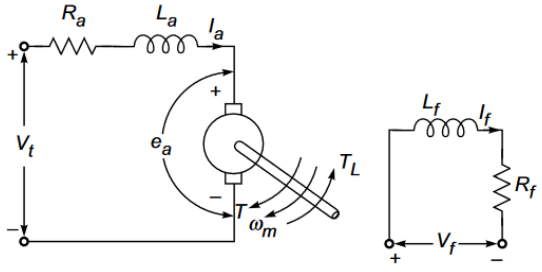
Dalam pengendalian *sensorless*, kecepatan motor diestimasi menggunakan pengukuran sensor listrik menggunakan sebuah *observer*. *Observer* tersebut melakukan estimasi kecepatan sudut motor menggunakan model matematika motor dan sinyal listrik, umumnya dengan nilai arus dan tegangan pada armature [2]. 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.

II. TINJAUAN PUSTAKA

A. Motor DC

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.



Gambar. 1. Model motor DC separately excited

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:

$$v(t) = K_e i_f(t)\omega_m + R_a i_a(t) + L_a \frac{d}{dt} i_a(t) \quad (1)$$

dimana $v(t)$ adalah tegangan suplai armature, adalah back emf, R_a adalah resistansi armature, $i_a(t)$ adalah arus armature, L_a adalah induktansi armature, K_e adalah konstan back emf dan ω_m adalah kecepatan motor [3].

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

$$T(t) = J \frac{d}{dt} \omega_m(t) + T_L(t) + T_f(t) + D\omega_m(t) \quad (2)$$

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 [4].

Torsi beban T_L dapat diturunkan menjadi:

$$T_L = mgL\cos\theta + mL^2 \frac{d}{dt} \omega_m(t) \quad (3)$$

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 (1), (2), dan (3) dengan mencari persamaan untuk dua variabel state yaitu arus armature i_a dan kecepatan motor ω_m. Diperoleh persamaan state space motor DC sebagai berikut:

$$\begin{bmatrix} \dot{\omega_m} \\ \dot{i_a} \end{bmatrix} = \begin{bmatrix} -\frac{D}{mL^2+J} & \frac{K_t}{mL^2+J} \\ -\frac{K_e}{L_a} & -\frac{R_a}{L_a} \end{bmatrix} \begin{bmatrix} \omega_m \\ i_a \end{bmatrix} + \begin{bmatrix} \frac{-mgL\cos\theta - T_f}{mL^2+J} & 0 \\ 0 & \frac{1}{L_a} \end{bmatrix} \begin{bmatrix} 1 \\ v_t \end{bmatrix} \quad (4)$$

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

$$x_{k+1} = A_d x_k + B_d u_k + v_k \quad (5)$$

$$y_k = C_d x_k + w_k \quad (6)$$

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 di-diskritisasi pada waktu sampling T_s dalam persamaan berikut [5]:

$$A_d = I + AT_s \quad (7)$$

$$B_d \approx BT_s \quad (8)$$

$$C_d = C \quad (9)$$

B. 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.*

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. [6].

Langkah prediksi EKF adalah sebagai berikut:

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_k) \quad (10)$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q \quad (11)$$

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:

$$S_k = H_k P_{k|k-1} H_k^T + R \quad (12)$$

$$K_k = P_{k|k-1} H_k^T {S_k}^{-1} \quad (13)$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\left(y_{k+1} - H_k \hat{x}_{k|k-1}\right) \quad (14)$$

$$P_{k|k} = P_{k|k-1}(I - K_k H_k) \quad (15)$$

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:

$$F_k = \frac{\partial f(\hat{x}_k, u_k)}{\partial x} \quad (16)$$

$$H_k = \frac{\partial h(\hat{x}_{k|k-1})}{\partial x} \quad (17)$$

Persamaan (10)-(15) merupakan satu siklus algoritma EKF [7].

C. 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:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)\, dt + K_d \frac{de(t)}{dt} \quad (18)$$

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* [8].

D. PSO

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[9].

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

$$p_{best}(i,t) = \arg min\,[f(X_i(k))], \quad i \in \{1,2,\dots,N_p\} \tag{19}$$
$$g_{best}(t) = \arg min\,[f(X_i(k))] \tag{20}$$

dimana i adalah index partikel, t adalah nomor iterasi, N_p adalah jumlah total partikel, P adalah posisi dan *f* adalah fungsi objektif.

Posisi dan kecepatan di update sesuai dengan persamaan berikut:

$$V_i(t+1) = wV_i(t) + c_1 r_1\big(p_{best}(i,t) - X_i(t)\big) + c_2 r_2\big(g_{best}(t) - X_i(t)\big) \tag{21}$$
$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{22}$$

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] [10]. 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:

$$w = w_{max} - t\left(\frac{w_{max} - w_{min}}{t_{max}}\right) \tag{23}$$

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 [10]. Ini ditunjukkan oleh persamaan berikut:

$$V_{max} = 0.2(ub - lb) \tag{24}$$
$$V_{min} = -V_{max} \tag{25}$$

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

$$ITAE = \int_0^\tau t|e(t)|dt \tag{26}$$

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

III. PERANCANGAN ALAT

A. Modelling Motor DC

Pada tugas akhir ini motor DC yang digunakan adalah motor DC *separately excited* dengan parameter yang dapat dilihat pada tabel 1. Parameter tersebut diperoleh dari referensi [3].

Tabel 1.
Parameter model motor DC

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

Dengan menggunakan nilai parameter diatas dan memasukkannya ke (4), dapat diperoleh persamaan state space sebagai berikut:

$$\begin{bmatrix}\dot{\omega_m}\\ \dot{i_a}\end{bmatrix} = \begin{bmatrix}-0.0852 & 51.6595\\ -63.9286 & -92.1786\end{bmatrix}\begin{bmatrix}\omega_m\\ i_a\end{bmatrix} + \begin{bmatrix}-70.677\cos\theta - 14.873 & 0\\ 0 & 35.7143\end{bmatrix}\begin{bmatrix}1\\ v_t\end{bmatrix} \tag{27}$$

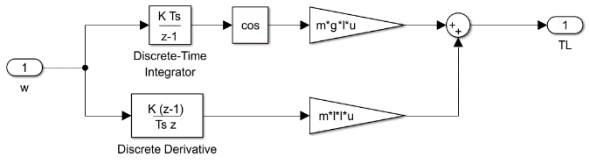
Untuk mengimplementasikan model secara digital, (4) perlu diubah menjadi diskrit. Matriks koefisien A, B, dan C perlu diubah menjadi diskrit menggunakan (7) – (9). Dengan memilih waktu sampling $T_s = 10^{-5}$ bentuk diskrit matriks dapat dihitung menjadi:

$$A_d = I + AT_s = \begin{bmatrix}1 & 0.0005\\ -0.0006 & -0.9991\end{bmatrix} \tag{28}$$

$$B_d \approx B(k)T_s = \begin{bmatrix}-7.0677e^{-4}cos\,\theta - 1.4873e^{-4} & 0\\ 0 & 3.57143e^{-4}\end{bmatrix} \tag{29}$$

$$C_d = C(k) = \begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix} \tag{30}$$

Setelah diketahui parameter motor DC dapat dibuat simulasi pada Simulink. Parameter tersebut dimasukkan pada blok *DC machine*. Model beban pada simulink berdasarkan (3) ditunjukkan pada gambar 2.



Gambar. 2. Model beban nonlinier

B. Perancangan PI-PSO

Tabel 2.
Parameter algoritma PSO

Parameter	Nilai
Jumlah iterasi $\boldsymbol{t}$	30
Jumlah variabel $\boldsymbol{N_{var}}$	2
Lower bound	[0 0]
$\boldsymbol{w_{max}, w_{min}}$	1, 0.1
$\boldsymbol{c_1, c_2}$	2, 2

Parameter yang terpilih untuk algoritma PSO dapat dilihat pada tabel 2 Jumlah iterasi ditentukan berdasarkan kecepatan ditemukan g_{best} yang umumnya memerlukan 20-30 iterasi.

Algoritma PSO terdiri dari inisialisasi dan update. Dalam inisialisasi dicari nilai awal posisi, kecepatan, p_{best}, g_{best}, V_{min}, dan V_{max}. Posisi diinisialisasikan sebagai (31) 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 (24) dan (25)

$$X_1 = (ub - lb)rand(1, N_{var}) + lb \quad (31)$$

Langkah update terdiri dari perhitungan *fitness* posisi X setiap partikel, update p_{best} dan g_{best}, update X, update V, dan output g_{best} [11]. 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 (23). Untuk menghitung X dan V digunakan (21) dan (22).

Umumnya PSO menggunakan kriteria error seperti ITAE, IAE, atau ISE sebagai fungsi objektif [12]. Tetapi, fungsi objektif tersebut kurang cocok untuk mendesain kontroler PI dengan baik [13]. Maka, pada penelitian ini akan digunakan fungsi objektif yang mempertimbangkan respon trnasien dan ITAE. Fungsi objektif tersebut dapat meningkatkan kualitas respon sistem dibanding fungsi ITAE saja. Fungsi objektif tersebut adalah:

$$f(K_p, K_i) = \mu \int_0^\tau t|e(t)|dt + \alpha O_v + \beta SS_e + \delta t_s + \gamma t_r \quad (32)$$

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 $35\%, 20\%, 5\%, 15\%, 25\%$. Pembobotan yang terpilih berdasarkan respon yang diinginkan, yaitu *overshoot* kecil, *error* minimum, *rise time* cepat, dan *settling time* yang lebih pelan. 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*.

C. Perancangan Estimator EKF

Algoritma *extended kalman filter* ditunjukkan oleh (10) – (15) yang terdiri dari langkah prediksi dan update, dengan perhitungan matriks *jacobian* pada (16) dan (17) untuk melakukan liniearisasi. Untuk mengaplikasikan algoritma EKF, matriks *jacobian* perlu di evaluasi terlebih dahulu. Pada sistem ini Dari persamaan state space (5) dan (6) ditemukan matriks *jacobian* sebagai berikut:

$$F_k = \begin{bmatrix} 1 & 0.0005 \\ -0.0006 & -0.9991 \end{bmatrix} = A_d \quad (33)$$

$$H_k = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} = C_d \quad (34)$$

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 pengukuran dan proses, Q dan R perlu ditentukan bersama dengan nilai awal kovarians P. Setelah melakukan tuning, nilai awal kovarians $P_{k-1|k-1}$, Q, dan R yang digunakan adalah:

$$P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (35)$$

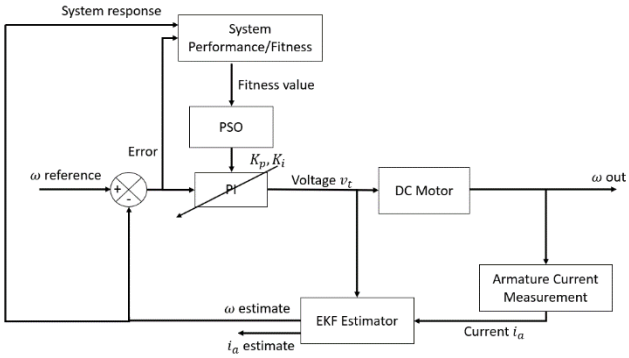
$$Q = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \quad (36)$$

$$R = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \quad (37)$$

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 [14]. Rumus mencari RMSE adalah:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2} \quad (38)$$

Sistem secara keseluruhan ditunjukkan pada gambar 3.



Gambar. 3. Diagram blok sistem estimasi dan kontrol kecepatan motor DC dengan EKF dan PI-PSO

IV. ANALISA HASIL

A. Hasil Pengujian algoritma PSO.

Algoritma PSO yang menggunakan fungsi objektif termodifikasi dijalankan menggunakan parameter yang tertera di tabel 3. 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* <1.5s, dan *rise time* <0.1s. Algoritma PSO yang menggunakan fungsi objektif ITAE juga akan dilihat responnya untuk dibandingkan

dengan PSO yang menggunakan fungsi objektif termodifikasi.

Tabel 3. Hasil pengujian PSO

Skenario	Populasi	Max K_p	Max K_i	Solusi$_{(K_p,K_i)}$	*Fitness* 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 3, 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.

Ditemukan bahwa nilai fitness optimal cenderung lebih baik ketika nilai upper bound lebih kecil, terutama pada skenario 3 sampai 10 ketika upper bound Kp≤50. Terdapat perbedaan nilai fitness sebesar 0.3246 antara skenario 1 dan 3, dengan Kp=100 dan Kp=50. Sedangkan, perbedaan skenario 3 dengan 7 yang memiliki solusi terbaik hanya sebesar 0.2446. Ini berarti range pencarian Kp=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.

Tabel 4. Respon transien kecepatan motor skenario 2 dan 7

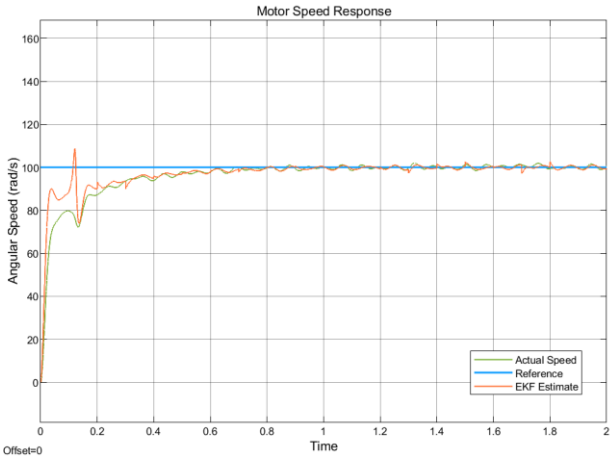
Parameter	Skenario 2	Skenario 7
Overshoot	8.8374	8.8283
Settling time	0.8589	0.6783
Rise time	0.0283	0.0312

Tabel 4 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 memilki 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.

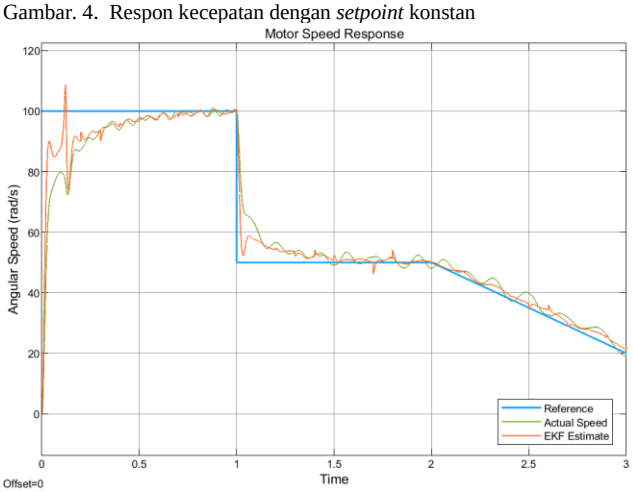
Dengan ITAE saja sebagai fungsi objektif, respon transien tidak dapat dikendalikan. Walaupun error ITAE sangat kecil, *overshoot* bernilai sangat besar dan tidak bisa dikendalikan. Dengan membandingkan respon transien tersebut dengan respon transien pada tabel IV, dapat dilihat bahwa respon transien dapat dikendalikan dengan menambahkan respon transien pada fungsi objektif.

B. Pengujian Estimator EKF

Pengujian pertama adalah dengan referensi sinyal konstan $100\,rad/s$. Karakteristik respon yang diinginkan adalah *overshoot* $< 10\%$, *settling time* $< 1.5s$, *rise time* $< 0.1s$, dan *steady state* yang berosilasi pada $100\,rad/s$. Simulasi dijalankan selama 2s. Gambar 4 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.



Gambar. 4. Respon kecepatan dengan *setpoint* konstan



Gambar. 4. Respon kecepatan dengan *setpoint* berubah-ubah

Tabel 5. Respon transien sistem dengan *setpoint* konstan

Parameter	$\boldsymbol{\omega}$ Estimasi	$\boldsymbol{\omega}$ Aktual
Overshoot	8.8283	1.9196
Settling time	0.6783	0.9465
Rise time	0.0312	0.2181

Tabel 5 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 6.8508 dan *settling time* sebesar 0.1634. *Rise time* kecepatan aktual lebih lambat dengan perbedaan sebesar $0.0072s$. Selain *overshoot*, respon transien kecepatan estimasi cukup mendekati kecepatan aktual.

Dengan melihat *steady state* dari $t = 1s$ sampai $t = 2s$, diperoleh error RMSE bernilai 1.5497 untuk sinyal estimasi dan 1.9927 untuk sinyal aktual. Ini menunjukkan bahwa kecepatan estimasi memiliki *error* dengan *setpoint* yang lebih kecil dibandingkan kecepatan aktual.

Lalu, 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.

Pada gambar 4 dilakukan pengujian dengan *setpoint* kecepatan referensi yang berubah-ubah. Terdapat sinyal step yang turun dari $\omega_{ref} = 100$ sampai $\omega_{ref} = 50$ pada $t = 1s$, dan sinyal ramp yang turun dari $\omega_{ref} = 50$ sampai $\omega_{ref} = 20$ pada $t = 2s$ sampai $t = 3s$. Dapat dilihat bahwa pada penurunan step dan ramp, sinyal estimasi dapat memiliki respon yang lebih cepat dengan *error* yang lebih kecil. Dari $t = 1s$ sampai $t = 2s$ pada sinyal step, sinyal estimasi memiliki nilai $RMSE = 9.244$ dan sinyal aktual memiliki nilai $RMSE = 15.436$. Dari $t = 2s$ sampai $t = 3s$ pada sinyal ramp, sinyal estimasi memiliki nilai $RMSE = 3.252$ dan sinyal aktual memiliki nilai $RMSE = 3.940$. 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.

V. 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.

DAFTAR PUSTAKA

[1] Aydogmus, O., Talu, M. F.. (2012). Comparison of Extended-Kalman- 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.

[2] Rigatos, G.G. (2009). Particle and Kalman filtering for state estimation and control of DC motors. ISA transactions, 48 1, 62-72

[3] Nise, S. N., (2010) Control Systems Engineering. Wiley, p. 79-82.

[4] 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

[5] Aydogmus, Zafer., Aydogmus. Omur. (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.

[6] 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

[7] Terzic, B.., Jadric, M., (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

[8] 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.

[9] 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

[10] Feng, Hao., 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.

[11] 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

[12] 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.

[13] 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

[14] 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