

Optimisasi Pengontrol LQR menggunakan Algoritma *Stochastic Fractal Search*

Yul Y. Nazaruddin, Franky & I G. N. A. Indra Mandala

Instrumentation and Control Research Group, Department of Engineering Physics,
Institut Teknologi Bandung, Bandung 40132, Indonesia

yul@tf.itb.ac.id

frankyl7997@gmail.com

inmangusti@gmail.com

Abstrak

Linear Quadratic Regulator (LQR) merupakan salah satu metoda kontrol optimal pada sistem yang berbasis ruang keadaan. Pengontrol LQR memiliki dua buah parameter yaitu matriks bobot Q dan R , yang harus ditentukan sehingga dapat menghasilkan aksi kontrol yang optimal sesuai harapan. Tidak seperti pengontrol Proporsional-Integral-Derivatif (PID) yang memiliki metoda penalaan dengan pendekatan sistematis seperti *Ziegler-Nichols* dan *Cohen-Coon*, pengontrol LQR tidak memiliki metode penalaan khusus secara sistematis untuk menentukan matriks bobot Q dan R . Pada makalah ini diusulkan suatu solusi untuk menentukan parameter - parameter desain pengontrol LQR tersebut dengan menggunakan algoritma meta heuristik terbaru yaitu *Stochastic Fractal Search (SFS)*. Algoritma SFS ini terinspirasi dari fenomena pertumbuhan alami dengan konsep matematis yang disebut fraktal. Dengan memanfaatkan sifat difusi dari fraktal secara acak (stokastik), partikel tersebut dapat melakukan eksplorasi dan eksploitasi dalam ruang pencarian secara efektif dan efisien. Pengontrol LQR dengan optimisasi menggunakan algoritma SFS yang diusulkan kemudian diterapkan pada suatu model kendaraan tanpa awak (*quadrotor*) dengan 12 variabel keadaan. Hasil simulasi memperlihatkan bahwa hanya dengan 100 iterasi, posisi *quadrotor* dapat dikontrol dengan baik sehingga mampu mengikuti perubahan *set-point* yang diberikan dengan cukup memuaskan.

Kata Kunci: linear quadratic regulator; optimisasi; quadrotor; stochastic fractal search

1 Pendahuluan

Linear Quadratic Regulator (LQR) adalah metoda kontrol optimal yang sudah sering digunakan dalam berbagai bidang, contohnya dalam industri [1], robotik [2], maupun bidang keteknikan lain [3]. Keunggulan metoda ini adalah dapat memberikan solusi yang optimal untuk masalah pengontrolan sistem yang didefinisikan dalam ruang keadaan. Karena berbasis ruang keadaan, metode LQR dapat dengan baik memecahkan masalah pengontrolan pada sistem *Multi Input Multi Output (MIMO)*.

Pengontrol LQR memiliki dua buah parameter yang harus ditentukan untuk menghasilkan aksi kontrol yang diharapkan, yaitu matriks bobot Q dan R . Namun, berbeda dengan pengontrol Proporsional-Integral-Derivatif (PID) yang memiliki banyak metoda penalaan secara sistematis diantaranya *Ziegler-Nichols* dan *Cohen-Coon*, pengontrol LQR tidak memiliki metoda penalaan khusus untuk menentukan kedua parameter matriks bobot Q dan R .

Relasi antara matriks bobot Q dan R dengan dinamika dari sistem lup tertutup (*closed-loop system*) sangat kompleks. Dalam praktik, efek dari pasangan matriks bobot terhadap perilaku lup tertutup dari sistem tidak dapat diprediksi. Salah satu pendekatan yang dilakukan sering kali hanyalah mencoba-coba serangkaian pasangan matriks bobot dalam rentang tertentu kemudian memilih pasangan yang menghasilkan respon dinamik yang diinginkan [4]. Oleh karena itu, dalam penelitian ini diusulkan penggunaan suatu algoritma meta-heuristik untuk melakukan optimisasi matriks bobot Q dan R .

Algoritma meta-heuristik adalah algoritma optimisasi yang terinspirasi dari karakteristik atau perilaku suatu objek atau makhluk hidup yang ada di alam. Algoritma meta-heuristik sering digunakan untuk memecahkan masalah sulit dalam ruang pencarian yang besar. Salah satu contoh algoritma meta-heuristik adalah algoritma *Stochastic Fractal Search (SFS)*. Algoritma ini merupakan salah satu algoritma meta-heuristik terbaru yang telah terbukti lebih baik dibandingkan banyak algoritma lain yang sering digunakan seperti *Particle Swarm Optimization (PSO)*, *Cuckoo Search (CS)*, *Modified Cuckoo Search (MCS)*, *Gravitational Search Algorithm (GSA)*, dan *Artificial Bee Colony (ABC)* [5]. Keunggulan dari algoritma SFS tidak hanya dari segi akurasi tinggi dan waktu konvergensi yang relatif singkat, algoritma ini juga relatif mudah untuk diimplementasikan.

Dalam penelitian ini, algoritma SFS digunakan untuk melakukan optimisasi terhadap matriks bobot Q dan R sehingga dapat mengurangi waktu pencarian nilai kedua parameter ini secara signifikan, sekaligus mendapatkan nilai matriks bobot yang optimal sehingga respon dinamik dari sistem yang dikontrol sesuai dengan kriteria tertentu. Sistem yang akan dikontrol adalah kendaraan tanpa awak yaitu *Quadrotor*. Model *Quadrotor* yang digunakan memiliki 12 variabel keadaan dengan 4 buah masukan dan 3 buah keluaran yaitu posisi dalam arah sumbu x , y , dan z (variabel yang akan dikontrol) [6].

2 Formulasi Masalah

2.1 Model Quadrotor

Quadrotor dikategorikan sebagai salah satu permasalahan yang sangat menantang dalam kontrol terapan dikarenakan kompleksitas, non-linearitas, dan karakteristik *underactuated*-nya [7]. Banyak peneliti yang telah memodelkan dinamika dari *Quadrotor* dengan beragam pendekatan. Dalam penelitian [6] terdapat dua model *Quadrotor*, yaitu model non-linier dan model hasil linierisasi. Penelitian ini akan menggunakan model hasil linearisasi sehingga metoda LQR dapat digunakan untuk merancang pengontrol untuk *Quadrotor*. Berikut merupakan model ruang keadaan dari *Quadrotor* yang digunakan.

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \quad (1)$$

$$\mathbf{x} = [\phi \ \theta \ \varphi \ p \ q \ r \ u \ v \ w \ x \ y \ z]^T \quad (2)$$

$$\mathbf{u} = [f_T \ \tau_x \ \tau_y \ \tau_z]^T \quad (3)$$

Dengan \mathbf{x} adalah variabel keadaan yang terdiri dari orientasi sudut (ϕ, θ, φ), kecepatan sudut (p, q, r), kecepatan translasi (u, v, w), dan posisi translasi (x, y, z) dalam arah sumbu x , y dan z secara berturut - turut. \mathbf{u} adalah variabel masukan yaitu gaya angkat (f_T), torsi dalam arah sumbu x (τ_x), torsi dalam arah sumbu y (τ_y), dan torsi dalam arah sumbu z (τ_z). \mathbf{A} adalah matriks transisi keadaan dan \mathbf{B} adalah matriks transisi masukan kontrol [6].

$$\mathbf{A} = \frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}} \bigg|_{\substack{\mathbf{x}=\bar{\mathbf{x}} \\ \mathbf{u}=\bar{\mathbf{u}}}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad (4)$$

$$\mathbf{B} = \frac{\partial f(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}} \bigg|_{\substack{\mathbf{x}=\bar{\mathbf{x}} \\ \mathbf{u}=\bar{\mathbf{u}}}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (5)$$

Variabel “ g ” pada persamaan (4) adalah percepatan gravitasi. Inersia pada arah sumbu x , sumbu y , dan sumbu z koordinat kartesian ditunjukkan oleh variabel “ I_x, I_y , dan I_z ” pada persamaan (5) secara berturut - turut. Sementara variabel “ m ” adalah massa *quadrotor* [7].

2.2 Optimisasi Pengontrol LQR Menggunakan Algoritma SFS

Linear Quadratic Regulator (LQR) merupakan salah satu metode kontrol yang banyak diterapkan, sehingga banyak referensi yang telah membahas pengaplikasian metode kontrol optimal ini. Tinjau sistem dengan persamaan keadaan seperti persamaan (1). Pada prinsipnya, metoda LQR mencari suatu sinyal kontrol \mathbf{u} yang meminimumkan indeks performansi J [7].

$$J = \int_0^{\infty} (\mathbf{x}^* \mathbf{Q} \mathbf{x} + \mathbf{u}^* \mathbf{R} \mathbf{u}) dt \quad (6)$$

Tanda “ $*$ ” menunjukkan transpose konjugat dari suatu vektor atau matriks. Sementara sinyal kontrol \mathbf{u} diberikan oleh persamaan berikut.

$$\mathbf{u} = -\mathbf{K}\mathbf{x} \quad (7)$$

$$\mathbf{K} = \mathbf{R}^{-1} \mathbf{B}^* \mathbf{P} \quad (8)$$

Matriks \mathbf{Q} dan \mathbf{R} merupakan matriks positif definit dan hermitian. Matriks \mathbf{Q} sering disebut dengan matriks bobot keadaan dan matriks \mathbf{R} sering disebut dengan matriks bobot pengontrolan. Kedua matriks ini harus dipilih sehingga

persamaan Algebraic Riccati Equation (ARE) yaitu persamaan (9) dapat dipecahkan, dan didapatkan suatu matriks positif definit P . Matriks P kemudian digunakan untuk mendapatkan matriks penguatan optimal K pada persamaan (8) sehingga didapatkan sinyal kontrol u [4].

$$A^*P + PA - PBR^{-1}B^*P + Q = 0 \quad (9)$$

Penelitian ini menggunakan algoritma SFS untuk melakukan optimisasi matriks Q dan R . Untuk menyederhanakan perhitungan, matriks Q dan R diasumsikan merupakan matriks diagonal positif definit.

Algoritma SFS merupakan algoritma optimisasi yang sangat *powerful* namun cukup mudah untuk diimplementasikan. Secara umum, algoritma SFS terdiri dari dua proses utama yaitu proses difusi dan proses pembaruan. Sebelum melakukan kedua proses utama tersebut, perlu dilakukan inisialisasi pada setiap titik atau disebut partikel pencarian (P). Inisialisasi yang dilakukan adalah pada batas minimum (LB) dan maksimum (UB) dari ruang pencarian partikel, jumlah partikel awal (M) dalam ruang pencarian, jumlah maksimum difusi (D) setiap partikel, dan jumlah iterasi maksimum (G) yang akan dilakukan. Fungsi kecocokan yang akan digunakan juga harus ditentukan terlebih dahulu. Penelitian ini menggunakan fungsi kecocokan berupa *Integral Absolut Error* (IAE). Sejumlah M partikel awal kemudian disebar dalam ruang pencarian menggunakan persamaan berikut:

$$P_j = LB + \varepsilon \times (UB - LB), j = 1, 2, \dots, M \quad (10)$$

Proses difusi (11-12) menghasilkan partikel baru yang melakukan pencarian dalam ruang yang didefinisikan. Algoritma SFS menggunakan distribusi *Gaussian* untuk melakukan proses difusi. Proses ini sering disebut dengan *Gaussian Walks* (GW_1 & GW_2) atau proses *Diffusion Limited Aggregation* (DLA) yang membentuk pola pertumbuhan fraktal [5].

$$GW_1 = \text{Gaussian}(\mu_{BP}, \sigma) + (\varepsilon \times BP - \varepsilon' \times P_i) \quad (11)$$

$$GW_2 = \text{Gaussian}(\mu_P, \sigma) \quad (12)$$

Nilai rata-rata (μ) dan standar deviasi (σ) untuk proses difusi pada persamaan (11-12) diberikan oleh persamaan berikut:

$$\mu_{BP} = |BP| \text{ \& } \mu_P = |P_i| \quad (13)$$

$$\sigma = \left| \frac{\log(g)}{g} \times (P_i - BP) \right| \quad (14)$$

dengan BP adalah partikel terbaik yang didapatkan dengan menguji setiap partikel dengan fungsi kecocokan yang digunakan, P_i adalah partikel ke- i yang melakukan difusi, dan g adalah jumlah iterasi yang telah dilakukan pada saat partikel melakukan difusi. Variabel ε dan ε' adalah angka acak yang terdistribusi *uniform* pada rentang nilai 0 – 1.

Setelah melalui proses difusi, partikel kemudian melalui proses pembaruan. Proses pembaruan terdiri dari proses pembaruan pertama dan proses pembaruan kedua. Proses pembaruan pertama diawali dengan pemberian nilai probabilitas kepada setiap partikel berdasarkan urutan nilai kecocokan partikel terhadap fungsi yang diminimumkan. Pemberian probabilitas ini bertujuan untuk memberikan nilai probabilitas tertinggi kepada partikel dengan nilai kecocokan terbaik sehingga memperbesar peluang partikel terburuk untuk diganti. Nilai probabilitas (Pa_i) ditunjukkan oleh persamaan berikut.

$$Pa_i = \frac{\text{urutan partikel } (P_i)}{\text{jumlah partikel } (N)} \quad (15)$$

Setelah diberi nilai probabilitas, maka partikel yang memiliki nilai $Pa_i < \varepsilon$ akan diperbarui nilainya dengan persamaan:

$$P'_i(j) = P_r(j) - \varepsilon \times (P_t(j) - P_i(j)) \quad (16)$$

dengan P'_i adalah posisi terbaru partikel, P_i adalah posisi partikel sebelumnya, sementara P_r dan P_t adalah partikel yang dipilih secara acak.

Setelah tahap pembaruan pertama selesai, maka seluruh partikel akan melakukan proses pembaruan kedua yang mirip dengan proses sebelumnya. Seluruh partikel diberi probabilitas sesuai persamaan (15) kemudian dilakukan pembaruan dengan dua kondisi berikut:

$$P''_t = P'_t - \hat{\varepsilon} \times (P'_t - BP), \text{ jika } \hat{\varepsilon} \leq 0.5 \quad (17)$$

$$P''_t = P'_t + \hat{\varepsilon} \times (P'_t - P'_r), \text{ jika } \hat{\varepsilon} > 0.5 \quad (18)$$

dengan P''_t adalah posisi partikel terbaru, P'_r dan P'_t adalah partikel setelah proses pembaruan pertama yang dipilih secara acak, dan $\hat{\varepsilon}$ adalah angka acak yang dihasilkan berdasarkan distribusi *Gaussian* standar. Gambaran lengkap dari algoritma SFS adalah sebagai berikut [5]:

Algoritma Pseudo dari SFS

#Inisialisasi partikel berdasarkan (10)

#Cari nilai partikel terbaik (BP)

#Dibuat suatu fungsi untuk melakukan difusi

```

def Difusi ( $P_i$ ,  $g$ ,  $D$ ):
    for i in range( $D$ ):
        #Untuk setiap partikel
        if (kondisi untuk  $GW_1$ ):
            #buat titik baru berdasarkan  $GW_1$ 
        else (kondisi untuk  $GW_2$ ):
            #buat titik baru berdasarkan  $GW_2$ 
    return  $P_{baru}$ 

#algoritma utama
for g in range( $G$ ):
    #lakukan difusi untuk setiap partikel
    Difusi ( $P_i$ ,  $g$ ,  $D$ )
    #perbarui nilai  $BP$ 

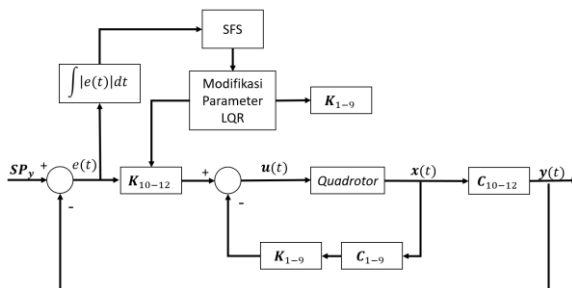
    #lakukan proses pembaruan pertama
    #pemberian probabilitas berdasarkan (15)
    for i in range(jumlah seluruh partikel):
        #untuk setiap komponen pada partikel
        if  $\varepsilon \geq Pa_i$ :
            #perbarui titik berdasarkan (16)
        else:
            #tidak lakukan apa-apa
    #perbarui nilai  $BP$ 

    #lakukan proses pembaruan kedua
    #pemberian probabilitas berdasarkan (15)
    for i in range(jumlah seluruh partikel):
        #untuk setiap partikel baru  $P'_i$ 
        if  $\varepsilon \geq Pa'_i$ :
            #perbarui titik berdasarkan persamaan (17)
            dan (18)
        else:
            #tidak lakukan apa-apa
    #perbarui nilai  $BP$ 

```

3 Implementasi dan Evaluasi

Posisi translasi *Quadrotor* (x, y, z) akan dikontrol menggunakan pengontrol LQR yang dioptimisasi menggunakan algoritma SFS. Keseluruhan proses digambarkan dalam diagram blok berikut:



Gambar 1. Diagram blok proses optimisasi pengontrol LQR pada *quadrotor*

dengan SP_y merupakan nilai *set-point* untuk posisi translasi *Quadrotor* (x, y, z), $u(t)$ merupakan sinyal

kontrol, $x(t)$ merupakan seluruh variabel keadaan yang berjumlah 12 yaitu persamaan (2), C adalah matriks keluaran dan $y(t)$ merupakan keluaran sistem yaitu posisi translasi dari *Quadrotor* (x, y, z). Indeks pada matriks K dan C pada Gambar 1 menunjukkan indeks dari variabel keadaan yang digunakan. Karena output $y(t)$ mengambil variabel keadaan x, y , dan z yang memiliki indeks 10, 11, dan 12 maka digunakan matriks C_{10-12} , begitu seterusnya untuk menyederhanakan penggambaran.

Simulasi dilakukan dengan menggunakan parameter *Quadrotor* seperti Tabel 1. Fungsi kecocokan yang digunakan pada proses optimisasi pengontrol LQR menggunakan algoritma SFS adalah *Integral Absolute Error* (IAE), berbentuk.

$$IAE = \int_0^t |e(t)| dt \quad (19)$$

$$e(t) = SP_y - y(t) \quad (20)$$

Tabel 1. Parameter model *Quadrotor*

I_x	0,01 kg m ²
I_y	0,01 kg m ²
I_z	0,2 kg m ²
m	0,5 kg
g	9,8 m/s ²

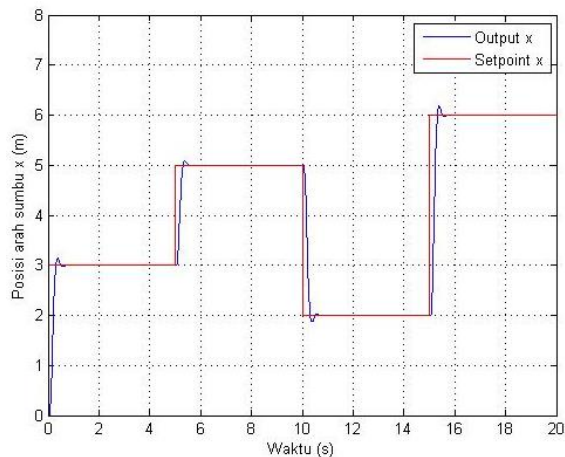
Parameter SFS yang digunakan diperlihatkan pada Tabel 2. Karena matriks Q dan R harus positif definit, maka ruang pencarian tidak boleh negatif. Matriks Q dan R yang digunakan adalah matriks diagonal, dengan dimensi 12 x 12 untuk Q dan 4 x 4 untuk R sehingga satu buah partikel (P) berada pada ruang vektor dengan dimensi 16.

Tabel 2. Parameter SFS

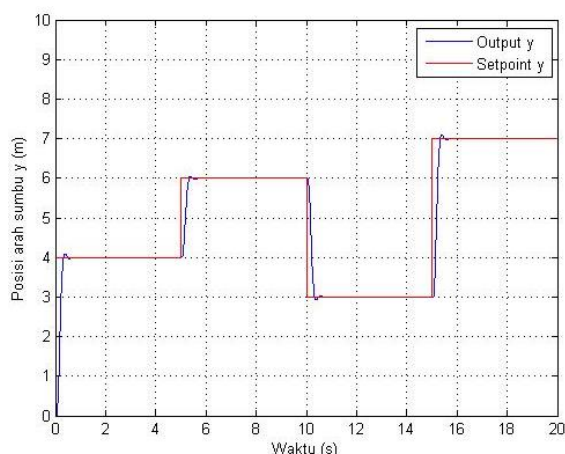
LB	0,01
UB	1000
M	100
D	5
G	100
Probabilitas GW_1	0,3
Dimensi vektor	16

Setelah melakukan 100 iterasi, didapatkan nilai matriks Q dan R yang menghasilkan sinyal kontrol optimal. Respons *Quadrotor* (posisi x, y , dan z)

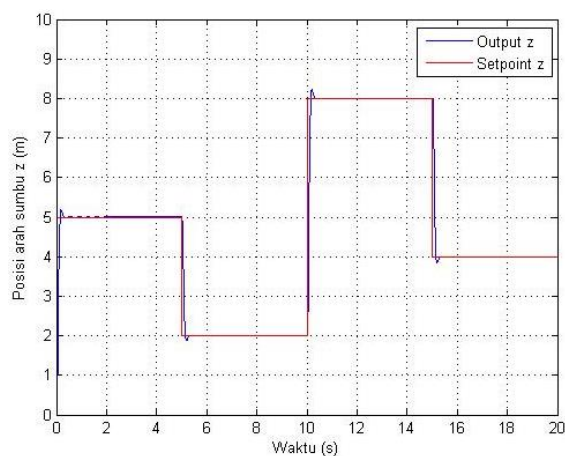
terhadap beberapa nilai *set-point* yang berbeda diperlihatkan pada Gambar 2, Gambar 3, dan Gambar 4.



Gambar 2. Posisi x *Quadrotor* dengan beberapa perubahan nilai *set-point*



Gambar 3. Posisi y *Quadrotor* dengan beberapa perubahan nilai *set-point*



Gambar 4. Posisi z *Quadrotor* dengan beberapa perubahan nilai *set-point*

Berdasarkan hasil simulasi, pengontrol LQR mampu mengontrol posisi *Quadrotor* dengan baik. Nilai IAE akhir yang didapatkan adalah 630,7 dengan persentase *maximum overshoot* respon terbesar hanya 2,81% dan *settling time* kurang dari 1 detik. Hal ini menunjukkan superioritas dari algoritma SFS dalam mengoptimisasi parameter pengontrol LQR. Hanya dengan 100 iterasi, algoritma SFS telah mendapatkan nilai optimal untuk matriks Q dan R dalam ruang pencarian yang besar.

4 Kesimpulan

Suatu alternatif teknik perancangan pengontrol LQR telah dikembangkan pada penelitian ini dengan menggunakan algoritma SFS, sehingga diperoleh matriks perancangan Q dan R yang optimal dalam waktu yang relatif cepat pada ruang pencarian yang relatif besar. Metode LQR yang dikembangkan ini kemudian diterapkan untuk mengontrol suatu model kendaraan tanpa awak (*Quadrotor*). Hasil simulasi menunjukkan bahwa *Quadrotor* dapat mengikuti beberapa perubahan *set-point* yang diberikan dengan performansi yang sangat baik. Penelitian ini menunjukkan superioritas dari algoritma SFS, yang dapat menjadi salah satu alternatif metode untuk memecahkan masalah optimisasi, terutama dalam bidang kontrol.

5 Daftar Pustaka

- [1] J. D. Lopez, J. J. Espinosa, and J. R. Agudelo, "LQR control for speed and torque of internal combustion engines," *Proceed. of the 18th World Congress, The International Federation of Automatic Control (IFAC), Milano, Italy, August 28th - September 2nd, 2011, Vol. 44*, pp. 2230-2235
- [2] M. Shweda, J. L. Nandagopal, and S. Amritha, "Coupled dynamic control of unicycle robot using integral linear quadratic regulator and sliding mode controller," *Proceed. of the International Conference on Processing of Materials, Minerals, and Energy, Ongole, Andhra, Pradesh, India, July 29th - 30th, 2016, Vol. 5*, pp. 1447-1454
- [3] P. Escarate, J. C. Aguero, S. Zuniga, M. Castro, and J. Garces, "Linear quadratic regulator for laser beam shaping," *Journal of Optics and Lasers in Engineering*, 2017, Vol. 94, pp. 90-96
- [4] B. Friedland, "Control system design: an introduction to state space methods," Dover, 2004
- [5] H. Salimi, "Stochastic fractal search: a powerful metaheuristic algorithm," *Journal of*

- Knowledge-Based Systems, 2015, Vol. 75, pp. 1-18
- [6] F. Sabatino, "Quadrotor control: modeling, nonlinear control design, and simulation," Master's Degree Project, KTH Electrical Engineering, Stockholm, Sweden, June, 2015
- [7] B. Anditio, A. D. Andrini and Y. Y. Nazaruddin, "Integrating PSO optimized LQR controller with virtual sensor for quadrotor position control," in IEEE Conference on Control Technology and Application, Copenhagen, Denmark, August 21st -24th, 2018