Jurnal Ilmiah Mustek Anim Ha Vol. 6 No. 3, Desember 2017
ISSN 2089-6697

SIMULASI TUNNING SISTEM KENDALI PID QUADROTOR MENGGUNAKAN ALGORITMA GENETIKA

Nasra Pratama Putra[1], Fransiskus Xaverius Manggau[2]

Email: nasrapratama@unmus.ac.id , fransiskus@unmus.ac.id

Jurusan Sistem Informasi, Fakultas Teknik

Universitas Musamus

ABSTRAK

Sistem kendali dibutuhkan quadrotor agar dapat melayang mendekati keadaan stasioner. Salah satu sistem kendali yang dapat dirancang dan diimplementasikan pada quadrotor adalah kendali PID. Metode tradisional yang popular untuk kendali PID adalah menggunakan *Ziegler Nichol*. Kelemahan metode tersebut sering menghasilkan gelombang overshoot tinggi. Di sisi lain, beberapa pendekatan *intelligence* juga diusulkan untuk meningkatkan hasil tuning tradisional PID seperti menggunakan optimalisasi *heuristic*.

Optimalisasi *heuristic* merupakan sebuah teknik pencarian solusi terbaik dalam dunia komputasi dengan ketetapan yaitu tidak memiliki batasan dalam mencapai nilai optimal suatu parameter tertentu. Salah satu metode yang termasuk dalam optimalisasi *heuristic* adalah Algoritma Genetika. *Tunning* PID menggunakan simulasi berbasis Algoritma Genetika dapat memepercepat proses pencarian konstanta PID yang optimal. Parameter-parameter yang digunakan pada simulasi ini yaitu massa, panjang lengan, radius, torsi motor, dan kecepatan motor. Beberapa asumsi yang diterapkan dalam melakukan simulasi dari quadrotor ini yaitu, struktur dari quadrotor dianggap kaku, struktur dari quadrotor dianggap simetris, titik berat beban quadrotor diasumsikan berada tepat di tengah (pusat massa) quadrotor, dan efek getaran masing-masing propeller dianggap tidak terjadi.

Hasil perhitungan simulasi sistem menggunakan Algoritma Genetika didapatkan konstanta PID yang optimal yaitu Kp bernilai 0.0996, Ki bernilai 0.001 dan Kd bernilai 0.0289. memiliki nilai maksimum peak yaitu 3,57 derajat. Artinya quadrotor akan stabil melayang dengan tetap dapat menjaga sudut kemiringan tidak lebih dari 3,88 derajat. Nilai *settling time* yang didapat pada kondisi tersebut adalah 1,54 detik. Artinya tingkat kemiringan sudut akan mengalami penurunan hingga menuju keadaan stationer dengan waktu tempuh sekitar 1,54 detik. Jumlah generasi untuk mencapai kondisi tersebut adalah sekitar 26 generasi. Waktu eksekusi simulasi untuk kondisi tersebut adalah 494 detik.

Kata Kunci: Quadrotor, PID, Algoritma Genetika

[1] Dosen Jurusan Teknik Sistim Informasi, Universitas Musamus Merauke
[2] Dosen Jurusan Teknik Informatika, Universitas Musamus Merauke

PENDAHULUAN

Quadrotor atau sering disebut juga *quadcopter* merupakan jenis *Unmanned Aerial Vehicle* (UAV) yang terbang dengan menggunakan empat motor dan empat buah propeller pada masing-masing lengannya (Dharmawan dan Putera, 2012). Gerakan *quadrotor* dihasilkan dengan menggunakan gaya dorong dari perputaran tiap motor. Pergerakan tersebut diatur oleh sistem kontrol *attitude*. Untuk menghasilkan *attitude* yang baik, diperlukan aturan pengendali yang benar dan pemodelan sistem. Dua metode yang berbeda telah diketahui untuk mencapai dinamika *quadrotor* yakni persamaan Lagrange dan hukum II Newton(Dharmawan dkk., 2014). Hasil pemodelan dinamika *quadrotor* secara langsung dipengaruhi oleh sinyal masukan, sehingga dibutuhkan aturan sistem kendali lebih lanjut agar dapat menghasilkan kondisi *output* yang sesuai dengan kondisi *plant.* Salah satu sistem kendali yang dapat dirancang dan diimplementasikan pada *quadrotor* adalah kendali PID(Bouabdallah dkk., 2004).

Kendali PID memiliki evektivitas yang luar biasa, ketahanan yang kuat dan kesederhanaan implementasi. Terdapat 3 buah *gain* yang bekerja pada *error* yaitu Kp, Ki, dan Kd. Pemilihan nilai Kp, Ki, dan Kd sangat mempengaruhi tingkat optimal dari kendali PID (Priyambodo dkk., 2015). Ketika seluruh nilai konstanta PID tidak tepat, maka akan menghasilkan respon sistem yang menyimpang dari hasil yang diinginkan. Untuk itu dibutuhkan metode *tuning* Kp, Ki, dan Kd yang akurat untuk mendapatkan respon sistem yang sesuai. Terdapat dua metode *tunning* PID, yaitu menggunakan pendekatan tradisional dan *intellegence*. Metode tradisional yang popular adalah menggunakan *Ziegler Nichol*. Di sisi lain, beberapa pendekatan *intelligence* juga diusulkan untuk meningkatkan hasil *tunning* tradisional PID seperti menggunakan optimalisasi berbasis evolusi alam (*Evolutionary Computation*).

Evolutionary Computation merupakan metode yang meniru cara kerja teori evolusi Charles Darwin yaitu "*Survival of the fittest*". (Sivanandam dan Deepa, 2008). Salah satu metode yang termasuk dalam *evolutionary computation* adalah Algoritma Genetika. Algoritma Genetika merupakan metode yang menerapkan teknik optimasi berbasis evolusi alam yaitu melalui proses mutasi, pindah silang dan seleksi(Jones dan de Moura Oliveira, 1995). Pada kasus sistem kendali PID, ketiga nilai dari parameter PID yang tidak diketahui yaitu Kp, Ki, dan Kd

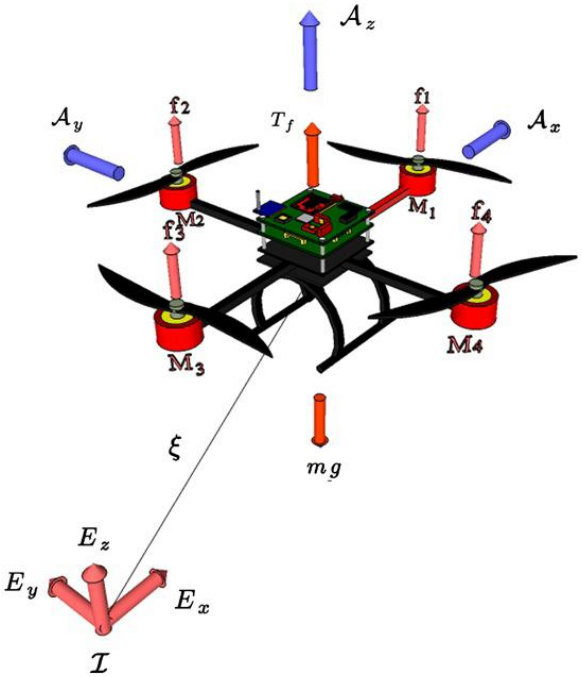
dikodekan sebagai sebuah kromosom dalam bentuk kode biner. Kode biner tersebut dievaluasi dengan suatu fungsi objektif berdasarkan biaya dalam *time domain.* Nilai *cost* yang dianggap memiliki *fitness* tertinggi akan dipilih sebagai kromosom terbaik. Kromosom yang mampu bertahan hingga akhir generasi dikodekan kembali menjadi konstanta Kp, Ki dan Kd yang optimal.

TINJAUAN PUSTAKA

1. Analisis Pemodelan

Quadrotor merupakan kendaraan tanpa awak dengan mengunakan empat buah rotor agar mampu terbang. Quadrotor menempatkan keempat buah motor pada sisi lengan dengan jarak dari titik tengah sudah ditentukan. Quadrotor tipe plus (+) baling-baling memiliki arah putaran sama diletakkan bersebrangan. Motor 1 dan 3 memiliki arah putaran *clockwise*, sedangkan motor 2 dan 4 memiliki arah putaran *counter clockwise.*

Kombinasi dari gerak translasi dan rotasi merupakan gerakan dasar quadrotor di udara. Berdasarkan Gambar 1, maka didefinisikan *I* sebagai *earth fix frame* dan *A* sebagai *body fix frame*. Pusat massa dan *body fix frame* diasumsikan bertepatan. Dengan menggunakan parameter sudut *euler*, orientasi *quadrotor* di udara didapat dari rotasi R dari A ke I, dimana $\boldsymbol{R} \epsilon SO(3)$ yang merupakan rotasi matrix. *fi* mewakili *thrust* dari motor ke i dan *Tf* adalah *main thrust*.



Gambar 1 Quadrotor Inersia *Frame*

Menggunakan pendekatan *Newton-Euler*, dinamika dari *quadrotor* dinyatakan pada persaman (1) (2) (3) (4)

$$\dot{\xi} = v \quad (1)$$
$$m\dot{v} = f \quad (2)$$
$$\dot{R} = R\hat{\Omega} \quad (3)$$
$$I\dot{\Omega} = -\Omega \mathrm{x}\, I\Omega + \tau \quad (4)$$

Dimana $\xi = (x, y, z)^T$ yang menunjukan posisi dari pusat massa dari *quadrotor* sama dengan pusat *I*. $v \epsilon I$ menunjukkan kecepatan linear

dinyatakan dalam kerangka inersia. $\Omega \epsilon A$ menunjukkan kecepatan sudut dari badan *quadrotor* yang dinyatakan dalam *body fix frame. m* melambangkan massa dari body kaku. $I \epsilon R^{3\,x\,3}$ menunjukkan matriks inersia konstan di sekitar pusat massa. $f \epsilon I$ merupakan vector dari gaya tidak tetap pada objek, termasuk *thrust Tf* dan gesekan pada rotor. $\tau \epsilon A$ merupakan perbedaan *thrust* antar rotor dengan efek aerodinamik dan gyroscopic.

Momen inersia *quadrotor* dinyatakan pada persamaan (5) (6) (7).

$$I_{xx} = \frac{mr^2}{4} + \frac{mh^2}{6} + 2mr^2 + \frac{MR^2}{4} + \frac{MH^2}{12} \quad (5)$$

$$I_{yy} = \frac{mr^2}{4} + \frac{mh^2}{6} + 2mr^2 + \frac{MR^2}{4} + \frac{MH^2}{12} \quad (6)$$

$$I_{zz} = \frac{MR^2}{2} + 4mr^2 \quad (7)$$

M adalah massa kotak tengah *quadrotor*, m adalah massa motor *brushless*, R adalah jarak antara ujung bagian tengah *quadrotor* ke pusat massa, r adalah jari-jari motor *brushless*, h adalah tinggi motor *brushless*, H adalah tinggi bagian tengah *quadrotor*, dan l adalah panjang antara ujung *quadrotor* ke pusat massa.

Persamaan akhir untuk percepatan sudut $\ddot{\phi}, \ddot{\theta}, dan\, \ddot{\psi}$ diliniearisasi berdasarkan keadaan melayang dinyatakan pada persamaan(8) (9) (10).

$$\frac{\phi(s)}{F(s)} = \frac{l}{I_{xx}s^2} \quad (8)$$

$$\frac{\theta(s)}{F(s)} = \frac{l}{I_{yy}s^2} \quad (9)$$

$$\frac{\psi(s)}{F(s)} = \frac{l}{I_{zz}s^2} \quad (10)$$

2. Penentuan Fugsi Transfer

$G_{PID}(s)$ merupakan PID *Controller* domain Laplace, $G_x(s)$ merupakan fungsi transfer pada sudut *roll*, $G_y(z)$ merupakan fungsi transfer pada sudut *pitch*, dan $G_z(s)$ merupakan fungsi transfer pada sudut *yaw*. Ketiga blok diagram di atas dapat disederhanakan menggunakan aturan *unityfeedback*. Untuk sumbu x disederhanakan menjadi $G_{xuf}(s)$ yang dijabarkan dalam persamaan (11) dan persamaan (12).

$$G_{xuf}(s) = \frac{G_{PID\phi}(s)G_x(s)}{1 + G_{PID\phi}(s)G_x(s)} \quad (11)$$

$$G_{xuf}(s) = \frac{K_{D\phi}ls^2 + K_{P\phi}ls + K_{i\phi}l}{I_{xx}s^2 + K_{D\phi}ls^2 + K_{P\phi}ls + K_{i\phi}l} \quad (12)$$

Untuk sumbu y disederhanakan menjadi $G_{yuf}(s)$ yang dijabarkan dalam persamaan (13) dan persamaan (14).

$$G_{yuf}(s) = \frac{G_{PID\theta}(s)G_y(s)}{1 + G_{PID\theta}(s)G_y(s)} \quad (13)$$

$$G_{yuf}(s) = \frac{K_{D\theta}ls^2 + K_{P\theta}ls + K_{i\theta}l}{I_{yy}s^2 + K_{D\theta}ls^2 + K_{P\theta}ls + K_{i\theta}l} \quad (14)$$

Untuk sumbu z disederhanakan menjadi $G_{zuf}(s)$ yang dijabarkan dalam persamaan (15) dan persamaan(16).

$$G_{zuf}(s) = \frac{G_{PID\psi}(s)G_z(s)}{1 + G_{PID\psi}(s)G_z(s)} \quad (15)$$

$$G_{zuf}(s) = \frac{K_{D\psi}ls^2 + K_{P\psi}ls + K_{i\psi}l}{I_{zz}s^2 + K_{D\psi}ls^2 + K_{P\psi}ls + K_{i\psi}l} \quad (16)$$

Dari ketiga fungsi transfer *unityfeedback* di atas diharapkan kombinasi yang tepat antara Kp, Ki, Kd dapat meminimalkan nilai *peak* dan *settling time.*

3. Komponen Quadrotor

Simulasi quadrotor dapat dibangun dengan memperhatikan data-data dari komponen yang akan digunakan. Beberapa komponen utama dalam menyusun sebuah simulasi *quadrotor* terdapat pada Tabel 1.

Tabel 1. Komponen *quadrotor*

No.	Nama	Jumlah
1.	*Brushless motor*	4
2.	*Electronic Speed Controller (ESC)*	4
3.	Kerangka *quadrotor*	1
4.	*Arduino Due*	1
5.	Baterai *LiPo*	1
6.	*Sensor Accelerometer*	1
7.	*Sensor Gyroscope*	1
8.	*Sensor Magnetometer*	1
9.	*Propeller*	4
10.	Modul RF	1
11.	Sensor *Ultrasonic Range*	1

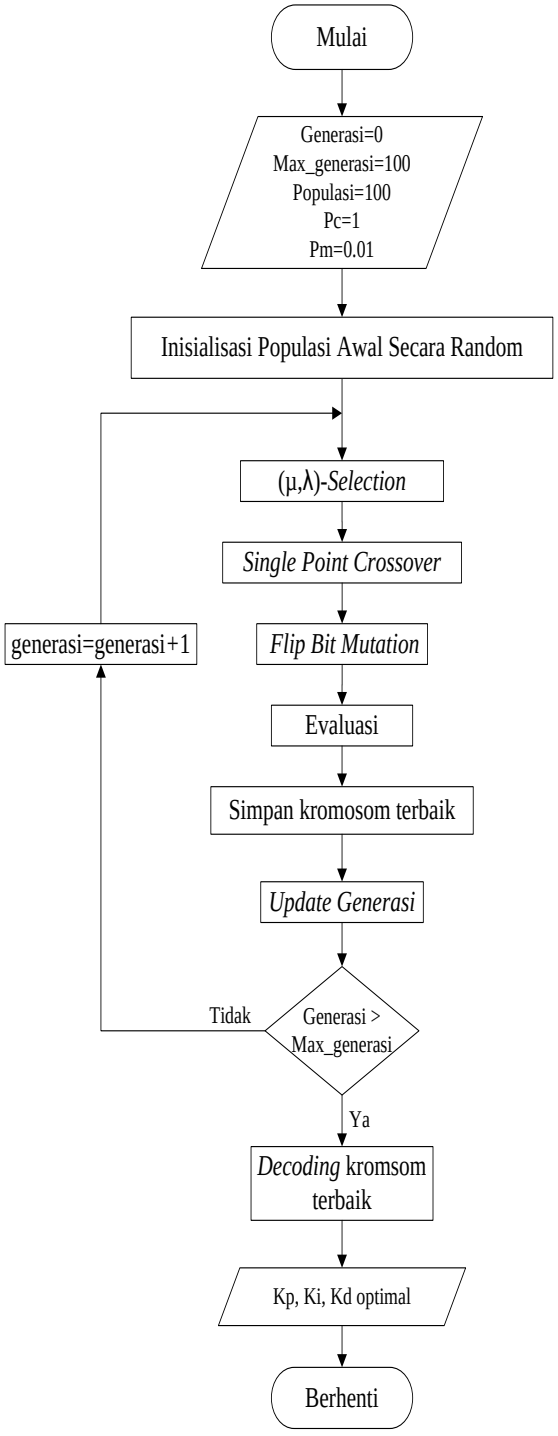
4. Algoritma Genetika

Algoritma genetika atau *Genetic Algorithm* (GA) merupakan metode optimasi dan teknik pencarian yang menerapkan prinsip genetik dan seleksi alamiah. *Algoritma Genetika* pertama kali diperkenalkan oleh John Holland pada tahun 1975 dalam bukunya "*Adaptation in natural and artificial systems*" (Saad dkk., 2012).

Terdapat dua jenis transformasi. *Mutation* merupakan proses transformasi untuk menciptakan individu baru dengan melakukan perubahan pada satu individu. Sedangkan *crossover* merupakan proses transformasi untuk menciptakan individu baru dengan menggabungkan bagian-bagian dari dua individu. Setelah melalui tahapan transformasi, maka dilakukan evaluasi terhadap beberapa individu baru yang disebut anak atau keturunan C(t).

Sebuah populasi baru dapat terbentuk dengan melakukan proses seleksi. Seleksi bertujuan untuk memilih individu dari populasi induk dan populasi anak berdasarkan nilai *fitness*. Setelah melalui beberapa generasi, akan ditemukan individu terbaik. Individu terbaik dapat dikatakan sebagai solusi optimal atau suboptimal untuk masalah yang sedang dihadapi. Berikut adalah tahapan-tahapan dalam

simulasi *tunning* PID menggunakan Algoritma Genetika.



Gambar 2. Flowchart *tunning* PID menggunakan Algortima Genetika

METODOLOGI PENELITIAN

Metode penelitian yang digunakan menggunakan Algoritma Genetika dengan mengacu pada konsep SDLC, dengan tahapan antara lain:

1. Analisis Sistem

Pada tahap ini dilakukan analisis terhadap model quadrotor yang akan digunakan. Merumuskan kembali persamaan sudut dan fungsi transfer PID yang telah dilinierisasi berdasarkan keadaan melayang dari quadrotor.

2. Spesifikasi Kebutuhan Sistem

Dalam rangka membangun simulasi, tentu dibutuhkan beberapa nilai komponen dari quadotor. Beberapa komponen quadrotor tang digunakan dijabarkan pada Tabel 1.

3. Perancangan simulasi

Perancangan simulasi dibangun dengan menggunakan Metode Algoritma Genetika. Beberapa tahapan utama dari Algoritma Genetika seperti insisalisasi, seleksi, mutasi dan evaluasi dijabarkan pada Gambar 2. Untuk jumlah sampling pencarian nilai terbaik dilakukan dengan menggunakan ukuran generasi sebanyak 100 kali. Penentuan nilai terbaik ditentukan berdasarkan ukuran *fitness* yang paling bugar. Nilai Fitness didapat melalui fungsi objektif. Pada penelitian ini, fungsi objektif didapat dengan mencari nilai

terkecil dari penjumlahan *overshoot*, *rise time*, dan *settling time* dari setiap individu.

4. Implementasi dan Pengujian Sistem

Simulasi sistem dibangun dengan menggunakan data awal yang diperoleh melalui perhitungan PID konvensional *Ziegler Nichols*. Software yang digunakan untuk membangun simulasi adalah Matlab 2013. Pada proses pengujian, nilai ambang atas dan bawah dari variable P, I, dan D dapat dipersempit dan diperluas guna menemukan nilai fitness yang paling sesuai.

HASIL DAN PEMBAHASAN

1. Implementasi Simulasi

Tahap awal adalah memasukan parameter yang berpengaruh terhadap fungsi transfer. Kode program Matlab untuk melakukan insialisasi parameter awal ditunjukan pada Gambar 3.

```
1    t=0;
2    %inisiisasi populasi awal
3    a_kp=0.125;
4    a_ki=0.075;
5    a_kd=0.075;
6    b_kp=0.075;
7    b_ki=0.001;
8    b_kd=0.001;
9    presisi=3;
10   L_kp=ceil(log2(((a_kp-
     b_kp)*(10^presisi))+1));
11   L_ki=ceil(log2(((a_ki-
     b_ki)*(10^presisi))+1));
12   L_kd=ceil(log2(((a_kd-
     b_kd)*(10^presisi))+1));
13   L=L_kp+L_ki+L_kd;
14   Npop=50;
15   Niter=100;
16   pc=1;
17   pm=0.01;
18   Nmuts=ceil(pm*(Npop-1)*L);
populasi = round(rand(Npop,L));
```

Gambar 3 Kode program Inisialisasi

Setelah dilakukan inisialisai parameter awal, maka dibentuklah populasi awal. Pada penelitian ini populasi awal dibentuk dari kumpulan kode biner sepanjang L, dan sebanyak jumlah populasi.

Proses *selection* digunakan untuk mencari nilai-nilai *fitness* yang tertinggi untuk dijadikan sebagai *parent* pada proses evolusi. Metode *selection* pada penelitian ini menggunakan metode seleksi (μ,λ). Metode ini dimulai dengan melakukan pengurutan individu dalam populasi berdasarkan besar nilai *fitness*.

Individu dengan nilai *fitness* tertinggi akan menempati urutan pertama dalam populasi *matting poll*. Jumlah terbanyak kromosom (induk) yang

diperbolehkan melakukan reproduksi sejumlah setengah ukuran populasi. Kode program Matlab untuk melakukan proses *selection* ditunjukan pada Gambar 4.

```
%SELECTION
1  mattingpoll=[];
2  indfit=[];
3  [fitness,indfit]=sort(fitness,'
   descend');
4  mattingpoll
   =populasi((indfit(1:size(popula
   si,1)/2)),:);
```

Gambar 4 Kode program (μ,λ)-*Selection*

Setelah populasi mengalami proses *selection*, maka tahapan selanjutnya adalah melakukan proses *crossover*. Proses *crossover* dimulai dengan membangkitkan bilangan acak *r* dalam rentang [0..1]. Jika *r* < *Pc*, maka individu tersebut diperbolehkan melakukan proses reproduksi. Untuk setiap individu yang terpilih, dilakukan proses *Single Point Crossover*.

Proses *Single Point Crossover* dilakukan dengan cara memilih satu titik secara acak pada kromosom individu. Kemudian individu yang telah terpilih dapat melakukan pertukaran gen terhadap individu terpilih berikutnya. Kode program Matlab untuk proses *crossover* ditunjukan pada Gambar 5.

```
%Reproduksi
1   parent=[];
2   j=1;
3   for k=1:size(mattingpoll,1)
4   rk=rand();
5   if rk<pc
6   parent(j)=k;
7   j=j+1;
8   end
9   end
%crossover
10  offspring=[];
11  for ic=1:2:length(parent)
12  if ic>=length(parent)
13  break;
14  else
15  cross=ceil((L-1)*rand());
16  offspring(ic,1:cross)=matting
    poll(parent(ic),1:cross);
17  offspring(ic,cross+1:L)=matti
    ngpoll(parent(ic+1),cross+1:L
    );
18  offspring(ic+1,1:cross)=matti
    ngpoll(parent(ic+1),1:cross);
19
    offspring(ic+1,cross+1:L)=mat
    tingpoll(parent(ic),cross+1:L
    );
20          end
21      end
22
```

Gambar 5 Kode program *Single Point Crossover*

Setelah populasi mengalami proses *crossover*, maka tahapan selanjutnya adalah melakukan proses *mutation*. Proses *mutation* yang akan digunakan adalah *mutation flib bit*, dimana merubah nilai 1 menjadi 0 dan 0 menjadi 1. Kode program Matlab untuk *mutation* ditunjukan pada Gambar 6.

```
%MUTATION
1.  Nofspr= size(offspring,1);
2.  for im=1:Nmuts
3.  ix=ceil(size(mattingpoll,1)*ra
    nd);
4.  iy=ceil(L*rand);
5.  offspring(Nofspr+im,:)=matting
    poll(ix,:);
6.  offspring(Nofspr+im,iy)=1-
    offspring(Nofspr+im,iy);
7.  end
```

Gambar 6 Kode program *mutation* biner

Setelah didapat kromosom keturunan hasil dari proses *crossover* dan *mutation*, maka langkah selanjutnya adalah evaluasi. Setiap kromosom anak tersebut diubah kedalam bentuk *fenotype*, yaitu dalam bentuk nilai riil.

Setiap kromosom yang telah diketahui nilai Kp, Ki, dan Kd dalam bentuk desimal akan dimasukkan ke dalam fungsi transfer. *GC* merupakan *gain* dari sistem kontrol PID. *GYD* merupakan *disturbance* dari sistem, yaitu sebesar fungsi transfer dibagi dengan *gain* kontrol. Selisih fungsi transfer dengan *disturbance* pada sudut *pitch*, sudut *roll*, dan sudut *yaw* disesuaikan dengan persamaan (12), (14), dan (16). Sedangkan untuk nilai maksimum *peak* dari sistem dengan menggunakan fungsi *impulse*. *Settling Time* didapat dari waktu yang dibutuhkan oleh respon dalam mencapai nilai stabil, yaitu waktu yang dibutuhkan sinyal PID untuk kembali ke titik nol setelah menerima gangguan.

Setelah diketahui *fitness* kromosom anak yang akan menjadi induk pada generasi selanjutnya, maka tahapan selanjutnya adalah melakukan proses *update generation*. Proses *update generation* akan memastikan kromosom anak kembali ke populasi dan menggantikan kromosom induk yang memiliki *fitness* terendah pada generasi sebelumnya. Untuk hal itu perlu dilakukan seleksi terhadap kromosom induk yang bertahan, dan sisanya akan langsung digantikan oleh kromosom anak hasil proses seleksi (μ,λ). Kode program Matlab untuk melakukan proses *update generation* ditunjukan pada Gambar 7.

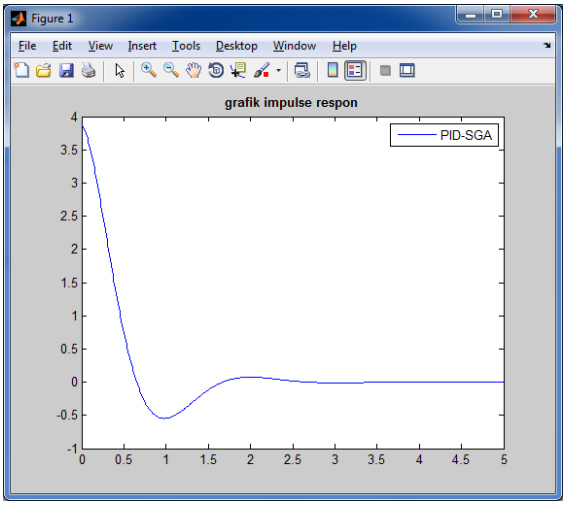
```
%sort fitness offspring terbaik
1  [fitness,indoff]=sort(fitness,'
   descend');
%update generasi
2  j=1;
3  for
   i=(size(populasi,1)/2)+1:size(p
   opulasi,1)
4  populasi(indfit(i),:)=offspring
   (indoff(j),:);
5  j=j+1;
6  end
```

Gambar 7 Kode program *update generation*

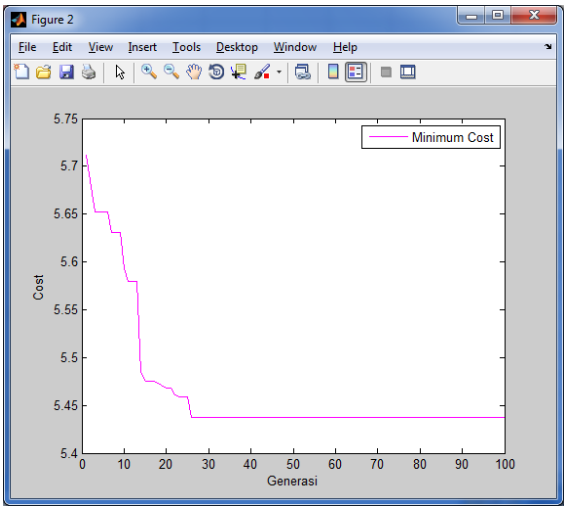
2. Pengujian Simulasi

Hasil simulasi *tunning* menunjukkan Ketika Kp bernilai 0.0996, Ki bernilai 0.001 dan Kd bernilai 0.0289

memiliki nilai maksimum *peak* yaitu 3,57 derajat. Artinya *quadrotor* akan stabil melayang dengan tetap dapat menjaga sudut kemiringan tidak lebih dari 3,88 derajat. Nilai *settling time* yang didapat pada kondisi tersebut adalah 1,54 detik. Artinya tingkat kemiringan sudut akan mengalami penurunan hingga menuju keadaan stationer dengan waktu tempuh sekitar 1,54 detik. Jumlah generasi untuk mencapai kondisi tersebut adalah sekitar 26 generasi. Waktu eksekusi simulasi untuk kondisi tersebut adalah 494 detik. *Impulse response* sudut *pitch* dan *roll* beserta progress *tunning* tiap generasi ditunjukkan oleh Gambar 7 dan Gambar 8.



Gambar 7 *Impulse response* hasil *tunning* PID dengan Algoritma Genetika



Gambar 8 Minimum *cost* tiap generasi dengan Algoritma Genetika

PENUTUP

1. Kesimpulan

Berdasarkan hasil perhitungan dan simulasi maka diambil kesimpulan bahwa telah berhasil dibuat simulasi *tunning* sistem kendali PID menggunakan Algoritma Genetika pada sebuah *quadcopter* yang ideal, dimana tidak dipengaruhi oleh efek aerodinamis dan efek giroskopik.

Sedangkan untuk jumlah generasi dalam mencapai kondisi tersebut adalah sekitar 26 generasi dari total 100 generasi sehingga waktu eksekusi simulasi untuk kondisi tersebut hanya memakan waktu 494 detik.

2. Saran

Nilai Kp, Ki, dan Kd hasil *tunning* dengan menggunakan simulasi perlu

dilakukan perbandingan kembali pada kondisi yang sesungguhnya untuk menemukan faktor-faktor lain yang dapat menyebabkan perbedaan hasil.

DAFTAR PUSTAKA

1. Dharmawan, A. dan Putera, C.A.L., 2012. Purwarupa Sistem Integrasi Quadcopter dan Mobile Robot. *IJEIS - Indonesian Journal of Electronics and Instrumentation Systems*, **2**: 97–108.
2. Dharmawan, A., Simanungkalit, Y.Y., dan Megawati, N.Y., 2014. Pemodelan Sistem Kendali PIDpada Quadcopterdengan Metode Euler Lagrange. *IJEIS - Indonesian Journal of Electronics and Instrumentation Systems*, **4**: 13–24.
3. Bouabdallah, S., Noth, A., dan Siegwart, R., 2004. 'PID vs LQ control techniques applied to an indoor micro quadrotor', dalam: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings*. Dipresentasikan pada 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings, hal. 2451–2456 vol.3.
4. Priyambodo, T.K., Putra, A.E., dan Dharmawan, A., 2015. 'Optimizing control based on ant colony logic for Quadrotor stabilization', dalam: *2015 IEEE International Conference on Aerospace Electronics and Remote Sensing Technology (ICARES)*. Dipresentasikan pada 2015 IEEE International Conference on Aerospace Electronics and Remote Sensing Technology (ICARES), hal. 1–4.
5. Sivanandam, S.N. dan Deepa, S.N., 2008. *Introduction to Genetic Algorithms*. Springer Berlin Heidelberg, Berlin, Heidelberg.
6. Jones, A.H. dan de Moura Oliveira, P.B., 1995. 'Genetic auto-tuning of PID controllers', dalam: *Genetic Algorithms in Engineering Systems: Innovations and Applications, 1995. GALESIA. First International Conference on (Conf. Publ. No. 414)*. Dipresentasikan pada Genetic Algorithms in Engineering Systems: Innovations and Applications, 1995.
7. Saad, M.S., Jamaluddin, H., dan Darus, I.Z.M., 2012. Implementation of PID controller tuning using differential evolution and genetic algorithms. *International Journal of Innovative Computing Information and Control*, 8: 7761–7779.