DAFTAR PUSTAKA

K Ogata K. 2002. *Modern control engineering*. 4th ed., Prentice Hall, Aeeizh.

J. Kennedy and R. Eberhart. 1995 *Particle Swarm Optimization*. Proceedings of IEEE International Conference on Neural Networks.

Bharti J., Phadke G., Patil D. 2020. *Optimization of DC Motor Speed Control Using LQR Technique*. In: Kumar A., Paprzycki M., Gunjan V. (eds) ICDSMLA 2019. Lecture Notes in Electrical Engineering, Springer, Singapore, vol 601, pp. 1492-1499.

Arnisa Myrtellari, Petrika Marango, and Margarita Gjonaj. 2016. *Analysis and Performance of Linear Quadratic Regulator and PSO algorithm in optimal control of DC motor*. International Journal of Latest Research in Engineering and Technology (IJLRET). ǁ Vol. 2, No. 4, pp. 88-93.

A.P. Engelbrecht. 2007. *Computational Intelligence: An Introduction. 2nd ed*., John Wiley & Sons, Ltd.

Debabrata Pal. 2016. *Modelling, Analysis, and Design of a DC Motor based on State Space Approach*. International Journal of Engineering Research & Technology (IJERT), Vol. 5, Issue 02.

P. M. Meshram and R. G. Kanojiy. 2012. *Tuning of PID Controller using Ziegler-Nichols Method for Speed Control of DC Motor*. IEEE- International Conference On Advances In Engineering, Science And Management (ICAESM - 2012), pp. 117–122.

Wang, D., Tan, D. & Liu, L. 2018. *Particle swarm optimization algorithm: an overview*, Soft Computing, Vol. 22, pp. 387–408.

Yan HE, Wei Jin Ma, dan Ji Ping Zhang. 2016. *The Parameters Selection of PSO Algorithm Influencing on Performance of Fault Diagnosis*. MATEC Web of Conferences, 63.

HE, yan. Wei Jin MA, dan Ji Ping ZHANG. 2016. , MATEC Web of Conferences.*The Parameter Selection of PSO Algorithm Influencing on Performance of Fault Diagnosis*

Abut, Tayfun. 2016. *Modeling and Optimal Control of DC Motor*, International Journal of Engineering Trends and Technology (IJETT), Vol. 32, No.3, pp 146-150.

Mohammed, Nadia Qasim. 2017. *DC Motor Drive with P, PI, and Particle Swarm Optimization Speed Controllers*. International Journal of Computer Applications. Vol. 166, No.12, pp. 42-4.

Ali A. Hassan et al. 2018. *Comparative Study for DC Motor Speed Control Using PID Controller.* International Journal of Engineering and Technology (IJET). Vol .9, No. 6, pp. 4181-4192.

LAMPIRAN

Lampiran 1 Skrip Simulasi Kendali Motor DC LQR

```
%parameter motor 3

rm = 2;

lm = 0.5;

j  = 1.2

b  = 0.2;

kt = 0.2;

km = 0.2;

   A= [-b/j kt/j ; -km/lm -rm/lm];

   B = [0; 1/lm];

   C = [1 0];

   D =  [0];

   Q = transpose(C)*(C);

   R = [0.01];

kf = 1/dcgain(A,B,C,D);

   k =lqr(A,B,Q,R);

   k1 = k(:,1);

   k2 = k(:,2);

   k = [k1 ; k2];

%run simulation

simout = sim('motordclqr', 'SimulationMode',
'normal','AbsTol','1e-5','SaveState',
'on','StateSaveName','xout','SaveOutput','on','OutputSaveName','
yout','SaveFormat','Array');

        output = simout.get('yout');

        time = simout.get('tout');
```

```
kec = output(:,1);

curr = output(:,2);

ref = output(:,4);

waktuta = output(:,3) ;

%plotting

% plot(waktuta, ref)

% hold on;

plot(waktuta,kec, 'LineWidth',2)

ylim([0 6])

xlim([0 100])

title(["LQR Controlled"])

xlabel('time(s)')

ylabel('ω(rad/s)')

grid on;

hold on;

plot(waktuta, ref)

stepinfo(kec, waktuta)

%     simout=sim('motordclqr');

%     kecepatan = simout.get('kecepatan');

%     arus = simout.get('arus');

%     time = simout.get('tout');

%     u = simout.get('u');

%     x = [kecepatan arus];

%     xtopi = transpose(x);

%     utopi= transpose(u);

%     figure(1)

%     plot(time,kecepatan), grid on
```

```
%

%    J =sum( sum(x*Q*xtopi) + sum(u*r*utopi));
```

Lampiran 2 Skrip Simulasi Kendali Motor DC PID - ZN

```
% %parameter

% rm = 4;

% lm = 0.01;

% j  = 0.0044;

% b  = 0.0011;

% kt = 0.22;

% km = 0.22;

rm = 2;

lm = 0.5;

j  = 1.2

b  = 0.2;

kt = 0.2;

km = 0.2;

%run simulation

simout = sim('tanpa_pengendali', 'SimulationMode',
'normal','AbsTol','1e-5','SaveState',
'on','StateSaveName','xout','SaveOutput','on','OutputSaveName','
yout','SaveFormat','Array');

        output = simout.get('yout');

        time = simout.get('tout');


%parsing

kec = output(:,1);

curr = output(:,2);

K = kec(end);
```

```
L_index = find(kec>=.05*K,1);

L = time(L_index);

T_index = find(kec>=(1-exp(-1))*K,1);

T = time(T_index);

ZN = [T/L 0 0;0.9*T/L L/.3 0;1.2*T/L 2*L 0.5*L]; %Kp Ti Td

% ZN = [T/L 0 0;0.9*T/L 0.3/L 0;1.2*T/L 0.5/L 0.5*L]; %Kp Ki Kd

P=ZN(3,1)

I=ZN(3,2)

D=ZN(3,3)

% P  =0; I=0; D=0;

simout = sim('PID_ZN', 'SimulationMode',
'normal','AbsTol','1e-5','SaveState',
'on','StateSaveName','xout','SaveOutput','on','OutputSaveName','
yout','SaveFormat','Array');

        output2 = simout.get('yout');

        time2 = simout.get('tout');

%parsing

kec2 = output2(:,1);

curr2 = output2(:,2);

waktuta= output2(:,3);

%plot(waktuta,kec2, 'LineWidth',2)

ylim([0 6])

xlim([0 100])

title(["PID-Ziegler Nichols"])

xlabel('time(s)')

ylabel('ω(rad/s)')

grid on;

hold on;

plot(waktuta, ref, "--")
```

```
hasil = stepinfo(kec,time)

hasil = stepinfo(kec,time)

hasil2 = stepinfo(kec2,time2)
```

Lampiran 3 Skrip Simulasi Kendali Motor DC PID- PSO

```
clear

clc

%Parameter Motor

rm = 4;

lm = 0.01;

j  = 0.0044;

b  = 0.0011;

kt = 0.22;

km = 0.22;

%Parameter PSO

n   = 30    ;

ns  = 50   ;

dim = 3   ;

w = 0.9;

c1  = 2.05     ;

c2  = 2.05     ;

fitness  = 0*ones(n,ns);

r1       = rand(n,dim);

r2       = rand(n,dim);

current_fitness = 0*ones(n,1);

for i = 1:n

   for j = 1:dim

       current_position(i,j) = 10*rand;
```

```
   end
end
velocity                = 0.3*randn(n,dim);
local_best_position     = current_position;
K                       = current_position;
for i = 1:n
   P=K(i,1);I=K(i,2); D=K(i,3);


simopt =
simset('solver','ode15s','SrcWorkspace','Current','DstWorkspace'
,'Current');  % Initialize sim options
        simout = sim('PID', 'SimulationMode',
'normal','AbsTol','1e-5','SaveState',
'on','StateSaveName','xout','SaveOutput','on','OutputSaveName','
yout','SaveFormat','Array');
        output = simout.get('yout');
        kec    = output(:,1);
        kec2   = kec(end);
        e      = kec2-50;
        Error  = sum(abs(e));
        F(i)=Error;
end
local_best_fitness = F;
[global_best_fitness,g]=min(local_best_fitness);
for i = 1:n
  global_best_position(i,:) = local_best_position(g,:) ;
end
dif1 = local_best_position - current_position ;
dif2 = global_best_position - current_position;
%update kecepatan
```

```
velocity = w*velocity + c1*(r1.*(local_best_position -
current_position)) + c2*(r2.*(global_best_position -
current_position));

%update posisi

current_position = current_position + velocity ;

iter = 0;

while (iter < ns)


   iter = iter + 1;

   for i = 1:n

        P=K(i,1);I=K(i,2); D=K(i,3);

        simopt =
simset('solver','ode15s','SrcWorkspace','Current','DstWorkspace'
,'Current');  % Initialize sim options

        simout = sim('PID', 'SimulationMode',
'normal','AbsTol','1e-5','SaveState',
'on','StateSaveName','xout','SaveOutput','on','OutputSaveName','
yout','SaveFormat','Array');

        output = simout.get('yout');


       kec    = output(:,1);

        kec2   = kec(end);

        e      = kec2-50;

        Error  = sum(abs(e));

        F(i)=Error;

   end



   if iter == 10

       figure(1)
```

```
        plot(tout,yout);
        title(['Iterasi=', num2str(iter),' Kp=',num2str(P),'
Ki=',num2str(I), 'Kd=', num2str(D)]), grid on;
    end
    if iter == 25
        figure(2)
        plot(tout,yout);
        title(['Iterasi=', num2str(iter),' Kp=',num2str(P),'
Ki=',num2str(I), 'Kd=', num2str(D)]), grid on;
    end
     if iter == 50
        figure(3)
        plot(tout,yout);
        title(['Iterasi=', num2str(iter),' Kp=',num2str(P),'
Ki=',num2str(I), 'Kd=', num2str(D)]), grid on;
    end
    for i = 1:n
        if F(i)< local_best_fitness(i)
            local_best_fitness(i) = F(i);
            local_best_position(i,:) = current_position(i,:);
        end
    end

    [current_global_best_fitness,g] = min(local_best_fitness);
    disp = min(local_best_fitness)

    if current_global_best_fitness < global_best_fitness
        global_best_fitness = current_global_best_fitness;
        for i = 1:n
```

```
            global_best_position(i,:)= local_best_position(g,:);
        end
    end

    if global_best_position(i) == global_best_position(i+1)
            break
    end
%update kecepatan
velocity = (w-(((w-0.4)/100)*iter)) *velocity +
c1*(r1.*(local_best_position - current_position)) + c2*
(r2.*(global_best_position - current_position));
%update posisi
current_position = current_position + velocity ;


end
K(i,:) = global_best_position(i,:);
P=K(i,1);I=K(i,2); D=K(i,3);
simopt =
simset('solver','ode15s','SrcWorkspace','Current','DstWorkspace'
,'Current');  % Initialize sim options
        simout = sim('PID', 'SimulationMode',
'normal','AbsTol','1e-5','SaveState',
'on','StateSaveName','xout','SaveOutput','on','OutputSaveName','
yout','SaveFormat','Array');
        output = simout.get('yout');
        tout   = simout.get('tout');
        v    = output(:,1);
figure(4)
plot(tout,v)
title(['Iterasi=', num2str(ns),' Kp=',num2str(P),'
Ki=',num2str(I), 'Kd=', num2str(D)]), grid on;
```

Lampiran 4 Skrip Simulasi Motor DC LQR-PID

```
clear

clc

close

%parameter PO

n   = 50    ;

ns  = 30;

dim = 2     ;

w   = 0.9  ;

c1  = 1.5  ;

c2  = 2 ;

fitness  = 0*ones(n,ns);

kolom1   = abs(rand(n,1));

kolom2   = abs(rand(n,1));

matrix   = [kolom1 kolom2];

% matrix   = abs(rand(n,dim));

r1       = rand(n,dim);

r2       = rand(n,dim);

v1       = rand(n,1);

v2       = rand(n,1)

current_fitness = 0*ones(n,1);

current_position = [matrix(:,1) matrix(:,2)]

velocity               = [v1 v2];

local_best_position    = current_position;

for i = 1:n

q1        = current_position(i,1);

r         = current_position(i,2);

q2 = 0;
```

```
   run_lqr;
% x= sum(output(:,2));
% u= sum(output(:,3));
%   J = x*Q*transpose(x) +u*R*transpose(u);
% % J = sum(abs(output(:,5)));
step_info=(stepinfo(output(:,1), output(:,4)));
J=
step_info.SettlingTime+step_info.RiseTime+step_info.Overshoot;
F(i)=J;
end
local_best_fitness = F;
[global_best_fitness,g]=min(local_best_fitness);
for i = 1:n
  global_best_position(i,:) = local_best_position(g,:) ;
end
%update kecepatan
ican= w*velocity;
yogi= c1*(r1.*(local_best_position - current_position));
beki=  c2*(r2.*(global_best_position - current_position));
velocity = ican + yogi +beki;
%update posisi
current_position = current_position + velocity ;
for i = 1:n %Q dan R tak boleh negatif ya ^o^
   if current_position(i,1) <= 0
       current_position(i,1) = abs(current_position(i,1))
   end

   if current_position(i,2) <= 0
```

```
       current_position(i,2) = abs(current_position(i,2))
   end
end
%plotting parcticle movement awal
figure(1)
plot3(current_position(:,1),current_position(:,2),current_posito
n(:,3), 'o') ;grid on
hold on;
plot3(current_position(g,1),current_position(g,2),current_positi
on(g,3), 'x') ;grid on
 xlim([0 0.5]);
 ylim([0 0.5]);
 zlim([0 0.5]);
title(g);
 hold off;
 drawnow

for iter = 1:ns
%     iter = iter + 1
   for i = 1:n
      q1 = current_position(i,1);
      r  = abs( current_position(i,2));
      q2 = 0;

       run_lqr;

%         x = sum(output(:,2));
%         u = sum(output(:,3));
%         J =x*Q*transpose(x) +u*R*transpose(u);
```

```
% J = sum(abs(output(:,5)));
step_info=(stepinfo(output(:,1), output(:,4)));
J=
step_info.SettlingTime+step_info.RiseTime+step_info.Overshoot;
              F(i)=J;
   end

   for i = 1:n
       if F(i)< local_best_fitness(i)
           local_best_fitness(i) = F(i);
           local_best_position(i,:) = current_position(i,:);
       end
   end

   [current_global_best_fitness,g] = min(local_best_fitness);
   disp = min(local_best_fitness);

   if current_global_best_fitness < global_best_fitness
       global_best_fitness = current_global_best_fitness;
       for i = 1:n
           global_best_position(i,:)= local_best_position(g,:);
       end
   end

%     if global_best_position(i) == global_best_position(i+1)
%         break
%     end
```

```
%update kecepatan

velocity = (w-(((w)/iter)*iter)) *velocity +
c1*(r1.*(local_best_position - current_position)) + c2*
(r2.*(global_best_position - current_position));

%update posisi

current_position = current_position + velocity ;

for i = 1:n %Q dan R tak boleh negatif ya ^o^

   if current_position(i,1) <= 0

       current_position(i,1) = abs(current_position(i,1))

   end


   if current_position(i,2) <= 0

       current_position(i,2) = abs(current_position(i,2))

   end


%     if current_position(i,3) <= 0

%         current_position(i,3) = abs(current_position(i,3))

%     end

end

%plotting parcticle movement

hasil(i,:) = global_best_position(i,:)

q1        = hasil(i,1);

r        = abs(hasil(i,2));

q2 = 0;

run_lqr

figure(1)

kec= output(:,1);

time= output(:,4);

plot(time,kec)
```

```
title(['g=', num2str(g), " iterasi=", num2str(iter)] )
xlabel('time(s)')
ylabel('kecepatan')
grid on;
drawnow;
gbest(iter) = global_best_fitness
end
hasil(i,:) = global_best_position(i,:)
q1         = hasil(i,1);
r         = hasil(i,2);
q2 = 0;
run_lqr
kec = output(:,1);
%plotting
figure(2)
plot(time,kec)
title(["Respon Motor Pengendali LQR"])
xlabel('time(s)')
ylabel('kecepatan')
grid on;
step_info = stepinfo(kec,time)
figure(3)
plot(gbest)
grid on;
```

Lampiran 5 Tabel Perhitungan Step Algoritma PSO

No	Statement	S/E	Frequency	Steps
1	Inisiasi 2 dimensional swarm;	1	2n	2n
2	Inisiasi parameter awal	1	7	7
3	For tiap particle *i* = 1,...,*n* do	1	n+1	n+1
	//tentukan local best position	0	0	0
4	**if** *f*(**x***i*) < *f*(**y***i*) **then**	1	n	n
5	**y***i* = **x***i*;	1	n	n
	end	0	0	0
	//tentukan global best position	0	0	0
6	**if** *f*(**y***i*) < *f*(**y**^) then	1	n	n
7	y^ = **y***i*;	1	n	n
	end	0	0	0
	end	0	0	0
	for tiap particle i = 1,...,n **do**	1	n+1	n+1
	update kecepatan ;	0	0	0
8	$v_{id}(t+1) = wv_{id}(t) + c_1 r_1\left(P_{id} - x_{id}(t)\right) + c_2 r_2\left(P_{gd} - x_{id}(t)\right)$	1	9n	9n
	update update posisi;			
9	$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1)$	1	n	n
	Total			18n+9