Tugas Besar Kendali Lanjut (KL)

Analisa Performansi

Segway with LQR, Kalman Filter, and LQG

KL – 41 – D (ZAR)

Gede Haris Widiarta (1102174038)

DAFTAR ISI

DAFTAR GAMBAR

A. Peninjauan Sistem Fisik

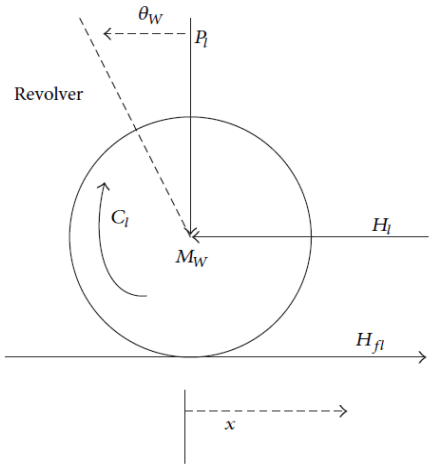
Sistem fisik yang ditinjau adalah *Segway*. *Segway* merupakan kendaraan listrik *(scooter)* beroda dua yang mempunyai fitur *self-balancing* dan hanya berkapasitas satu orang. *Segway* merupakan salah satu bentuk implementasi dari *prototype self-balancing robot.* Perbedaannya terletak pada desain dan komponen yang digunakan karena *Segway* memiliki ukuran yang lebih besar, lebih berat, dan memiliki pengendara.

Gambar 1. Segway

Pemodelan fisik sistem terbagi dua, yaitu pemodelan roda dan pemodelan rangka *Segway*. Berikut adalah uraiannya.

a. Pemodelan Roda[1][2]

   Pada pemodelan roda *Segway*, asumsinya yaitu model matematis roda kiri sama dengan roda kanan.



Gambar 2. Roda Segway

Untuk itu, dengan menggunakan Hukum Newton II, diperoleh persamaan gaya dan persamaan torsi pada roda. Untuk roda kanan yaitu:

$$\Sigma F_x = M_w a$$

$$M_w \ddot{x} = H_{fR} - H_R$$

$$I_w \ddot{\theta}_w = C_R - H_{fR} r$$

Sedangkan, untuk persamaan roda kiri,

$$\Sigma F_x = M_w a$$

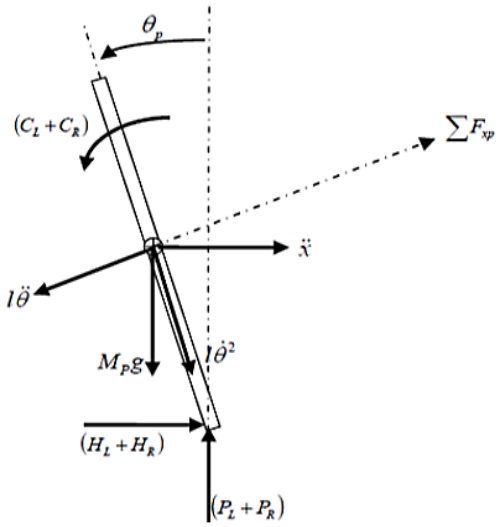
$$M_w \ddot{x} = H_{fL} - H_R$$

$$I_w \ddot{\theta}_w = C_L - H_{fL} r$$

Sehingga jika kedua persamaan roda kanan dan kiri digabung maka,

$$2\left(M_w + \frac{I_w}{r^2}\right)\ddot{x} = \frac{C_R + C_L}{R} - (H_R + H_L)$$

b. Pemodelan Rangka *Segway*[1][2]



Gambar 3. Rangka Segway

Untuk menemukan persamaan matematis dari rangka *Segway*, maka Hukum Newton II digunakan kembali. Resultan gaya pada sumbu horizontal yaitu:

$$\Sigma F_x = M_p \ddot{x}_p$$

$$M_p \ddot{x}_p = H_L + H_R - M_p l\ddot{\theta}_p sin\theta_p - Ml\ddot{\theta}_p cos\theta_p$$

$$M_p \ddot{x}_p = M_p(x + lsin\ddot{\theta}_p)$$

Keterangan, x_p merupakan titik pusat massa rangka *Segway*. Dengan menggunakan Hukum Newton II, maka diperoleh juga persamaan resultan gaya pada sumbu vertical.

$$\Sigma F_{xp} = M_p g sin\theta - (P_R + P_L)\, sin\theta_p + (H_R + H_L)\, cos\theta_p$$
$$= M_p(x + lsin\ddot{\theta}_p)\, cos\theta_p$$

Dan untuk jumlah torsi pada pusat massa *Segway* yaitu:

$$\Sigma M_o = I_p\ddot{\theta}_p = -(H_R + H_L)\, lcos\theta_p - (P_R + P_L)l\, sin\theta_p + (C_R + C_L)$$

B. Pemodelan *State Space*

Berdasarkan pemodelan matematis diatas, maka dialnjutkan ke proses linearisasi dengan asumsi $sin\theta_p = \theta_p\ ;\ cos\theta_p = 1\ ;\ (d\theta_p/dt)^2 = 0$, maka

$$\left(M_p + 2M_w - \frac{I_w}{r^2}\right)\ddot{x} = \frac{C_R + C_L}{R} - M_p l\ddot{\theta}_p$$

$$\left(M_p l^2 + I_p\right)\ddot{\theta}_p = M_p g l\theta_p + M_p l\ddot{x} - (C_R + C_L)$$

Sehingga, *state space* nya menjadi,

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta}_p \\ \ddot{\theta}_p \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{K_1(M_p lr - I_p - M_p l^2)}{Rr^2A} & \frac{M_p^2 g l^2}{A} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{K_1(rB - M_p l)}{Rr^2A} & \frac{M_p g l B}{A} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta_p \\ \dot{\theta}_p \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{2k_m(I_p + M_p l^2 - M_p lr)}{RrA} \\ 0 \\ \frac{2k_m(M_p l - rB)}{RrA} \end{bmatrix}$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta_p \\ \dot{\theta}_p \end{bmatrix}$$

Keterangan:

$A = [I_p B + 2M_p l^2(M_w + (I_w/r^2))]\ dan\ B = (2M_w + (2I_w/r^2) + M_p)$

$K_1 = 2k_m k_e$

$C_R + C_L = (2\frac{k_m}{R})U_a - (2\frac{k_m k_e}{Rr})\dot{x}$

Parameter Sistem:

Simbol	Keterangan	Satuan	Nilai
k_m	torsi konstan motor DC	Nm/A	0.1
k_e	*back* EMF motor DC	Vs/rad	0.12
R	resistor rangkaian motor DC	Ω	12
M_p	*massa Segway*	kg	5.4
M_w	*massa* roda *Segway*	kg	0.8
l	panjang rangka dari *shaft*	m	1.2
I_p	momen inersia *Segway*	kgm^2	0.048
I_w	momen inersia roda *Segway*	kgm^2	0.042
g	gravitasi	m/s^2	9.8
r	radius roda	m	0.15
$C_L = C_R$	torsi roda	Nm	$(k_m/R)U_a - (\frac{k_m k_e}{Rr})\dot{x}$

Kemudian, untuk deklarasi pada *matlab*, dapat dilihat pada *source code* berikut.

```
% Define Parameter System
km = 0.1;       %torque constant motor (Nm/A)
ke = 0.12;      %torque constant/back EMF motor (Vs/rad)
Ri = 12;        %resistor value in DC motor (ohm)
mp = 5.4;       %mass of segway robot (kg)
mw = 0.8;       %mass of segway robot wheels (kg)
l  = 1.2;       %lenght/height body form the shaft (m)
Ip = 0.048;     %moment inertia of segway robot (kg*m^2)
Iw = 0.042;     %moment inertia of segway robot wheels (kg*m^2)
g  = 9.8;       %gravity value (m/s^2)
r  = 0.15;      %radius of segway wheels (m)

B1  = (2*mw + (2*Iw/r^2) + mp);
A1  = (Ip*B1 + 2*mp*l^2*(mw + (Iw/r^2)));
%%
% Define State Space Matrix
A = [0                      1                              0              0;
     0   (2*km*ke)*(mp*l*r-Ip-mp*l^2)/Ri*r^2*A1   (mp^2*g*l^2)/A1       0;
     0                      0                              0              1;
     0      (2*km*ke)*(r*B1-mp*l)/Ri*r^2*A1       (mp*g*l*B1)/A1        0];

B = [0 ; 2*km*(Ip*+mp*l^2-mp*l*r)/Ri*r*A1 ; 0 ;(2*km)*(mp*l-r*B1)/Ri*r*A1];
C = [1 0 0 0;0 0 1 0];
D = [0];
```

Gambar 4. Deklarasi Parameter di Matlab

C. Performansi Sistem *Open Loop*

Kestabilan sistem *open loop* dapat diketahui dari nilai eigen yang didapat dari matriks A dengan menggunakan *source code* berikut.

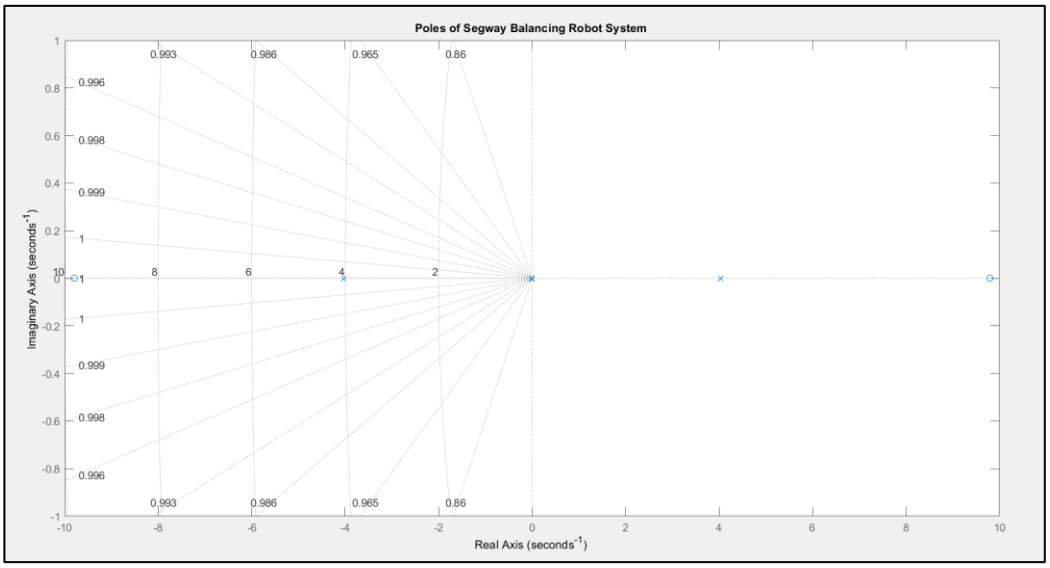
```
% Open-Loop Response
sysOL1 = ss(A,B,C(1,:),D);
sysOL2 = ss(A,B,C(2,:),D);
```

```
%PZmap
disp('Poles Sistem: ')
poles = eig(A)
subplot(2, 2, 1:4)
pzmap(sysOL1)
hold on
pzmap(sysOL2)
grid on
title('Poles of Segway Balancing Robot System')
%Bode Plot
figure(3)
bode(sysOL1)
hold on
bode(sysOL2)
grid on
set(legend('sysOL x','sysOL \theta'))
```

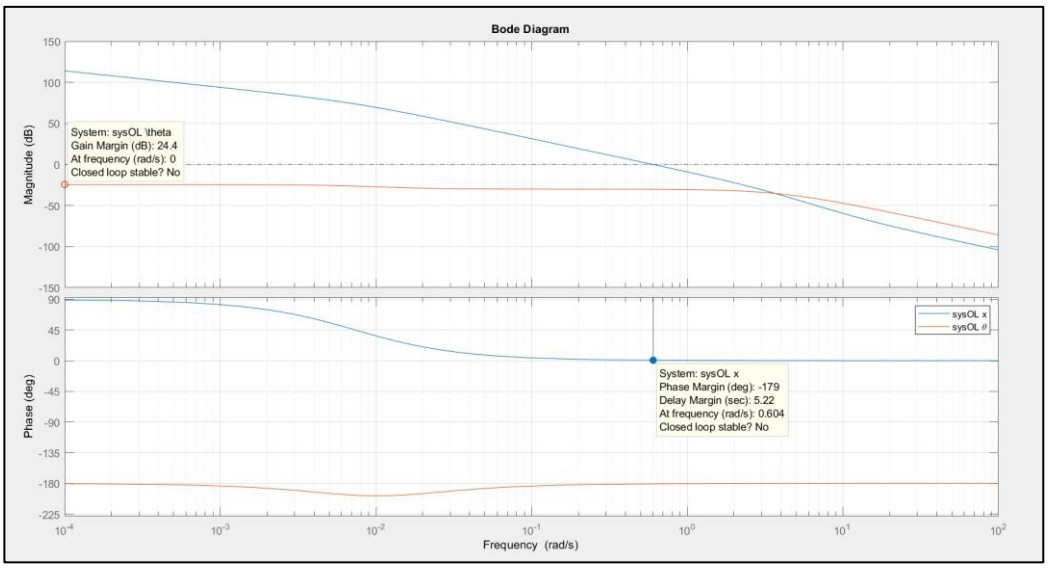
Hasil simulasi:

```
>> poles

poles =

         0
   -0.0074
   -4.0319
    4.0263
```

Gambar 5. Poles Sistem



Gambar 6. Pole Zero Map (pzmap)



Gambar 7. Bode Diagram Sistem

- Analisa hasil:

  Berdasarkan hasil simulasi, dapat dilihat dari semua nilai eigen atau *poles* sistem tidak berada pada *left half plane*, tetapi ada satu *pole* sistem yaitu 4.023 yang berada di *right half plane* sehingga dari hasil *poles* sistem, dapat ditarik kesimpulan bahwa sistem *open loop* tidak stabil. Hasil ini didukung juga oleh grafik *poles zero map (pzmap)* dan *bode diagram* sistem yang menunjukkan juga bahwa sistem *open loop* tidak stabil.

D. *Controllable* dan *Observable*

Dalam suatu sistem, penting untuk mengetahui apakah sistem tersebut *controllable* agar dapat dikendalikan serta penting juga untuk mengetahui apakah sistem tersebut *observable* agar kita dapat merancang *estimator* sistem. Berikut merupakan *source code* pada *matlab.*

```
% Controllability
Matrix_Co = ctrb(A,B)
Rank_Co   = rank(ctrb(A,B))
if Rank_Co == size(Matrix_Co);
    fprintf('System is Controllable\n')
else
    fprintf('System is Uncontrollable\n')
end
```

```
% Observeability
Matrix_Ob = obsv(A,C(1,:))
Rank_Ob   = rank(obsv(A,C(1,:)))
if Rank_Ob == size(Matrix_Ob);
    fprintf('System is Observeable\n')
else
    fprintf('System is Unobserveable\n')
end
```

Hasil simulasi:

```
Matrix_Co =

         0   -0.0628    0.0008    5.0101
   -0.0628    0.0008    5.0101   -0.0592
         0    0.5112    0.0006    8.2986
    0.5112    0.0006    8.2986   -0.0367


Rank_Co =

     4

System is Controllable
```

Gambar 8. Controllability Sistem

```
Matrix_Ob =

    1.0000         0         0         0
         0    1.0000         0         0
         0   -0.0129    9.8007         0
         0    0.0002   -0.1269    9.8007


Rank_Ob =

     4

System is Observeable
```

Gambar 9. Observeability Sistem

- Analisa hasil:

  Berdasarkan hasil simulasi diatas, sistem bersifat *controllable* sehingga dapat dikendalikan dan *observable* sehingga sistem dapat di estimasi. Kedua hal ini, sangat penting dilakukan sebelum melakukan perancangan sistem kendali dan estimasi, agar kita mengetahui sistem tersebut bisa atau tidak dikendalikan dan diestimasi.

E. *Linear Quadratic Controller (LQR)*

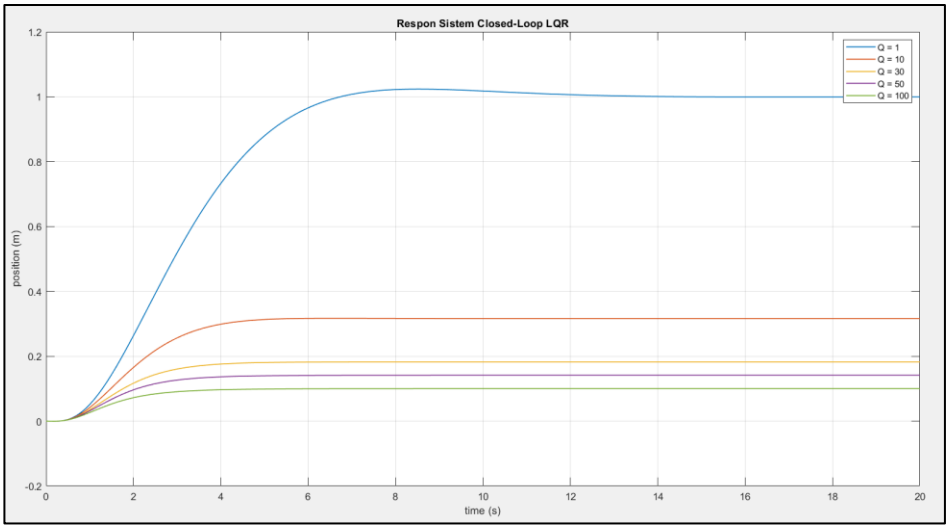
Untuk membuat sistem *Segway* stabil, salah satu metode kendali yang dapat digunakan yaitu *Linear Quadratic Regulator (LQR)*. Dengan menggunakan LQR penting untuk mengetahui nilai Q dan R yang digunakan karena sangat berpengaruh pada performansi sistem. Pada simulasi ini, terdapat dua skema LQR yaitu:

a. Variasi Q dan R konstan

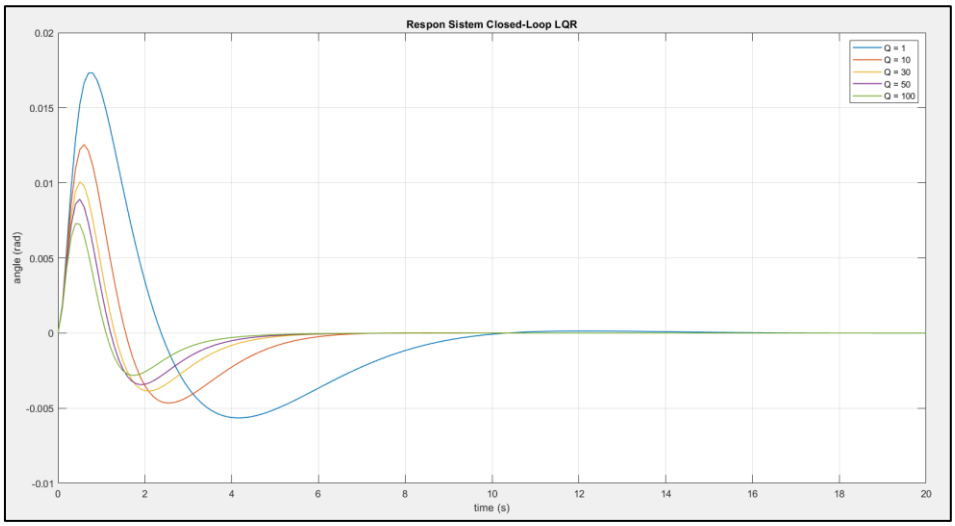
```
% Linear Quadratic Regulator (LQR)
% Define LQR Parameter
% Variasi nilai Q dan R konstan (Model 1)
for i = 1:5;
    switch i
        case 1
            Q = diag([1 1 1 1]); R = 1;
        case 2
            Q = diag([10 10 10 10]); R = 1;
        case 3
            Q = diag([30 30 30 30]); R = 1;
        case 4
            Q = diag([50 50 50 50]); R = 1;
        case 5
            Q = diag([100 100 100 100]); R = 1;
    end

K1 = lqr(A,B,Q,R);K1
sysCL1 = ss(A-B*K1,B,C(1,:),D);
sysCL2 = ss(A-B*K1,B,C(2,:),D);

%Plotting LQR Model 1
t = 0:0.1:20;
y1 = step(sysCL1,t);
y2 = step(sysCL2,t);
```

```
figure(4)
plot(t,y1,'-','LineWidth',1);
grid on
title('Respon Sistem Closed-Loop LQR')
xlabel('time (s)')
ylabel('position (m)')
set(legend('Q = 1','Q = 10','Q = 30','Q = 50','Q = 100'))
hold on

figure(5)
plot(t,y2,'-','LineWidth',1);
grid on
title('Respon Sistem Closed-Loop LQR')
xlabel('time (s)')
ylabel('angle (rad)')
set(legend('Q = 1','Q = 10','Q = 30','Q = 50','Q = 100'))
hold on
end
```

Hasil simulasi:



Gambar 10. Variasi Q (Posisi)



Gambar 11. Variasi Q (Sudut)

- Analisa hasil:

  Variasi Q dan R = 1

  Q = diag([1 1 1 1]) → K = [1.0000 2.9247 78.6409 17.9183]

  Q = diag([10 10 10 10]) → K = [3.1623 6.5330 94.7119 20.2791]

  Q = diag([30 30 30 30]) → K = [5.4772 10.0100 109.4358 22.5881]

  Q = diag([50 50 50 50]) → K = [7.0711 12.3352 119.1601 24.1824]

  Q = diag([100 100 100 100]) → K = [10.0000 16.5505 136.7678 27.174]

  Berdasarkan grafik respon sistem dengan variasi Q dan R konstan diatas, dapat ditarik kesimpulan, semakin besar nilai Q yang diberikan maka nilai

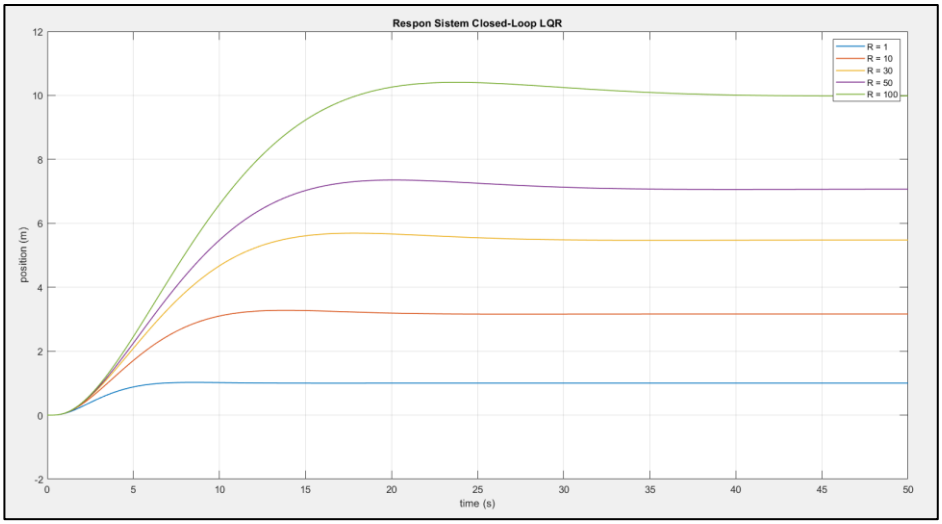
gain K semakin besar serta perubahan posisi dan sudut akan semakin kecil, sehingga dengan nilai Q yang lebih besar, sitem akan lebih cepat mencapai titik kestabilan.

b. Variasi R dan Q konstan

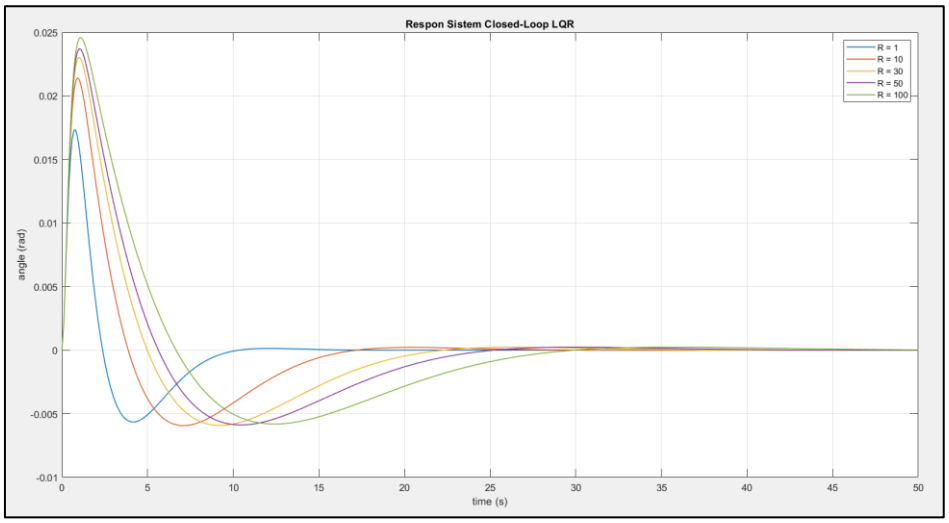
```
% Variasi nilai R dan Q konstan (Model 2)
for i = 1:5;
    switch i
        case 1
            Q = eye(4); R = 1;
        case 2
            Q = eye(4); R = 10;
        case 3
            Q = eye(4); R = 30;
        case 4
            Q = eye(4); R = 50;
        case 5
            Q = eye(4); R = 100;
    end

K2 = lqr(A,B,Q,R);K2
sysCL1 = ss(A-B*K2,B,C(1,:),D);
sysCL2 = ss(A-B*K2,B,C(2,:),D);
```

```
%Plotting LQR Model 1
t = 0:0.1:50;
y1 = step(sysCL1,t);
y2 = step(sysCL2,t);

figure(6)
plot(t,y1,'-','LineWidth',1);
grid on
title('Respon Sistem Closed-Loop LQR')
xlabel('time (s)')
ylabel('position (m)')
set(legend('R = 1','R = 10','R = 30','R = 50','R = 100'))
hold on

figure(7)
plot(t,y2,'-','LineWidth',1);
grid on
title('Respon Sistem Closed-Loop LQR')
xlabel('time (s)')
ylabel('angle (rad)')
set(legend('R = 1','R = 10','R = 30','R = 50','R = 100'))
hold on
end
```

Hasil simulasi:



Gambar 12. Variasi R (Posisi)



Gambar 13. Variasi R (Sudut)

- Analisa hasil:

  Variasi R dan Q = diag([1 1 1 1])

  R = 1 → K = [1.0000 2.9247 78.6409 17.9183]

  R = 10 → K = [0.3162 1.4303 71.3677 16.8867]

  R = 30 → K = [0.1826 1.0343 69.3233 16.5967]

  R = 50 → K = [0.1414 0.8913 68.5685 16.4894]

  R = 100 → K = [0.1000 0.7288 67.6998 16.3656]

  Berdasarkan hasil simulasi diatas, dapat ditarik kesimpulan, jika nilai R semakin besar maka nilai *gain* K akan semakin kecil. Tetapi jika nilai R

semakin besar perubahan posisi dan sudut juga akan membesar sehingga sistem akan membutuhkan waktu yang lama untuk mencapai titik kestabilan.

F. *Kalman Filter State Estimator*

Dalam sistem, pengukuran seluruh *state* dapat menimbulkan *noise*. Oleh karena itu, diperlukan suatu estimator untuk seluruh *state* agar dapat mengestimasi nilai *state* yang tidak dapat diukur dan memiliki *noise*. Untuk itu, parameter *noise* dan *disturbance* ditambahkan sehingga bentuk *state space* menjadi:

$$\dot{x} = Ax + Bu + W_d$$
$$y = Cx + Du + W_n$$

Dimana W_d merupakan koefisien *disturbance* dan W_n merupakan koefisien *noise*. Sistem estimator dapat direpresentasikan dalam bentuk *state space* berikut.

$$\dot{\hat{x}} = A\hat{x} + Bu + K_f(y - \hat{y})$$
$$\hat{y} = C\hat{x} + Du$$

Dengan nilai K_f yang didapatkan dari metode LQR yaitu:

$$K_f = YC'V_n$$

Dimana Y merupakan solusi dari persamaan ARE[3],

$$YA' + AY - YC'^{W_n^{-1}}CY + W_d = 0$$

```
% Kalman Filter Value
Kf = (lqr(A',C(1,:)',Wd,Wn))';
```

Sehingga, *state space estimator* yaitu:

$$\dot{\hat{x}} = (A - K_f C)\hat{x} + [K_f \quad (B - K_f)]\,[y \quad u]'$$

```
sysKF    = ss(A-Kf*C(1,:),[B Kf],eye(4),0*[B Kf]);
```

Berikut merupakan *source code* pada *matlab.*

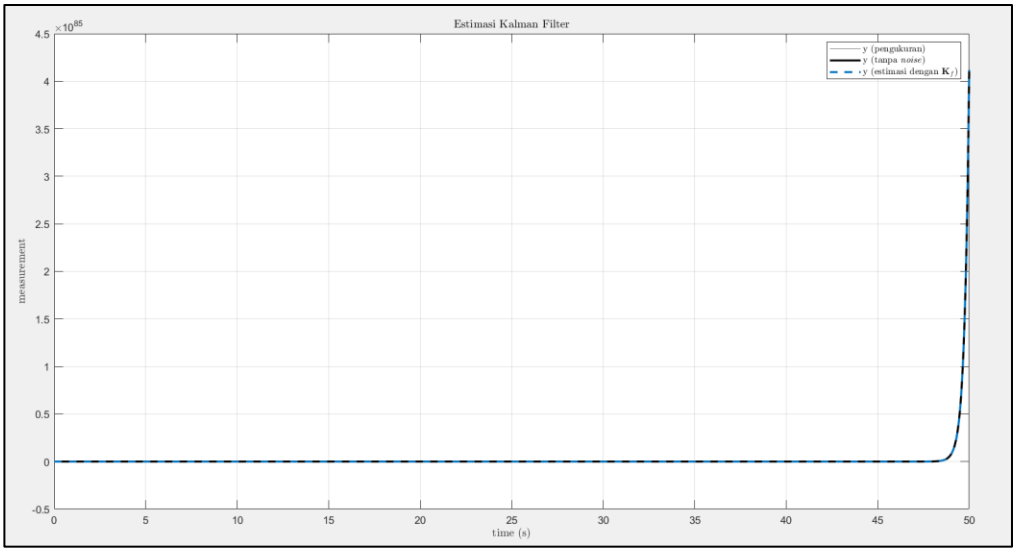
```
% Kalman Filter
% Define Value of Disturbances (Md) & Noise (Mn)
Wd = eye(4); %Matrix Disturbances
Wn = 1;      %Matrix Noise

% Kalman Filter Value
Kf = (lqr(A',C(1,:)',Wd,Wn))';
```

```
% Augmented System
% xdot = Ax + Bu + Md*d + 0*n
% y    = Cx + Du + 0*d + Mn*n
Baug = [B eye(4) 0*B];                                  %[Bu Md 0*n]
Daug = [0 0 0 0 0 1 ];                                  %[Du 0*d Mn]
sysC = ss(A,Baug,C(1,:),Daug);                          %Sistem Pengukuran
sysTruth = ss(A,Baug,eye(4),zeros(4,size(Baug,2)));     %Sistem Full Out.(
sysKF    = ss(A-Kf*C(1,:),[B Kf],eye(4),0*[B Kf]);      %Sistem Kalman Filt

% Estimasi Sistem Linear
dt = .01;
t  = dt:dt:50;

u_dist  = sqrt(Vd)*randn(4,size(t,2)); %Random Distrurbances Matrix
u_noise = sqrt(Vn)*randn(size(t));     %Random Noise Matrix
u = 0*t;
u(1/dt) = 20/dt;                       %Positive Impulse
u(15/dt) = -20/dt;                     %Negative Impulse
u_aug = [u; u_dist; u_noise];          %Input dengan Disturbance dan Noise

[y,t] = lsim(sysC,u_aug,t);            %Dengan Noise
[xtrue,t] = lsim(sysTruth,u_aug,t);    %State Sebenarnya
[xhat,t] = lsim(sysKF,[u; y'],t);      %State Estimasi
```

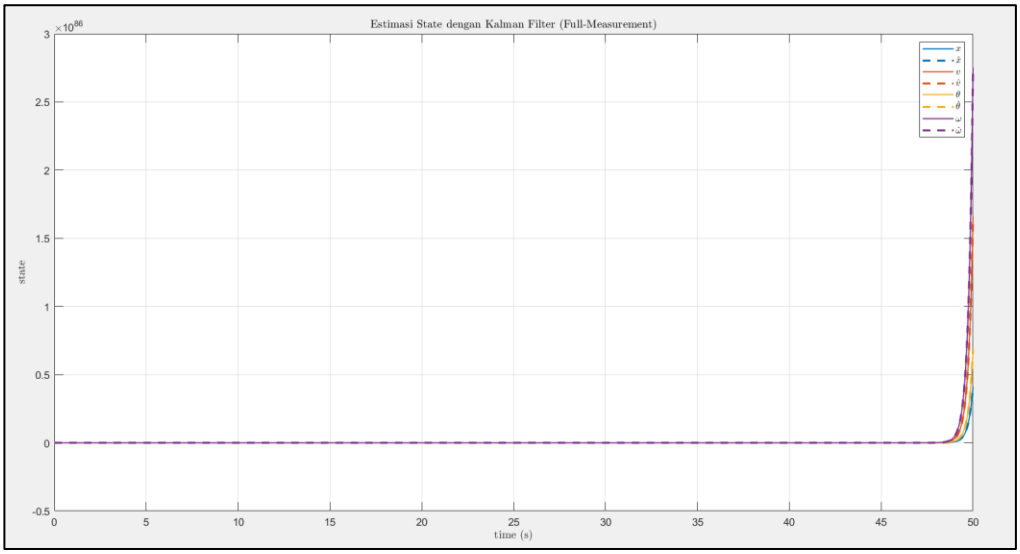
```
%Plotting
RGB_Index = [   0      0.4470    0.7410
             0.8500    0.3250    0.0980
             0.9290    0.6940    0.1250
             0.4940    0.1840    0.5560
             0.4660    0.6740    0.1880
             0.3010    0.7450    0.9330
             0.6350    0.0780    0.1840];

figure(1)
plot(t,y,'Color',[.5 .5 .5])
hold on
plot(t,xtrue(:,1),'Color',[0 0 0],'LineWidth',2)
plot(t,xhat(:,1),'--','Color',RGB_Index(1,:),'LineWidth',2)
grid on
title('Estimasi Kalman Filter')
xlabel('time (s)')
ylabel('measurement')
set(legend('y (pengukuran)','y (tanpa \textit{noise})','y (estimasi dengan
set(0,'defaulttextinterpreter','Latex')
```

```
figure(2)
for k=1:4
plot(t,xtrue(:,k),'-','LineWidth',1.2,'Color',RGB_Index(k,:));
hold on
plot(t,xhat(:,k),'--','LineWidth',2,'Color',RGB_Index(k,:))
end
title('Estimasi State dengan Kalman Filter (Full-Measurement)')
xlabel('time (s)')
ylabel('state')
set(legend('$x$','$\hat{x}$','$v$','$\hat{v}$','$\theta$','$\hat{\theta}$',
set(0,'defaulttextinterpreter','Latex')
grid on
```

Hasil simulasi:



Gambar 14. Estimasi Kalman Filter



Gambar 15. Estimasi State dengan Kalman Filter

- Analisa hasil:

  Berdasarkan hasil estimasi dengan Kalman Filter diatas, jika diamati grafik hasil simulasi menunjukkan bahwa hasil estimasi tidak stabil dan menuju ke sumbu positif tak hingga sehingga bisa ditarik kesimpulan yaitu hasil estimasi tidak stabil. Hal ini dikarenakan performansi sistem *open loop* yang mengandalkan *poles* atau nilai *eigen* dari matriks A tidak memenuhi syarat kestabilan sistem atau terdapat *poles* yang berada pada *right half plane.*

G. *Linear Quadratic Gaussian* (LQG)

Linear Quadratic Gaussian (LQG) merupakan suatu sistem kendali yang merupakan gabungan dari sistem kendali LQR dan Kalman Filter. Berikut merupakan *state space closed-loop* nya[4].

$$\begin{bmatrix} x \\ \varepsilon \end{bmatrix} = \begin{bmatrix} A - BK & BK \\ 0 & A - K_f C \end{bmatrix} \begin{bmatrix} x \\ \varepsilon \end{bmatrix}$$

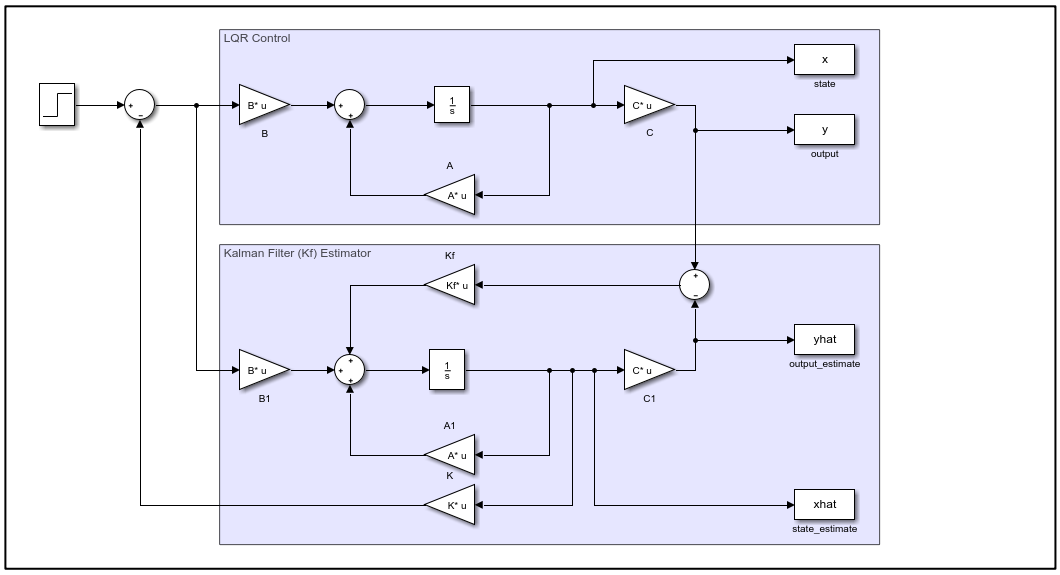
Dimana $\varepsilon = x - \hat{x}$; $A - BK$ merupakan LQR; dan $A - K_f C$ merupakan Kalman Filter. Berikut adalah *source code* pada *matlab.*

```
% LQR Control
Q = 50*eye(4);
R = 1;
K = lqr(A,B,Q,R);
%%
% Kalman Filter (Kf)
Wd = 4*eye(4);
Wn = .5;
Kf = (lqr(A',C',Wd,Wn))';
%%
% Simulink Model LGQ
t_final = 8;
sim('LQG_Sim')

t = y.time;
ytrue = y.signals.values;
yhat  = yhat.signals.values(:,1);
xtrue = x.signals.values;
xhat  = xhat.signals.values;
```

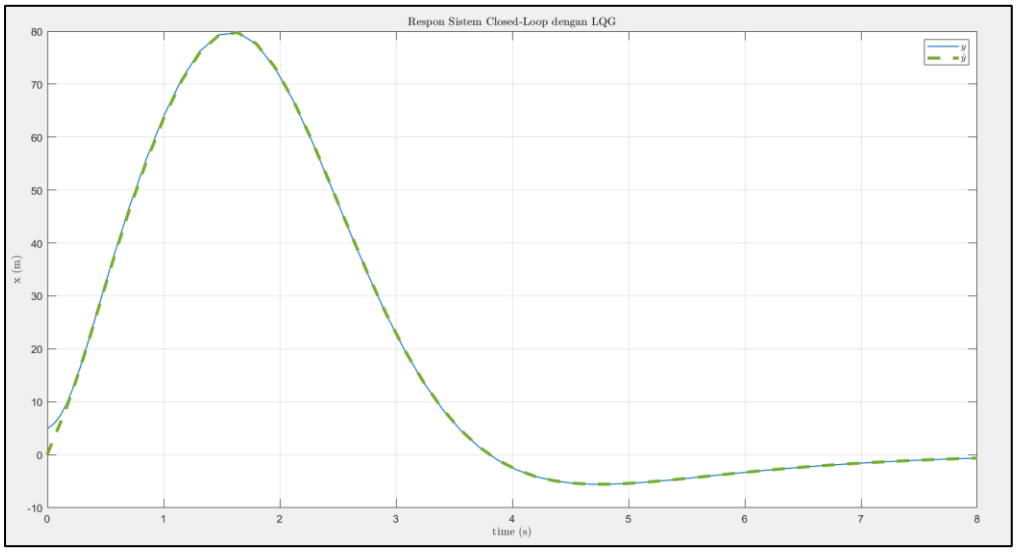
```
% Plotting Response
RGB_Index = [   0      0.4470    0.7410
             0.8500    0.3250    0.0980
             0.9290    0.6940    0.1250
             0.4940    0.1840    0.5560
             0.4660    0.6740    0.1880
             0.3010    0.7450    0.9330
             0.6350    0.0780    0.1840];

figure(1)
plot(t,ytrue,'-','LineWidth',1), hold on
plot(t,yhat,'--','LineWidth',3,'Color',RGB_Index(5,:)), grid on
title('Respon Sistem Closed-Loop dengan LQG')
xlabel('time (s)')
ylabel('x (m)')
set(legend('$y$','$\hat{y}$'),'interpreter','latex')
hold on
```

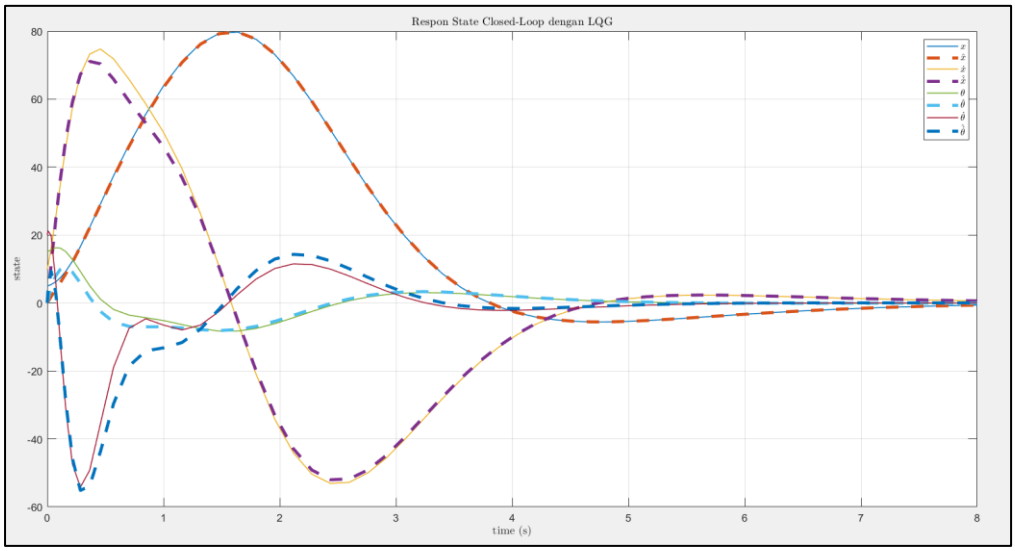
```
figure(2)
plot(t,xtrue(:,1),'-','LineWidth',1), hold on
plot(t,xhat(:,1),'--','LineWidth',3), hold on
plot(t,xtrue(:,2),'-','LineWidth',1), hold on
plot(t,xhat(:,2),'--','LineWidth',3), hold on
plot(t,xtrue(:,3),'-','LineWidth',1), hold on
plot(t,xhat(:,3),'--','LineWidth',3), hold on
plot(t,xtrue(:,4),'-','LineWidth',1), hold on
plot(t,xhat(:,4),'--','LineWidth',3), hold on
grid on
title('Respon State Closed-Loop dengan LQG')
xlabel('time (s)')
ylabel('state')
set(legend('$x$','$\hat{x}$','$\dot{x}$','$\hat{\dot{x}}$','$\theta$','$\ha
```

Simulink:



Hasil simulasi:





Gambar 16. Respon LQG

- Analisa hasil:

  Berdasarkan hasil simulasi diatas, LQG mampu mengendalikan suatu sistem yang performansi *open loop* nya bersifat stabil maupun yang tidak stabil. LQG juga mampu mengestimasi sistem yang *open loop* nya tidak stabil dengan mengejar nilai – nilai antara sistem estimator dengan sistem sebenarnya.

H. Kesimpulan

Kesimpulan yang dapat diambil yaitu performansi sistem *Segway* pada:

- *Open Loop System* → tidak stabil, nilai *pole* sistem ada yang berada pada *right half plane.*
- *Controllability* → sistem *controllable*, dapat dikendalikan.
- *Observability* → sistem *observable*, dapat diestimasi.
- *LQR (Q Variasi)* → sistem stabil, semakin tinggi nilai Q semakin cepat sistem mencapai kestabilan.
- *LQR (R Variasi)* → sistem stabil, semakin tinggi nilai R semakin lama sistem untuk mencapai kestabilan.
- *Kalman Filter* → hasil estimasi tidak stabil, karena performansi sistem *open loop* tidak stabil.
- *LQG* → sistem stabil dan mampu mengestimasi dengan cara mengejar ketertinggalan nilai sistem estimasi terhadap sistem sebenarnya.

I. Tautan Video Penjelasan

Link video:

https://youtu.be/53eGHaNZ1rY atau https://tinyurl.com/tubesKLharis

J. Referensi

[1] I. K. Mohammed and A. I. Abdulla, "Balancing a Segway robot using LQR controller based on genetic and bacteria foraging optimization algorithms," *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 18, no. 5, pp. 2642–2653, 2020, doi: 10.12928/TELKOMNIKA.v18i5.14717.

[2] J. Fang, "The LQR controller design of two-wheeled self-balancing robot based on the particle swarm optimization algorithm," *Math. Probl. Eng.*, vol. 2014, 2014, doi: 10.1155/2014/729095.

[3] S. Brunton, *Control Bootcamp- Kalman Filter Example in Matlab*. United States, 2017.

[4] S. Brunton, *Control Bootcamp: Linear Quadratic Gaussian*. United States, 2017.