به نام خدا

شبیه سازی این مقاله در دو گام شبیه سازی دینامیک ربات و سپس طراحی کنترل فازی تطبیقی و اعمال آن بر ربات و شبیه سازی آن نتایج آن انجام می گیرد.

گام اول: شبیه سازی دینامیک ربات و تست:

$$\ddot{x} = \frac{(\sum_{i=1}^{4} T_i)(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi - D_1)}{m}$$

$$\ddot{y} = \frac{(\sum_{i=1}^{4} T_i)(\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi - D_2)}{m}$$

$$\ddot{z} = \frac{(\sum_{i=1}^{4} T_i)(\cos\phi\cos\psi - mg - D_3)}{m}$$

$$\ddot{\theta} = l_1(-T_1 - T_2 + T_3 + T_4 - D_4)/J_1$$

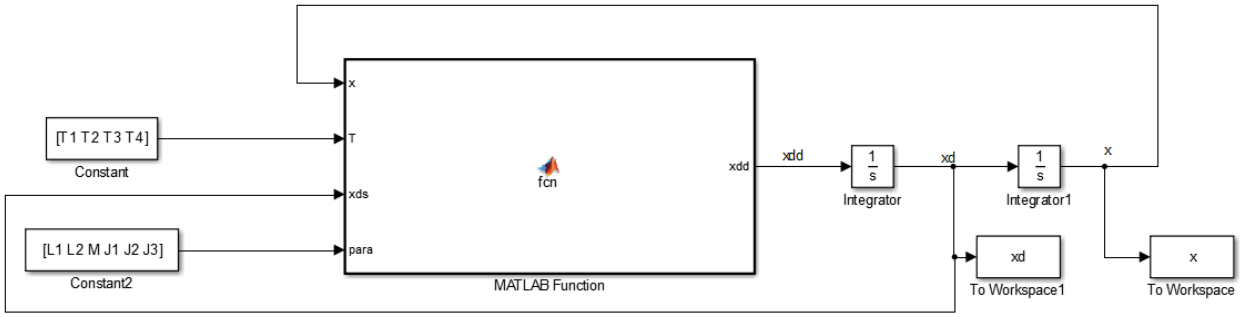
$$\ddot{\psi} = l_2(-T_1 + T_2 + T_3 - T_4 - D_5)/J_2$$

$$\ddot{\phi} = (M_1 - M_2 + M_3 - M_4 - D_6)/J_3$$

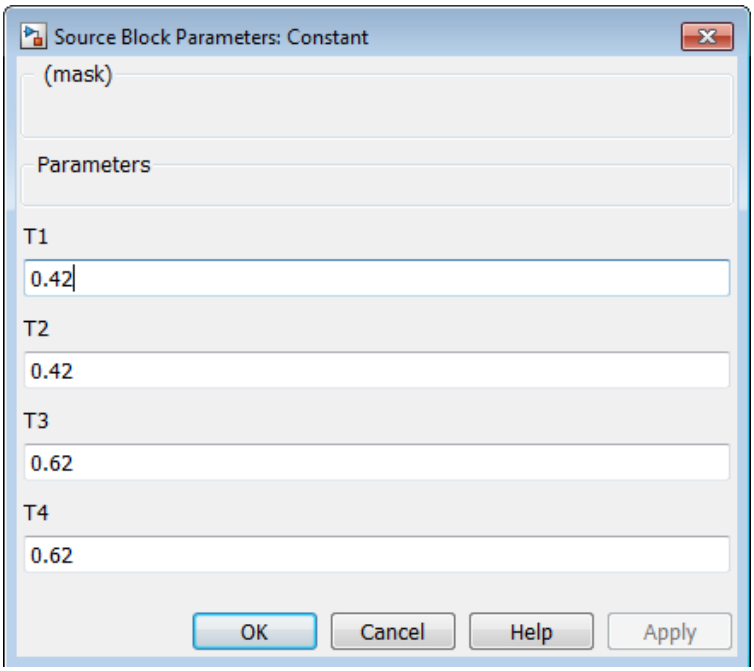
در معادلات فوق، ارتباط پارامترهای مربوط به درگ $(D_1, D_2, \dots, D_6)$ با سرعت های انتقالی و دورانی ربات داده نشده است و بنابراین در ضمن شبیه سازی با توجه به ارتباط توان دو درگ با این پارامترها، ضرایب آن به دلخواه انتخاب شده است.

هم چنین، پارامترهای $(M_1, M_2, \dots, M_6)$ ارتباط آن با میزان تراست تولیدی داده نشده است و بنابراین به صورت فرضی با ضریب ۵ منظور شده است.

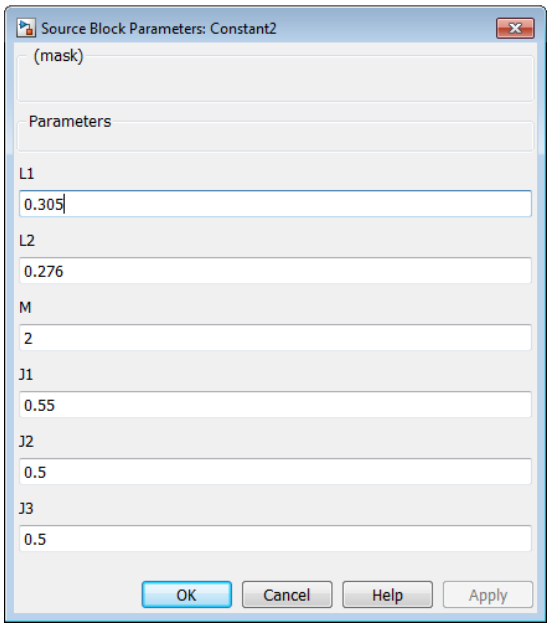
شبیه سازی دینامیک سیستم در نرم افزار سیمولینک مطابق با شکل شماتیک زیر انجام شد:



پارامترهای مربوط به تراست ورودی به سیستم، با دابل کلیک کردن بر روی بلوک constant موجود در شکل فوق به صورت زیر وارد می شود:



پارامترهای اصلی سیستم شامل جرم، طول بازو و .... نیز با دابل کلیک کردن بر روی بلوک constant2 به صورت زیر وارد می شوند:



و اما بلوک اصلی مربوط به یافتن شتاب ربات، با گرفتن ورودی های فوق به صورت زیر انجام می شود:

```
function xdd = fcn(x,T,xds,para)
D=zeros(6,1);
D=[0.072*(xds(1))^2;-
0.05*(xds(2))^2;0.065*(xds(3))^2;0.065*(xds(4))^2;0.001*(xds(5))^2;0.0001*(x
ds(6))^2];
xdd=zeros(6,1);
L1=para(1);
L2=para(2);
M=para(3);
J1=para(4);
J2=para(5);
J3=para(6);
g=9.81;
%#codegen
p=x(4);
r=x(5);
y=x(5);
xdd(1)=(sum(T)*g*(cos(y)*sin(p)*cos(r)+sin(y)*sin(r))-D(1))/M;
xdd(2)=(sum(T)*g*(sin(y)*sin(p)*cos(r)-cos(y)*sin(r))-D(2))/M;
xdd(3)=(sum(T)*g*(cos(y)*cos(r))-M*g-D(3))/M;
xdd(4)=(L1*(-T(1)-T(2)+T(3)+T(4))-D(4))/J1;
xdd(5)=(L2*(-T(1)+T(2)+T(3)-T(4))-D(5))/J2;
xdd(6)=(5*(T(1)-T(2)+T(3)-T(4))-D(6))/J3;
```

پس از یافتن شتاب ربات بوسیله تابع فوق، با دوبار انتگرال گیری از آن موقعیت و orientation ربات بدست می آید.

به منظور تست معادلات دینامیک سیستم، رفتار سیستم براساس تراست های ورودی مختلفی مورد بررسی قرار گرفته است، اما لازم به ذکر است که به علت پارامترهایی که مقدار آن ها در مقاله داده نشده بود نتایج با نتایج ارائه شده در مقاله کاملا یکسان نبود.

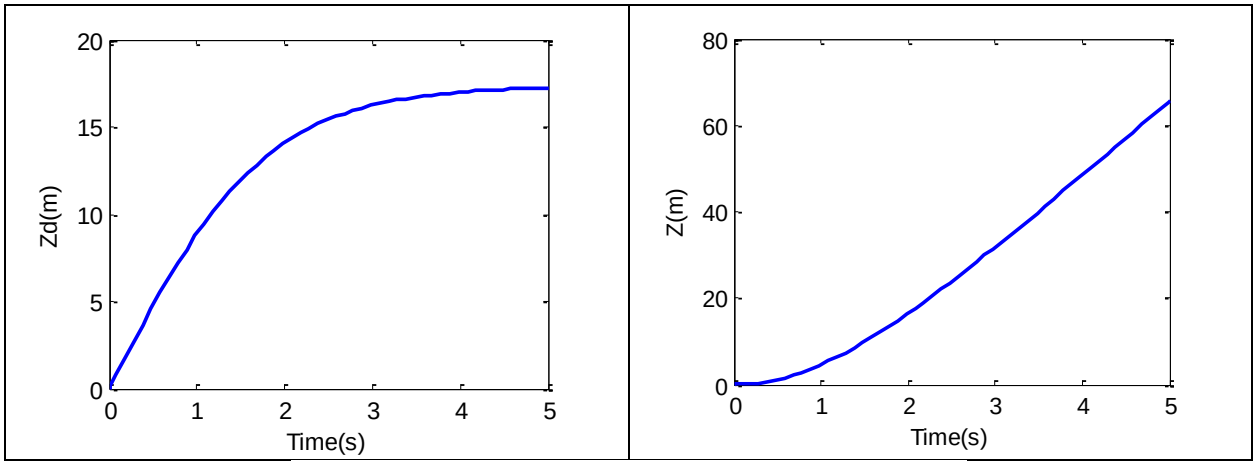


Fig. 4. Vertical motion: (T1, T2, T3, T4) = (1, 1, 1, 1).

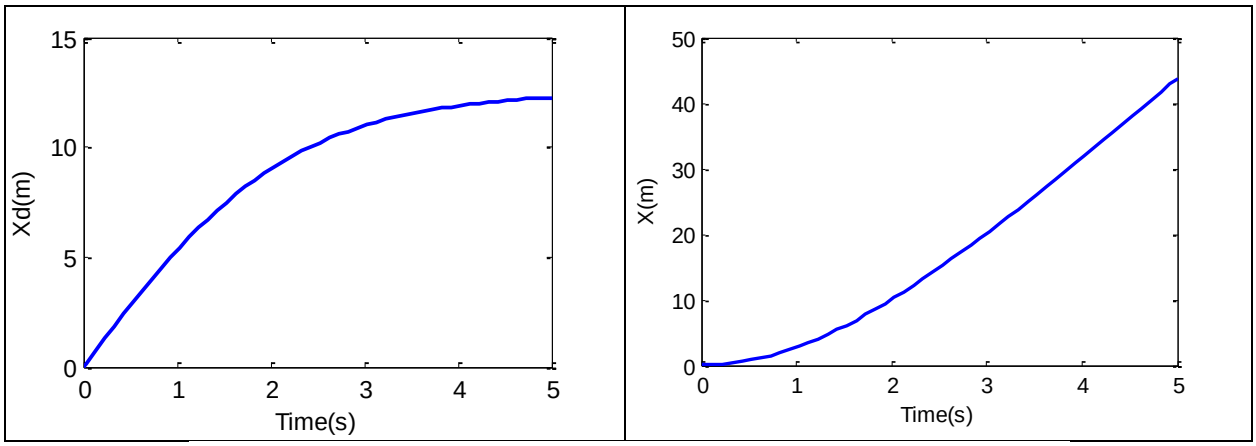


Fig. 5. Forward motion (tilting angle, 30°): (T1, T2, T3, T4) = (0.5/cos (30°), 0.5/cos (30°), 0.5/cos (30°), 0.5/cos (30°)).

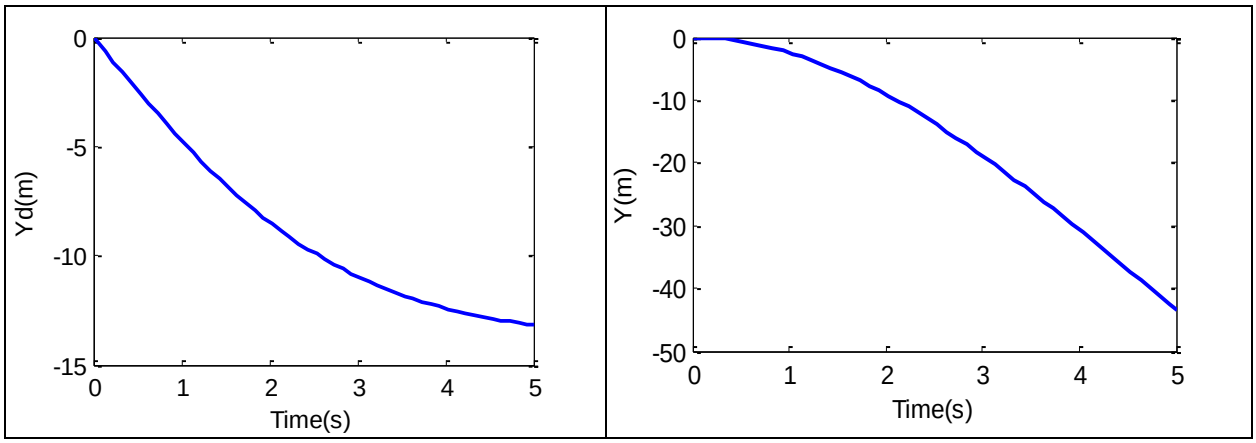


Fig. 6. Lateral motion (tilting angle, 30°): (T1, T2, T3, T4) = (0.5/cos (30°), 0.5/cos (30°), 0.5/cos (30°), 0.5/cos (30°)).

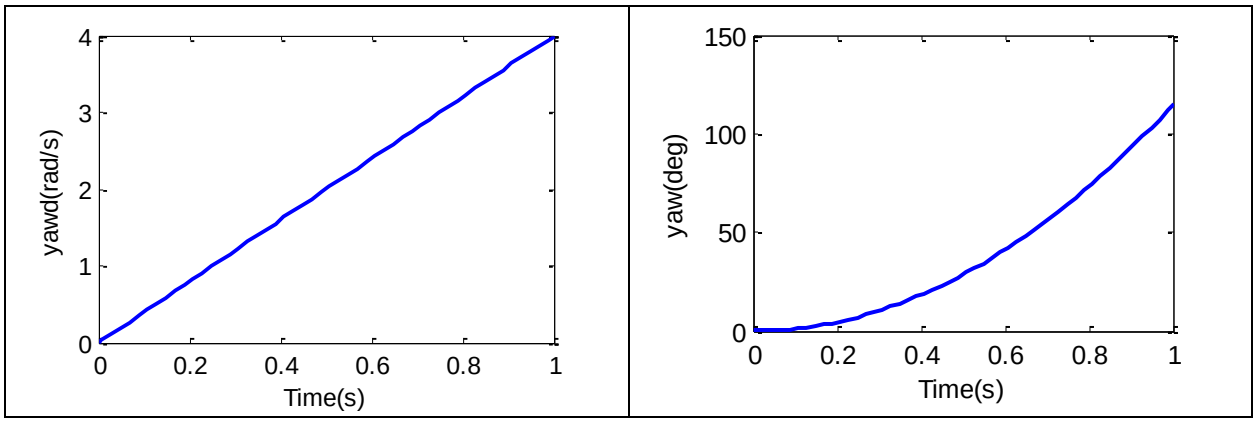


Fig. 7. Yawing motion; (T1, T2, T3, T4) = (0.6, 0.4, 0.6, 0.4).

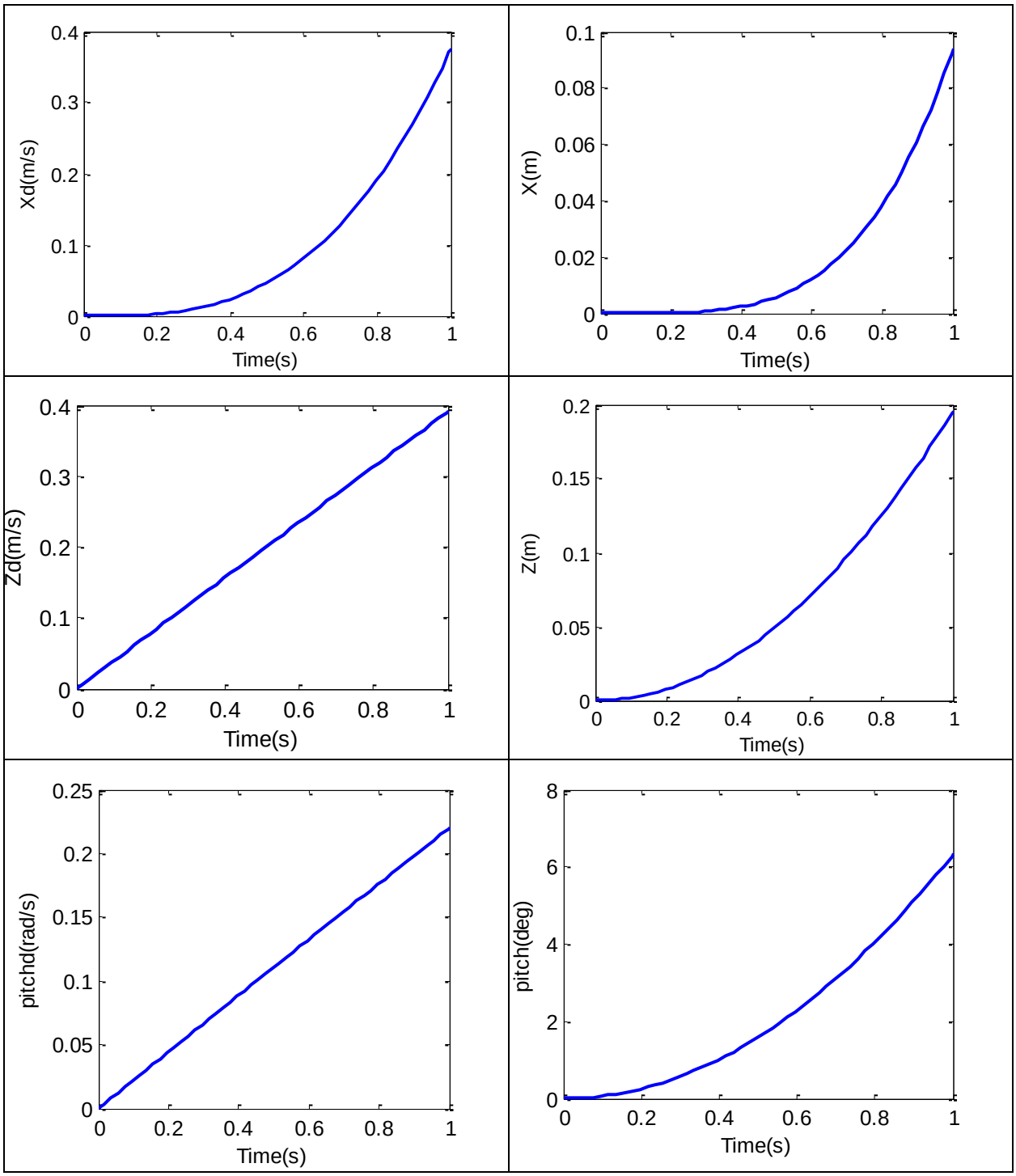


Fig. 8. Pitching motion: (T1, T2, T3, T4) = (0.42, 0.42, 0.62, 0.62).

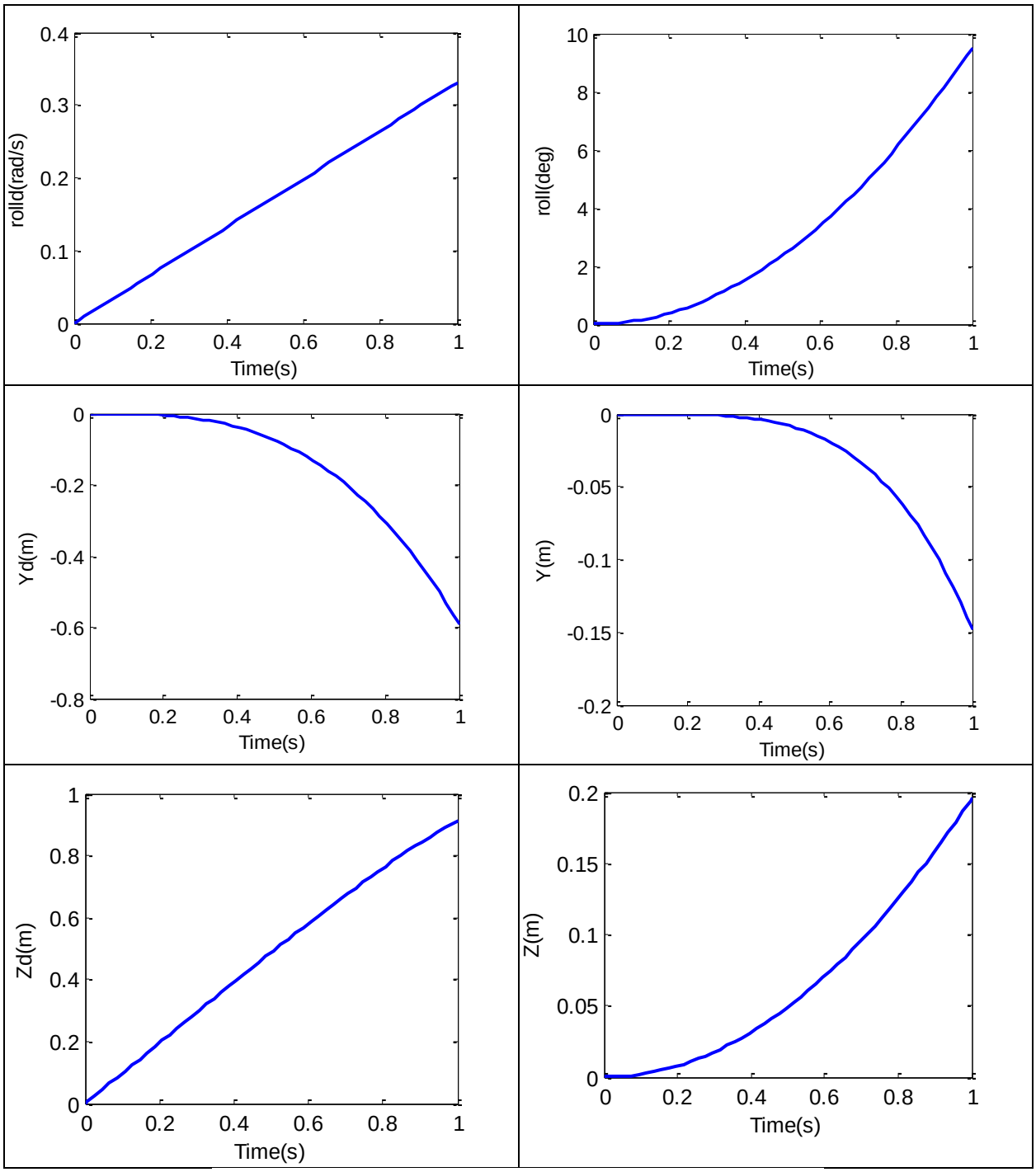


Fig. 9. Rolling motion: (T1, T2, T3, T4) = (0.4, 0.7, 0.4, 0.7).

لازم به ذکر است که این مقاله دارای اشتباهات زیادی می باشد. به عنوان مثال پارامترهای داده شده در شکل ۹، با توجه به معادلات دینامیک داده شده حرکت رول ایجاد نمی کند و بنابراین جانمایی این اعداد به گونه ای

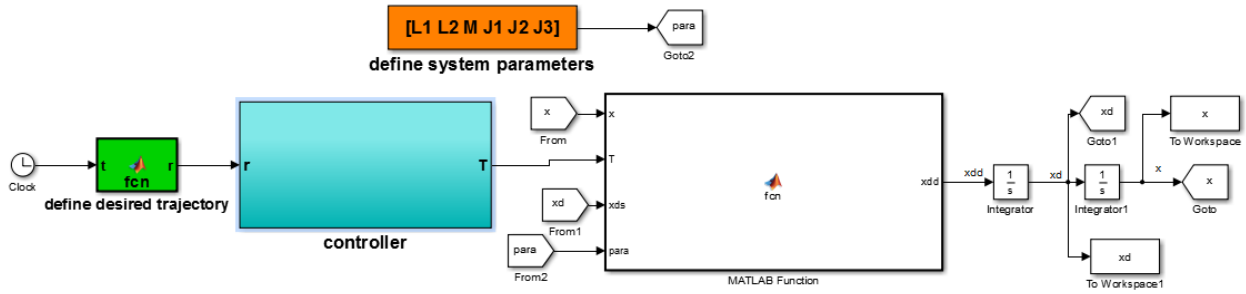
تغییر داده شد که حرکت رول ایجاد کند. دیگر اشتباه مربوط به جانمایی پارامترهای درگ در معادلات دینامیک می باشد به عنوان مثال رابطه زیر می بایست به صورت زیر نوشته شد:

$$\ddot{z} = \frac{(\sum_{i=1}^{4} T_i)(\cos\phi\cos\psi - mg - D_3)}{m} \qquad \rightarrow \quad \ddot{z} = \frac{\left(\sum_{i=1}^{4} T_i\right)\cos(\phi)\cos(\psi) - mg - D_3}{m}$$

جهت یافتن نتایج فوق کافیست ابتدا فایل simu.slx را با نرم افزار مطلب باز کنید و تراست های موردنظر را وارد کنید و سپس run کنید. نتایج به صورت اتوماتیک بعد از تمام شدن run برای تمام state ها مشاهده خواهد شد.

گام دوم: طراحی کنترلر فازی تطبیقی:

در این مرحله یک کنترلر فازی تطبیقی بر روی مدل دینامیک شبیه سازی شده در نرم افزار سیمولینک اعمال گردید نمایی از این سیستم در زیر نشان داده شده است:



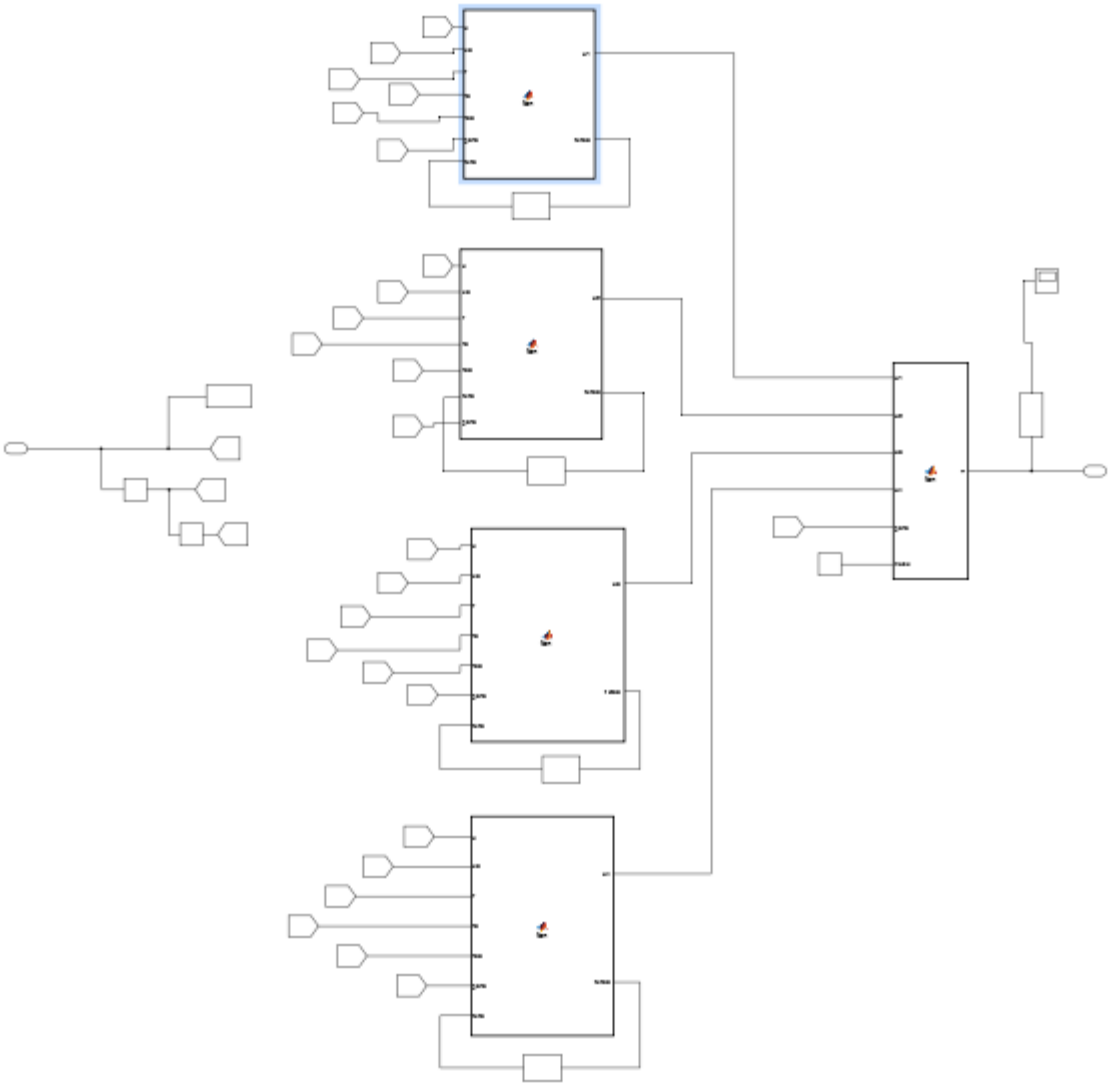
به علت پیچیده شدن سیستم از بلوک های from و go to استفاده شده است. بدین صورت که پارامترها در این بلوک ها ذخیره می شوند و در مواردی که بخواهیم از این پارامترها استفاده کنیم از بلوک from مربوط به این بلوک ها استفاده می کنیم.

بلوک define desired trajectory بلوکی می باشد که با گرفتن زمان، قابلیت تعریف مسیر مطلوب برای پارامترهای ارتفاع، پیچ، رول و یاو فراهم می کند:

```
function r = fcn(t)
r=zeros(4,1);
%#codegen
```

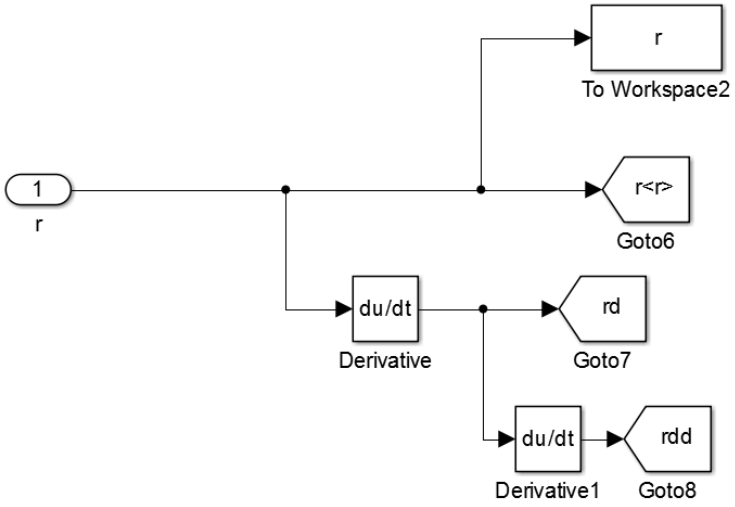
```
r(1)=0;
r(2)=0;
r(3)=0;
r(4)=0.8*sin(0.1*t);
```

بلوک مربوط به controller با دریافت مسیر مطلوب (r)، position و orientation، velocity و angular velocity، به کنترل سیستم می پردازد. در واقع با دریافت پارامترهای ذکر شده، T_1، T_2 ، T_3 و T_4 را به گونه ای تنظیم می کند که سیستم براساس مسیر مطلوب تعیین شده حرکت کند. این بلوک شامل زیر بخش های زیر است:

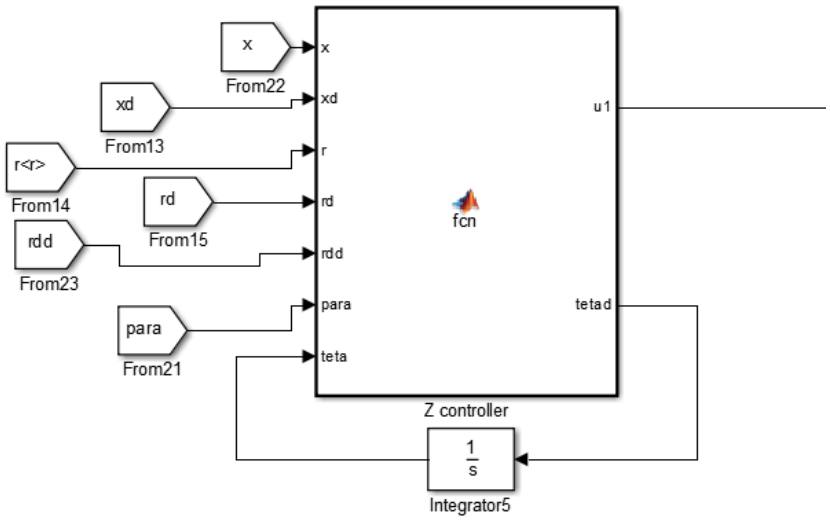


در زیر به ارائه هر کدام از این بخش ها پرداخته می شود:

سیستم زیر با گرفتن مسیر مطلوب، به محاسبه سرعت های مطلوب و شتاب مطلوب می پردازد.



کنترل فازی تطبیقی مربوط به پارامتر Z:

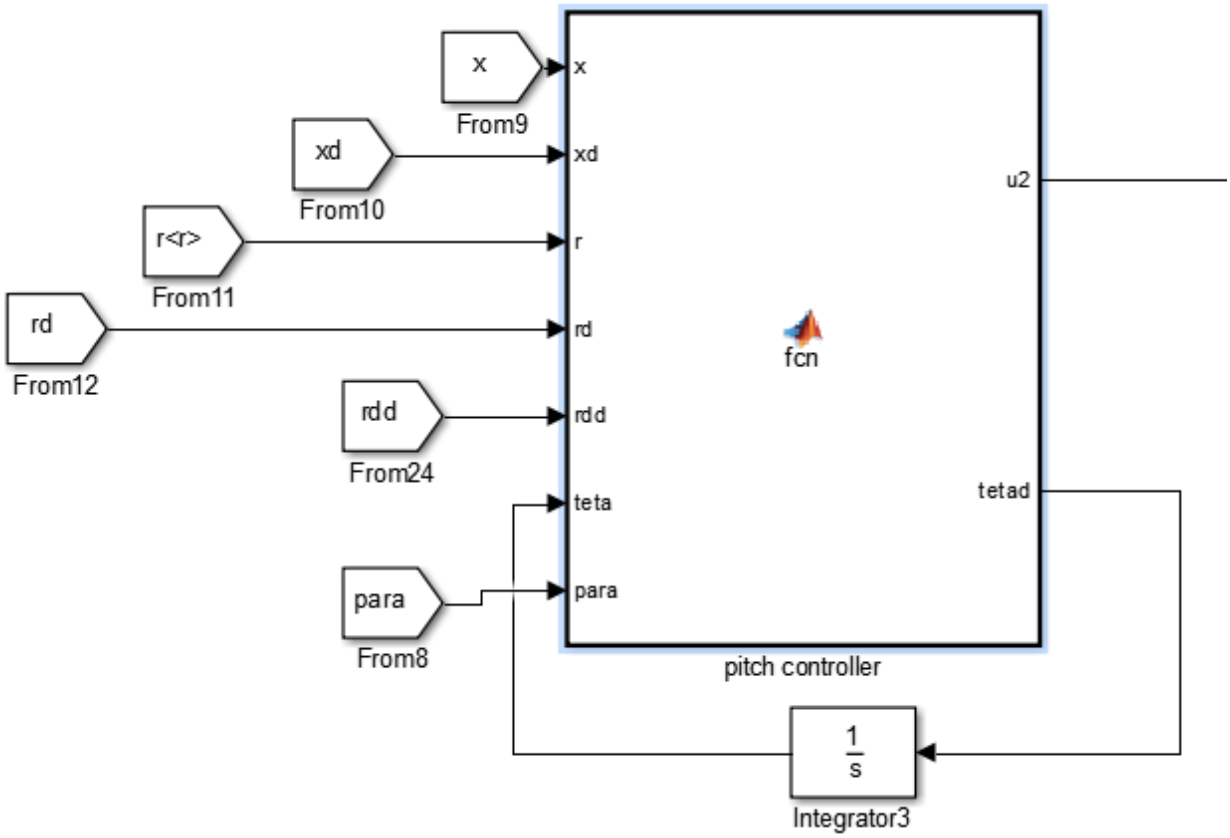


تابع z controller:

```
function [u1,tetad] = fcn(x,xd,r,rd,rdd,para,teta)
zeta=zeros(49,1);
%% ======================================
pit=x(4);
rol=x(5);
ya=x(5);
%System parameters
M=para(3);
g=9.81;
%% Control parameters:
b=(cos(ya)*cos(rol))*g/M;
Ac=[0 1;-1 -2];
bc=[0;b];

Q=[10 0;0 10];
P=[15 5;5 5];
Pn=P(:,2);
gama=2;
Mx=0.2;
Mtet=3;
bL=0.5;
k=[-Ac(2,2) -Ac(2,1)]';
%%
%#codegen
%Error on Z and ZD
z=r(1)-x(3);
zd=rd(1)-xd(3);

SS=[NB(z)*NB(zd),NB(z)*NM(zd),NB(z)*NS(zd),NB(z)*Z(zd),NB(z)*PS(zd),NB(z)*PM
(zd),NB(z)*PB(zd),...

NM(z)*NB(zd),NM(z)*NM(zd),NM(z)*NS(zd),NM(z)*Z(zd),NM(z)*PS(zd),NM(z)*PM(zd)
,NM(z)*PB(zd),...

NS(z)*NB(zd),NS(z)*NM(zd),NS(z)*NS(zd),NS(z)*Z(zd),NS(z)*PS(zd),NS(z)*PM(zd)
,NS(z)*PB(zd),...

Z(z)*NB(zd),Z(z)*NM(zd),Z(z)*NS(zd),Z(z)*Z(zd),Z(z)*PS(zd),Z(z)*PM(zd),Z(z)*
PB(zd),...

PS(z)*NB(zd),PS(z)*NM(zd),PS(z)*NS(zd),PS(z)*Z(zd),PS(z)*PS(zd),PS(z)*PM(zd)
,PS(z)*PB(zd),...

PM(z)*NB(zd),PM(z)*NM(zd),PM(z)*NS(zd),PM(z)*Z(zd),PM(z)*PS(zd),PM(z)*PM(zd)
,PM(z)*PB(zd),...

PB(z)*NB(zd),PB(z)*NM(zd),PB(z)*NS(zd),PB(z)*Z(zd),PB(z)*PS(zd),PB(z)*PM(zd)
,PB(z)*PB(zd)];
% Z controller:
for i=1:49
%% for 49 rule, zeta is evaluated
```

```
zeta(i)=SS(i)/sum(SS);
end
e=[z zd]';
tetad=gama*e'*Pn*zeta;
uc=teta'*zeta;
V=0.5*e'*P*e;


I=1;

us=I*sign(e'*P*bc)*(abs(uc)+1/bL*(-
0.3543372*xd(3)^2+abs(rdd(1))+abs(k'*e)));

u1=uc+us;
end
%=================================================================================
====================================
function y=NB(x)
y=exp(-25*(x+1)^2);
end
function y=NM(x)
y=exp(-25*(x+0.6666)^2);
end
function y=NS(x)
y=exp(-25*(x+0.3334)^2);
end
function y=Z(x)
y=exp(-25*(x+0)^2);
end
function y=PS(x)
y=exp(-25*(x-0.3334)^2);
end
function y=PM(x)
y=exp(-25*(x-0.6666)^2);
end
function y=PB(x)
y=exp(-25*(x-1)^2);
end
```

کنترل فازی تطبیقی مربوط به پارامتر pitch:



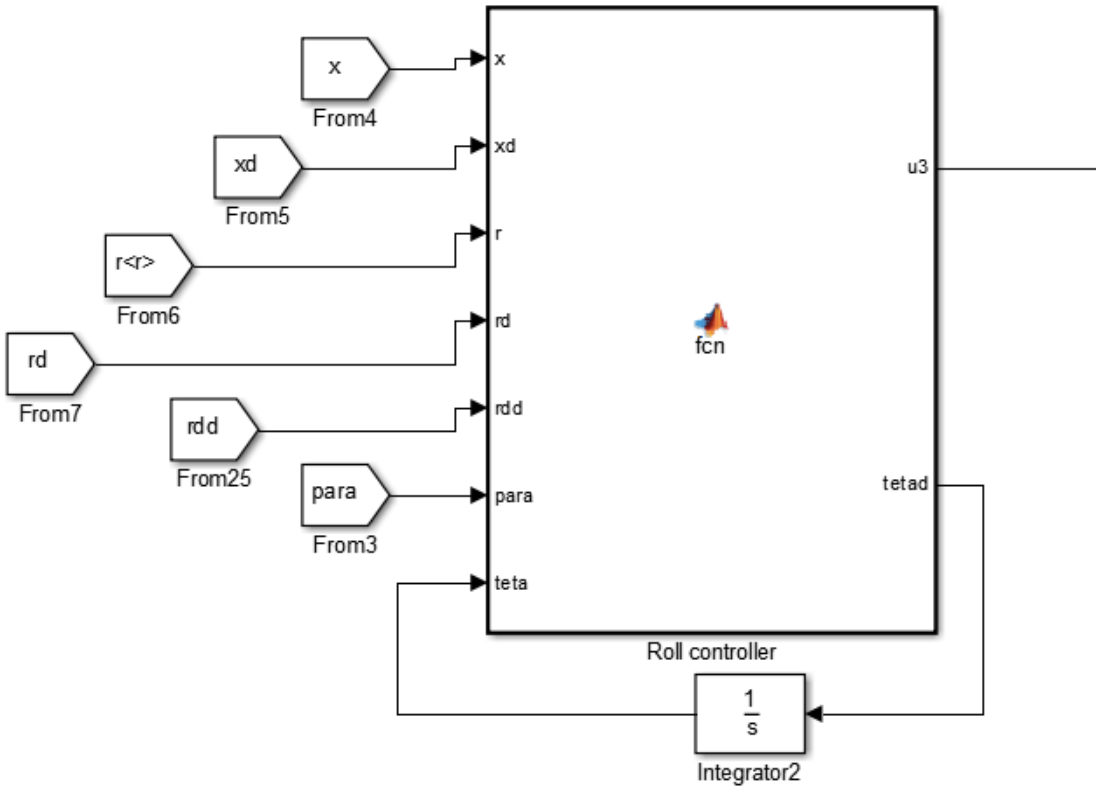
تابع pitch controller:

```
function [u2,tetad] = fcn(x,xd,r,rd,rdd,teta,para)
u=zeros(4,1);
zeta=zeros(49,1);
%% ======================================
%System parameters
J1=para(4);
g=9.81;
%% Control parameters:
b=1/J1;
Ac=[0 1;-1 -2];
bc=[0;b];

Q=[10 0;0 10];
```

```
P=[15 5;5 5];
Pn=P(:,2);
gama=2;
Mx=0.2;
Mtet=3;
bL=0.5;
k=[-Ac(2,2) -Ac(2,1)]';
%%
%#codegen
%Error on pitch and pitchD
pitch=r(2)-x(4);
pitchd=rd(2)-xd(4);
SS=[NB(pitch)*NB(pitchd),NB(pitch)*NM(pitchd),NB(pitch)*NS(pitchd),NB(pitch)
*Z(pitchd),NB(pitch)*PS(pitchd),NB(pitch)*PM(pitchd),NB(pitch)*PB(pitchd),..
.

NM(pitch)*NB(pitchd),NM(pitch)*NM(pitchd),NM(pitch)*NS(pitchd),NM(pitch)*Z(p
itchd),NM(pitch)*PS(pitchd),NM(pitch)*PM(pitchd),NM(pitch)*PB(pitchd),...

NS(pitch)*NB(pitchd),NS(pitch)*NM(pitchd),NS(pitch)*NS(pitchd),NS(pitch)*Z(p
itchd),NS(pitch)*PS(pitchd),NS(pitch)*PM(pitchd),NS(pitch)*PB(pitchd),...

Z(pitch)*NB(pitchd),Z(pitch)*NM(pitchd),Z(pitch)*NS(pitchd),Z(pitch)*Z(pitch
d),Z(pitch)*PS(pitchd),Z(pitch)*PM(pitchd),Z(pitch)*PB(pitchd),...

PS(pitch)*NB(pitchd),PS(pitch)*NM(pitchd),PS(pitch)*NS(pitchd),PS(pitch)*Z(p
itchd),PS(pitch)*PS(pitchd),PS(pitch)*PM(pitchd),PS(pitch)*PB(pitchd),...

PM(pitch)*NB(pitchd),PM(pitch)*NM(pitchd),PM(pitch)*NS(pitchd),PM(pitch)*Z(p
itchd),PM(pitch)*PS(pitchd),PM(pitch)*PM(pitchd),PM(pitch)*PB(pitchd),...

PB(pitch)*NB(pitchd),PB(pitch)*NM(pitchd),PB(pitch)*NS(pitchd),PB(pitch)*Z(p
itchd),PB(pitch)*PS(pitchd),PB(pitch)*PM(pitchd),PB(pitch)*PB(pitchd)];

% Z controller:
for i=1:49
%% for 49 rule, zeta is evaluated
zeta(i)=SS(i)/sum(SS);
end
e=[pitch pitchd]';
tetad=gama*e'*Pn*zeta;
uc=teta'*zeta;
V=0.5*e'*P*e;
if V <0.8
I=1;
else
I=0;
end
us=I*sign(e'*P*bc)*(abs(uc)+1/bL*(-0.3543372*xd(4)^2+abs(k'*e)));
if us>10
us=0;
end
u2=uc+us;
end
%=========================================================================
====================================
```

```
function y=NB(x)
y=exp(-(x+1)^2);
end
function y=NM(x)
y=exp(-(x+0.6666)^2);
end
function y=NS(x)
y=exp(-(x+0.3334)^2);
end
function y=Z(x)
y=exp(-(x+0)^2);
end
function y=PS(x)
y=exp(-(x-0.3334)^2);
end
function y=PM(x)
y=exp(-(x-0.6666)^2);
end
function y=PB(x)
y=exp(-(x-1)^2);
end%=================================================================================
=========================================
```

کنترل فازی تطبیقی مربوط به پارامتر ROLL:



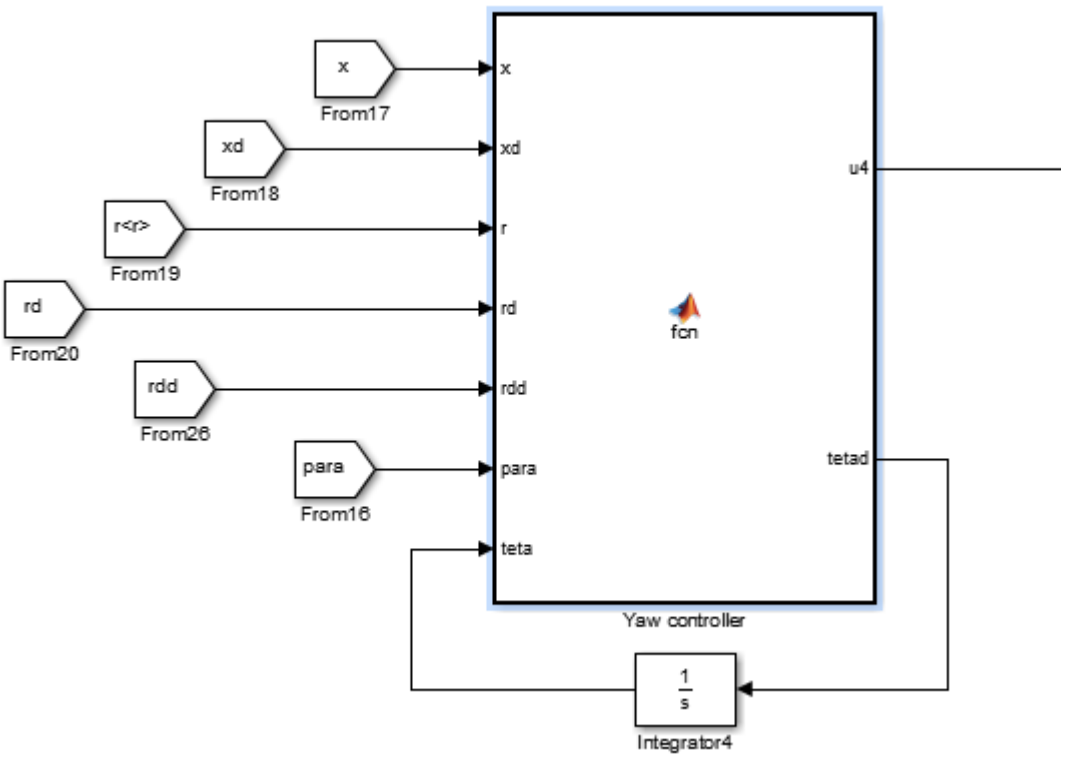
تابع roll controller:

```
function [u3,tetad] = fcn(x,xd,r,rd,rdd,para,teta)
zeta=zeros(49,1);
%% ========================================
%System parameters
J2=para(5);
%% Control parameters:
g=9.81;

b=1/J2;
Ac=[0 1;-1 -2];
bc=[0;b];

Q=[10 0;0 10];
P=[15 5;5 5];
Pn=P(:,2);
gama=2;
Mx=0.2;
Mtet=3;
bL=0.5;
k=[-Ac(2,2) -Ac(2,1)]';
%%
%#codegen
%Error on Roll and RollD
roll=r(3)-x(5);
rollD=rd(3)-xd(5);
```

```
SS=[NB(roll)*NB(rollD),NB(roll)*NM(rollD),NB(roll)*NS(rollD),NB(roll)*Z(roll
D),NB(roll)*PS(rollD),NB(roll)*PM(rollD),NB(roll)*PB(rollD),...

NM(roll)*NB(rollD),NM(roll)*NM(rollD),NM(roll)*NS(rollD),NM(roll)*Z(rollD),N
M(roll)*PS(rollD),NM(roll)*PM(rollD),NM(roll)*PB(rollD),...

NS(roll)*NB(rollD),NS(roll)*NM(rollD),NS(roll)*NS(rollD),NS(roll)*Z(rollD),N
S(roll)*PS(rollD),NS(roll)*PM(rollD),NS(roll)*PB(rollD),...

Z(roll)*NB(rollD),Z(roll)*NM(rollD),Z(roll)*NS(rollD),Z(roll)*Z(rollD),Z(rol
l)*PS(rollD),Z(roll)*PM(rollD),Z(roll)*PB(rollD),...

PS(roll)*NB(rollD),PS(roll)*NM(rollD),PS(roll)*NS(rollD),PS(roll)*Z(rollD),P
S(roll)*PS(rollD),PS(roll)*PM(rollD),PS(roll)*PB(rollD),...

PM(roll)*NB(rollD),PM(roll)*NM(rollD),PM(roll)*NS(rollD),PM(roll)*Z(rollD),P
M(roll)*PS(rollD),PM(roll)*PM(rollD),PM(roll)*PB(rollD),...

PB(roll)*NB(rollD),PB(roll)*NM(rollD),PB(roll)*NS(rollD),PB(roll)*Z(rollD),P
B(roll)*PS(rollD),PB(roll)*PM(rollD),PB(roll)*PB(rollD)];
% Z controller:
for i=1:49
%% for 49 rule, zeta is evaluated
zeta(i)=SS(i)/sum(SS);
end
e=[roll rollD]';
tetad=gama*e'*Pn*zeta;
uc=teta'*zeta;
V=0.5*e'*P*e;
if V <1
I=1;
else I=0;
end
us=I*sign(e'*P*bc)*(abs(uc)+1/bL*(-0.3543372*xd(5)^2+abs(k'*e)));
u3=uc+us;
end
function y=NB(x)
y=exp(-(x+2)^2);
end
function y=NM(x)
y=exp(-(x+1.33)^2);
end
function y=NS(x)
y=exp(-(x+0.66)^2);
end
function y=Z(x)
y=exp(-(x+0)^2);
end
function y=PS(x)
y=exp(-(x-0.66)^2);
end
function y=PM(x)
```

```
y=exp(-(x-1.33)^2);
end
function y=PB(x)
y=exp(-(x-2)^2);
end
%=========================================================================
=====================================
```

کنترل فازی تطبیقی مربوط به پارامتر yaw:



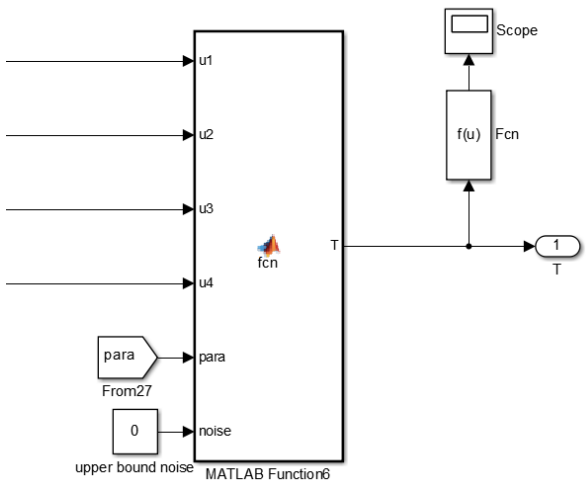
تابع yaw controller:

```
function [u4,tetad] = fcn(x,xd,r,rd,rdd,para,teta)
zeta=zeros(49,1);
%% ======================================
%System parameters
J3=para(6);
%% Control parameters:
g=9.81;
```

```
b=1/J3;
Ac=[0 1;-1 -2];
bc=[0;b];

Q=[10 0;0 10];
P=[15 5;5 5];
Pn=P(:,2);
gama=2;
Mx=0.2;
Mtet=3;
bL=0.5;
k=[-Ac(2,2) -Ac(2,1)]';
%%
%#codegen
%Error on Z and ZD
yaw=r(4)-x(6);
yawd=rd(4)-xd(6);


SS=[NB(yaw)*NB(yawd),NB(yaw)*NM(yawd),NB(yaw)*NS(yawd),NB(yaw)*Z(yawd),NB(ya
w)*PS(yawd),NB(yaw)*PM(yawd),NB(yaw)*PB(yawd),...

NM(yaw)*NB(yawd),NM(yaw)*NM(yawd),NM(yaw)*NS(yawd),NM(yaw)*Z(yawd),NM(yaw)*P
S(yawd),NM(yaw)*PM(yawd),NM(yaw)*PB(yawd),...

NS(yaw)*NB(yawd),NS(yaw)*NM(yawd),NS(yaw)*NS(yawd),NS(yaw)*Z(yawd),NS(yaw)*P
S(yawd),NS(yaw)*PM(yawd),NS(yaw)*PB(yawd),...

Z(yaw)*NB(yawd),Z(yaw)*NM(yawd),Z(yaw)*NS(yawd),Z(yaw)*Z(yawd),Z(yaw)*PS(yaw
d),Z(yaw)*PM(yawd),Z(yaw)*PB(yawd),...

PS(yaw)*NB(yawd),PS(yaw)*NM(yawd),PS(yaw)*NS(yawd),PS(yaw)*Z(yawd),PS(yaw)*P
S(yawd),PS(yaw)*PM(yawd),PS(yaw)*PB(yawd),...

PM(yaw)*NB(yawd),PM(yaw)*NM(yawd),PM(yaw)*NS(yawd),PM(yaw)*Z(yawd),PM(yaw)*P
S(yawd),PM(yaw)*PM(yawd),PM(yaw)*PB(yawd),...

PB(yaw)*NB(yawd),PB(yaw)*NM(yawd),PB(yaw)*NS(yawd),PB(yaw)*Z(yawd),PB(yaw)*P
S(yawd),PB(yaw)*PM(yawd),PB(yaw)*PB(yawd)];
% Z controller:
for i=1:49
%% for 49 rule, zeta is evaluated
zeta(i)=SS(i)/sum(SS);
end
e=[yaw yawd]';
tetad=gama*e'*Pn*zeta;
uc=teta'*zeta;
V=0.5*e'*P*e;
if V <1
I=1;
else I=0;
end
us=I*sign(e'*P*bc)*(abs(uc)+1/bL*(-0.3543372*xd(6)^2+abs(k'*e)));
u4=uc+us;
end
```

```
%=========================================================================
====================================
function y=NB(x)
y=exp(-0.9*(x+2)^2);
end
function y=NM(x)
y=exp(-0.9*(x+1.33)^2);
end
function y=NS(x)
y=exp(-0.9*(x+0.66)^2);
end
function y=Z(x)
y=exp(-0.9*(x+0)^2);
end
function y=PS(x)
y=exp(-0.9*(x-0.66)^2);
end
function y=PM(x)
y=exp(-0.9*(x-1.33)^2);
end
function y=PB(x)
y=exp(-0.9*(x-2)^2);
end
```

در نهایت با محاسبه u1 ، u2 ، u3 و u4 براساس بلوک زیر T1، T2، T3 و T4 مورد محاسبه قرار می گیرد که به مدل دینامیک سیستم ارسال می گردد:

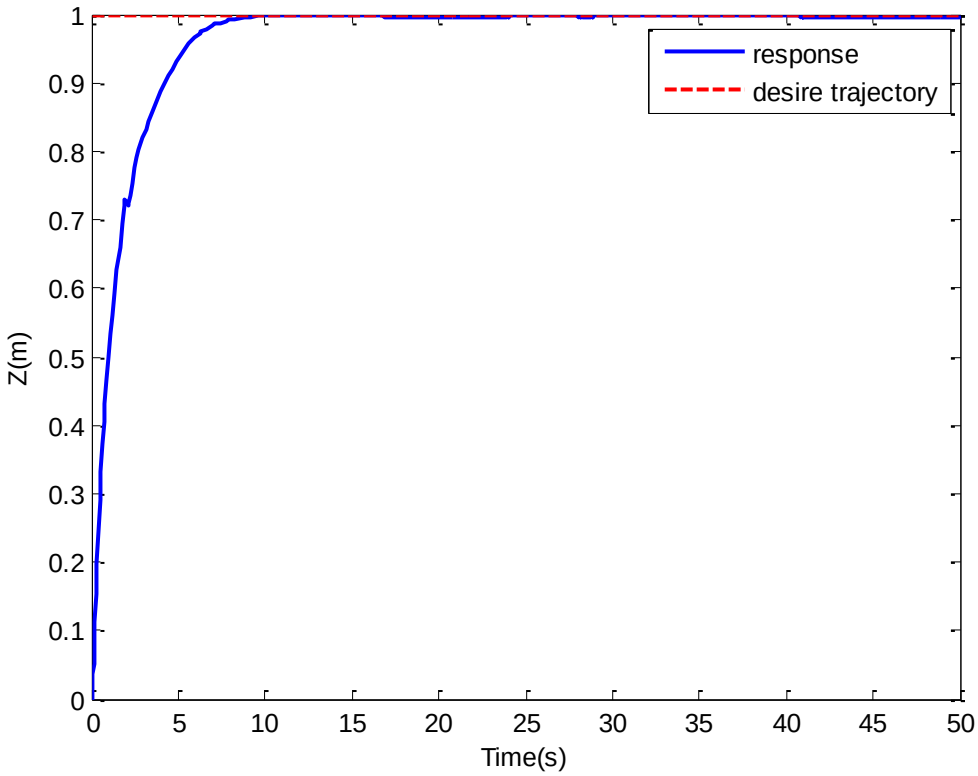



```
function T = fcn(u1,u2,u3,u4,para,noise)
%#codegen
L1=para(1);
```

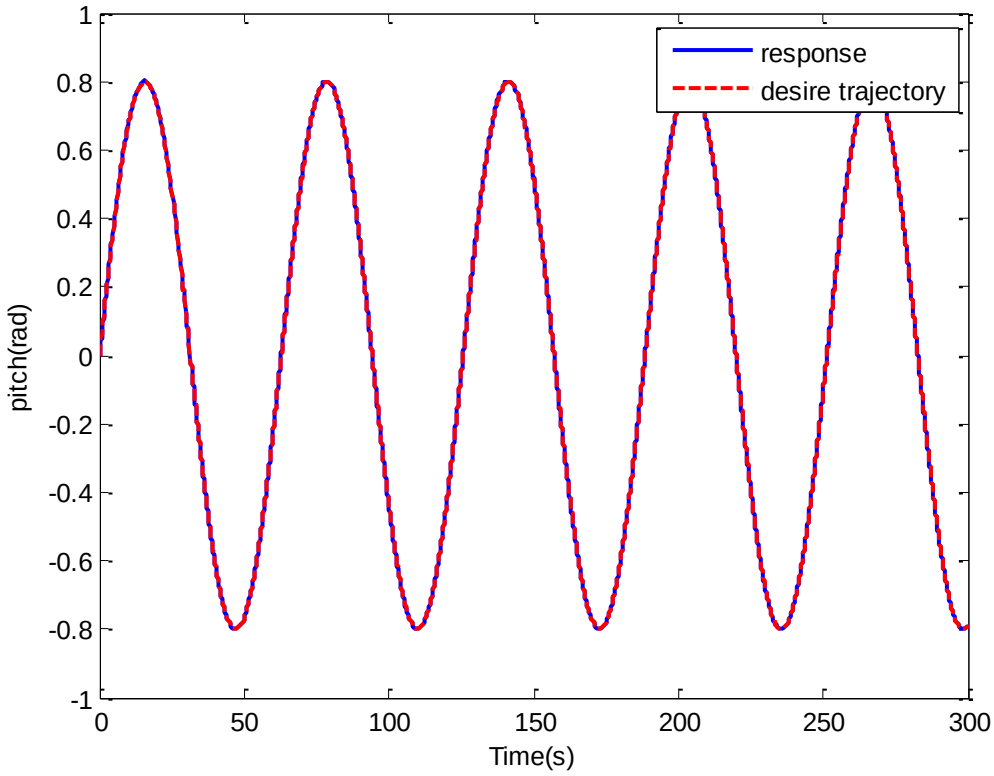
```
L2=para(2);
T=zeros(4,1);
noise =noise *rand;
[u1,u2,u3,u4];

T(1)=-(5*L1*u3 + 5*L2*u2 - 5*L1*L2*u1 - L1*L2*u4)/(20*L1*L2)+noise;
T(2)=(5*L1*u3 - 5*L2*u2 + 5*L1*L2*u1 - L1*L2*u4)/(20*L1*L2)+noise;
T(3)=(5*L1*u3 + 5*L2*u2 + 5*L1*L2*u1 + L1*L2*u4)/(20*L1*L2)+noise;
T(4)=-(5*L1*u3 - 5*L2*u2 - 5*L1*L2*u1 + L1*L2*u4)/(20*L1*L2)+noise;
for i=1:4
if T(i)>7
T(i)=7;
end
end
```

توابع عضویت فازی مورد استفاده، پارامترهای کنترلی، rull ها، قوانین کنترلی و .... همه براساس پارامترهای داده شده در مقاله، شبیه سازی شده است.

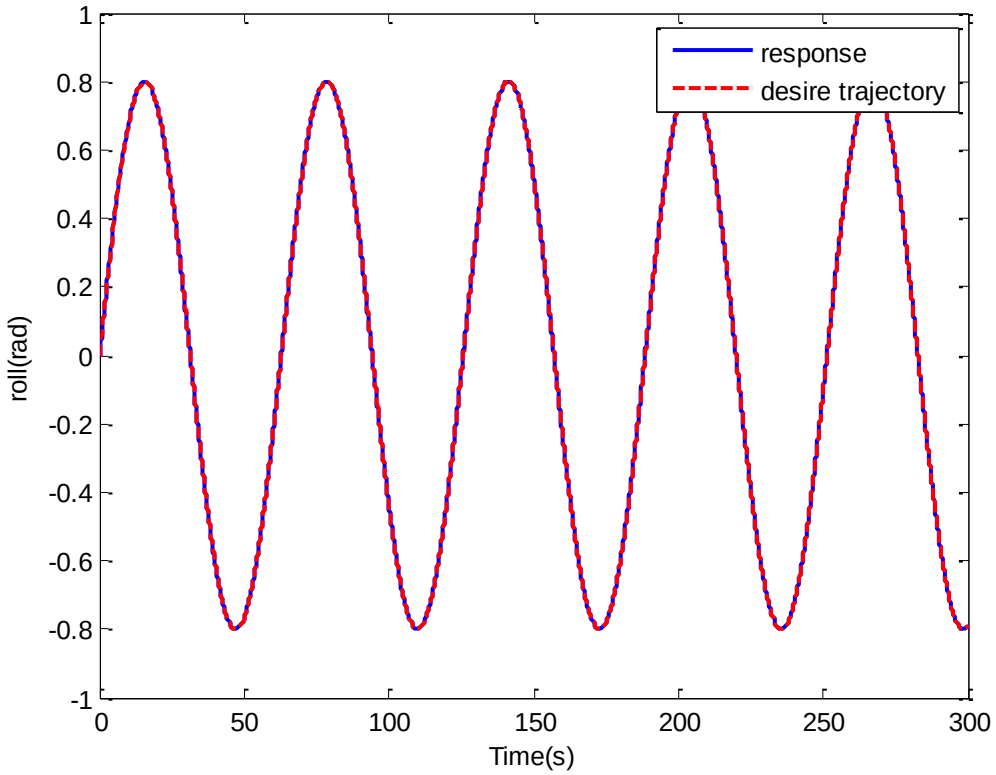
نتایج حاصل از اعمال کنترلر:



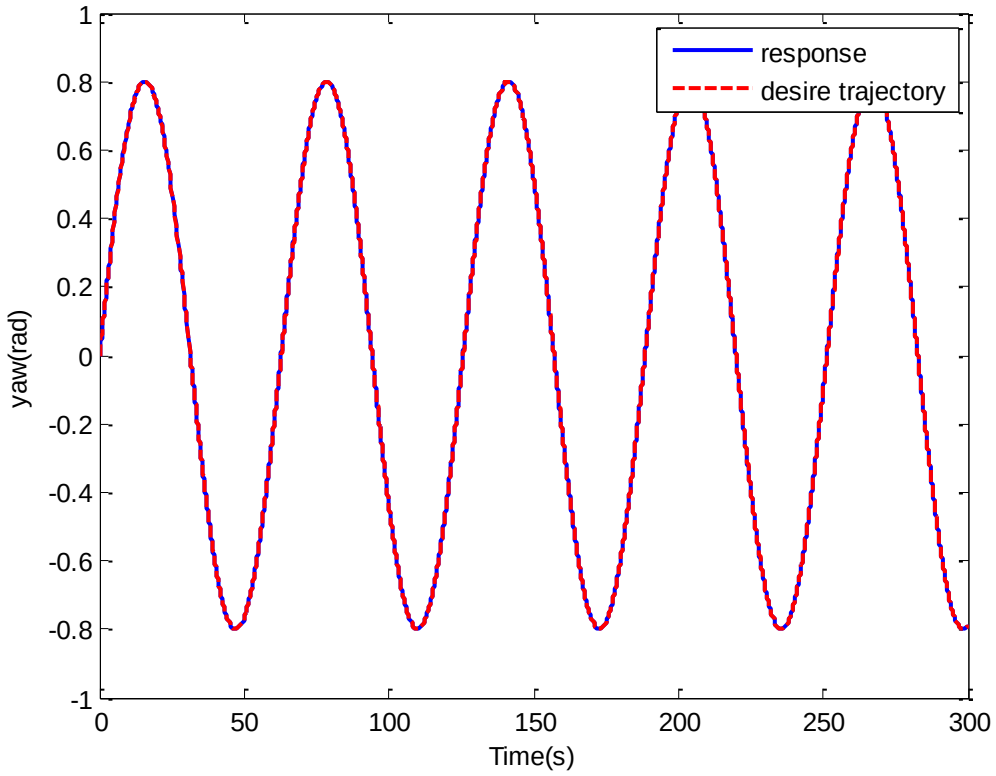
برای $Z_{ref} = 1$



برای $pitch_{ref} = 0.8*\sin(0.1*t)$



برای $roll_{ref} = 0.8*\sin(0.1*t)$



برای $yaw_{ref} = 0.8*\sin(0.1*t)$

همانطور که در نتایج فوق مشهود است کنترلر طراحی شده توانسته است به خوبی توانسته است مسیر مطلوب مورد نظر را دنبال کند.

جهت یافتن نتایج فوق کافیست ابتدا فایل simu_controller.slx را با نرم افزار مطلب باز کنید و مسیر مطلوب موردنظر را در بلوک موردنظر تعریف کنید و سپس run کنید. نتایج به صورت اتوماتیک بعد از تمام شدن run برای تمام state ها مشاهده خواهد شد.