به نام خدا



**گزارش تمرین شماره 8رباتیک**

نام دانشجو : **عرفان رادفر**

شماره دانشجویی : **99109603**

استاد درس : **دکتر سعید بهزادی پور**



پاییز 1402

فهرست

[سوال 1 2](#_Toc155753451)

[سوال 2 3](#_Toc155753452)

[سوال 3 7](#_Toc155753453)

[سوال 4 (امتیازی) 10](#_Toc155753454)

# سوال 1

با توجه به خواسته مسئله ، ربات و لینک های آن را در جعبه افزار Simulink ، بخش simscape طراحی می­کنیم. این مدل را در model\_1.slx مشاهده می­کنید. البته قبل اجرا کردن مدل، این بخش از کد تابع main را اجرا کنید تا پارامترهای مدل در workspace ذخیره شوند.(در تصویر مدل ربات سمت چپ، اجزا نسبت به ورودی صفر جابه­جا شده اند و در تصویر راست، تمام مفاصل در اندازه صفر قرار دارند.)

Ra=50;

Jm=0.1;

Jg=0.05;

r=50;

Bm=0.01;

Km=7.5; Kb=Km;

J=Jg+Jm;

B=Bm+Km\*Kb/Ra;

radius=0.010 ; % in meters

max\_v=1e6; %volt

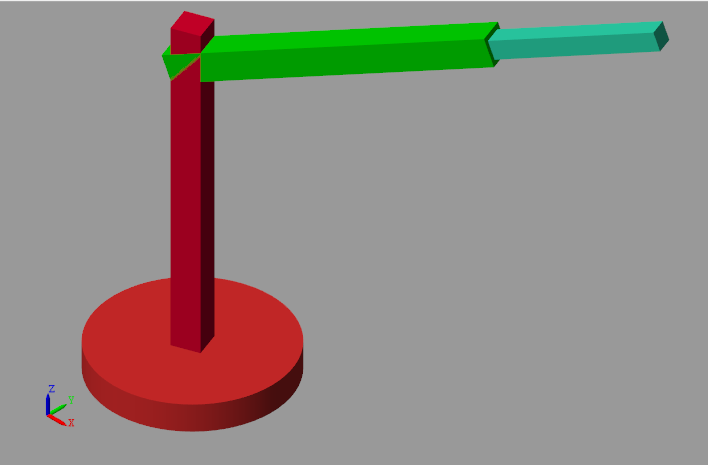
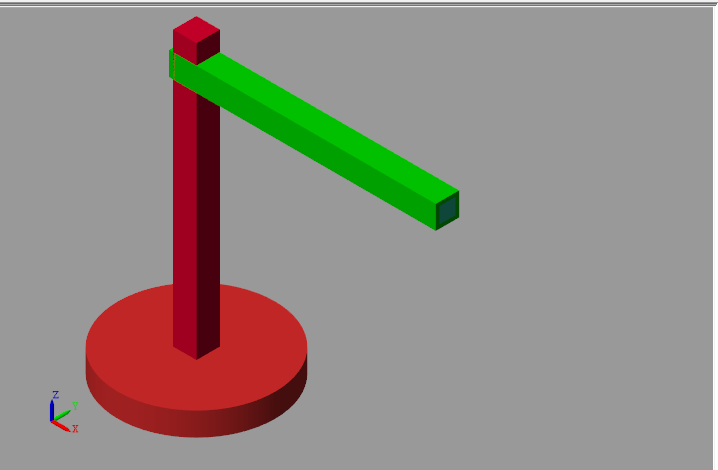
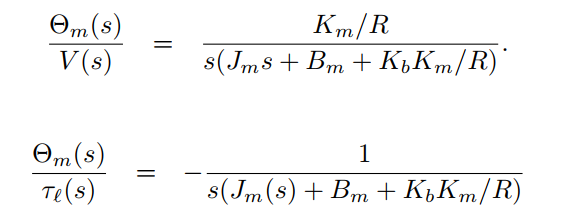


Figure 1:model\_1



# سوال 2

با توجه به روابط زیر، مدل های مورد نیاز موتور را طراحی می­کنیم.



تابع تبدیل سیستم موتورها که شامل موتور خطی (موتور دورانی ترکیب شده با چرخدنده و شانه) و موتور دورانی(با افزایش گشتاور از طریق گیربکس) بر روی مدل 1 پیاده می­کنیم و شکل نهایی model\_1 به فرمت زیر خواهد بود.

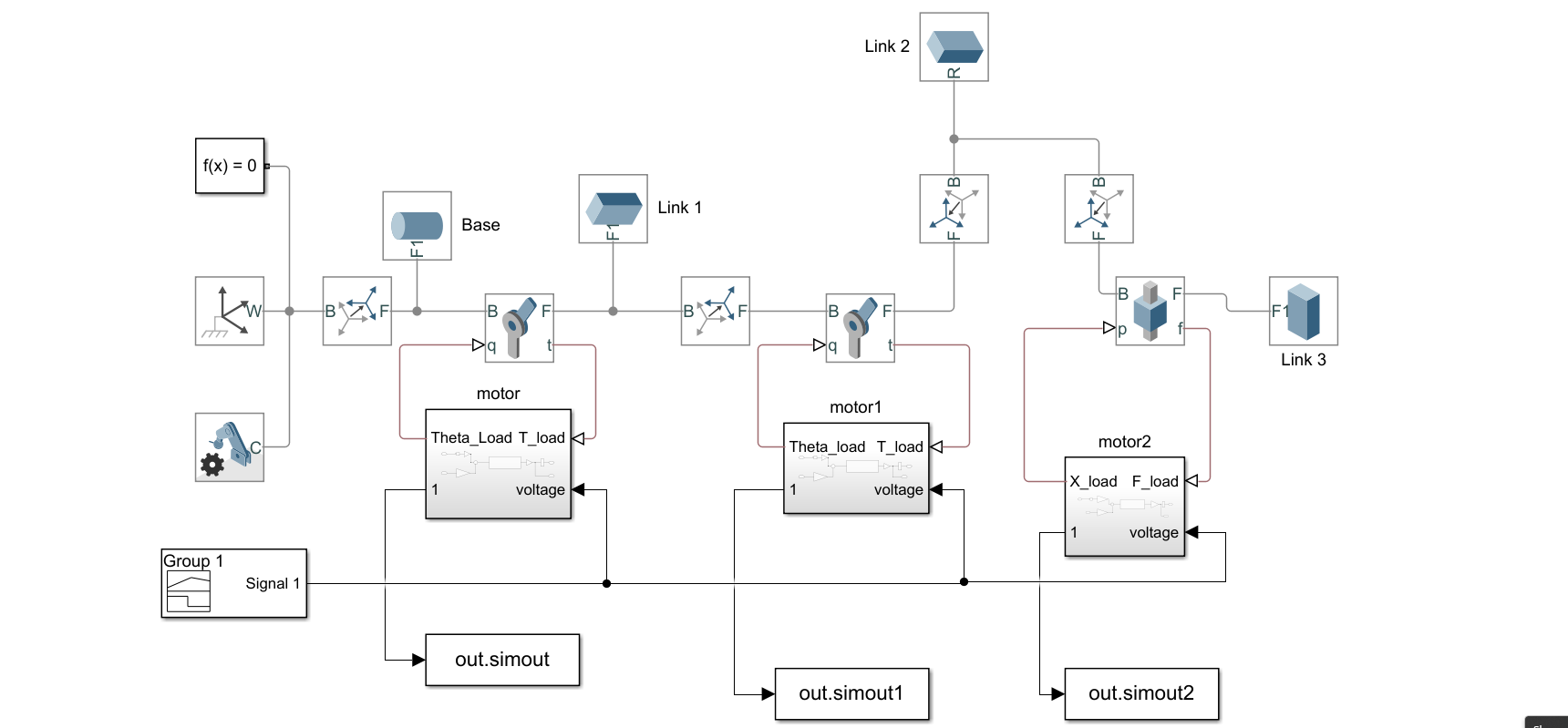


Figure 2: model\_1 Simulink diagram

بلوک های motor و motor1 که موتورهای دورانی هستند دارای سیستم داخلی زیر هستند که از معادلات ذکر شده در بالا پیروی می­کنند.

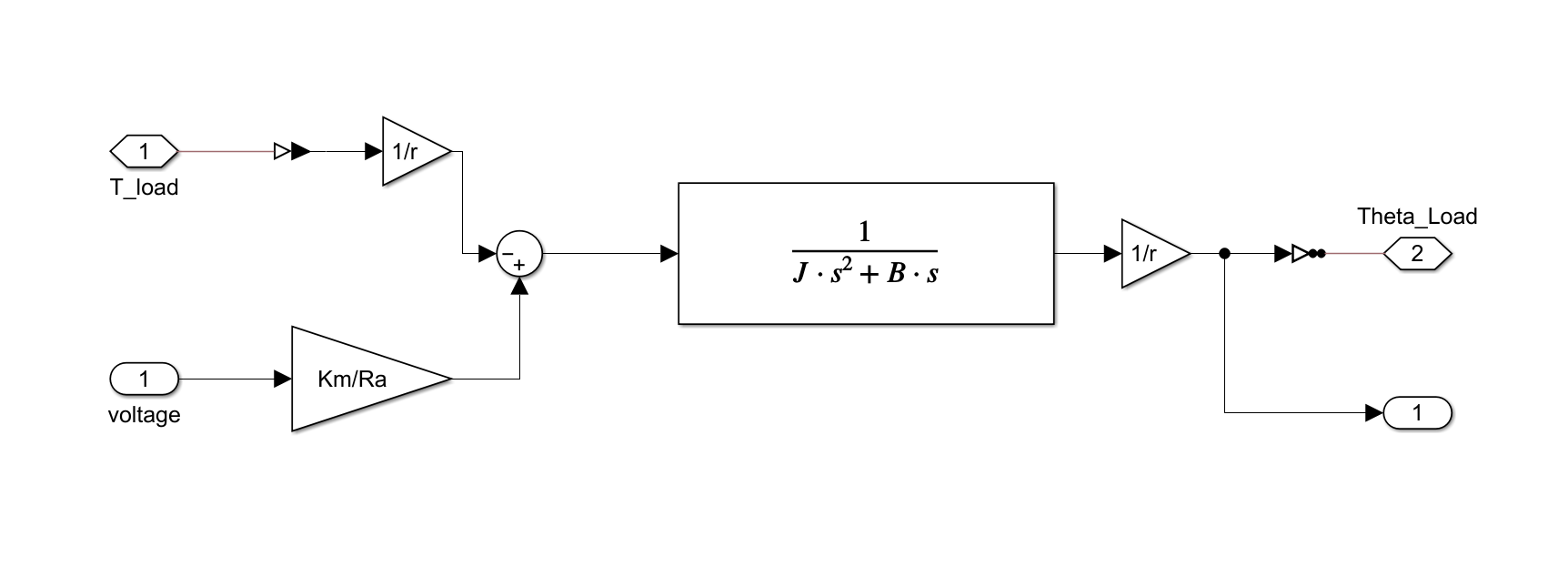


Figure 3: motor & motor1 block

همچنین برای motor2 که موتوری خطی می­باشد، به جای استفاده از نسبت تبدیل گیربکس، از نسبت تبدیل دوران به جابه­جایی چرخدنده و شانه ( همان شعاع چرخدنده که در اینجا با توجه به ابعااد ربات و توان موتور با radius مقدار 1 سانتی­متر ارزش گذاری شده است) استفاده می­کنیم.

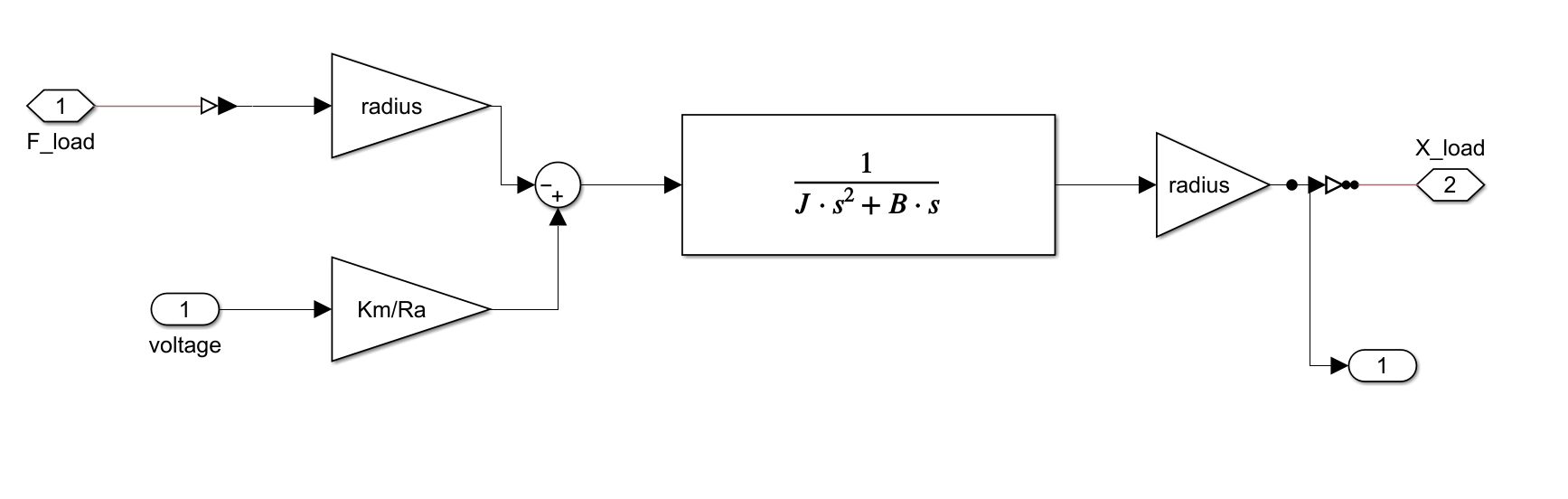


Figure 4: motor2 block

سپس از signal builder استفاده می­کنیم تا سیگنال ثابت 48V را تا زمان 3 ثانیه اعمال کنیم و پس از آن سیگنال صفر می­شود.

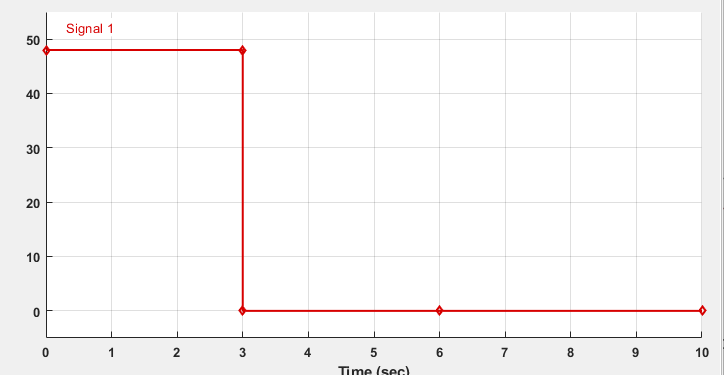


Figure 5: signal builder properties

در نهایت مدل را اجرا کرده و پس از آن با تابع “plott” زاویه هر مفصل را رسم می­کنیم. تابع plott صرفا خروجی های گرفته شده مفاصل از مدل را ترسیم می­کند.

function plott(out)

t=out.simout.Time;

theta1=out.simout.Data\*180/pi;

theta2=out.simout1.Data\*180/pi;

d=out.simout2.Data\*100;

plot(t,theta1);

xlabel("time\_{(S)}");

ylabel("\Theta\_1\_{(deg)}");

title("\Theta\_1\_{(time)}");

figure

plot(t,theta2);

xlabel("time\_{(S)}");

ylabel("\Theta\_2\_{(deg)}");

title("\Theta\_2\_{(time)}");

figure

plot(t,d);

xlabel("time\_{(S)}");

ylabel("d\_{(cm)}");

title("d\_{(time)}");

end

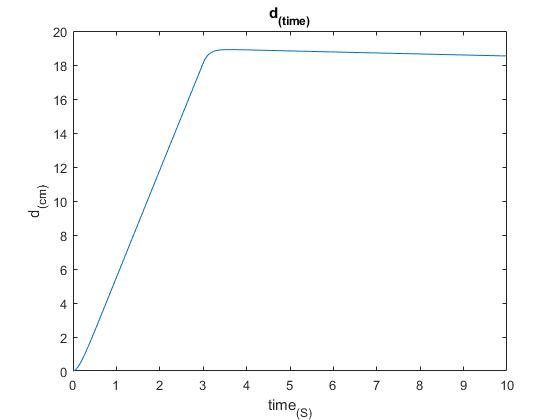
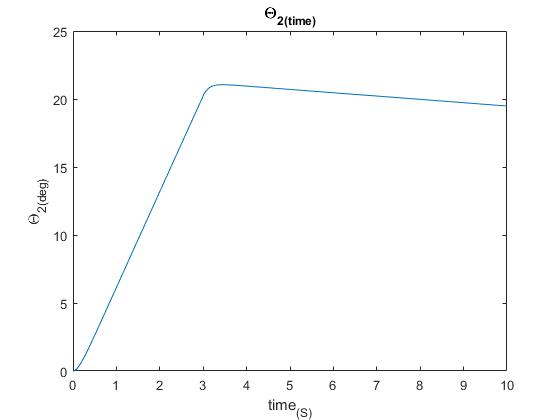
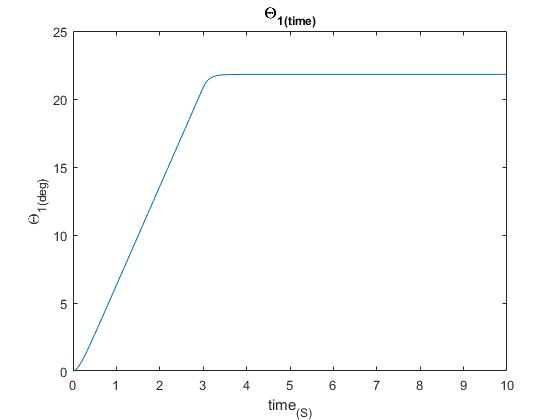


Figure 6:joints parameters

همانطور که در تصویر بالا مشاهده می­شود ، تا زمان 3 ثانیه ، هر سه موتور مقادیر مفاصل را افزایش می­دهند تا زمانی که در انتهای 3 ثانیه، سیگنال ورودی دوباره صفر شود. در این زمان موتورها قطع می­شوند و مفاصل تحت بار، به آرامی تغییر می­کنند تا پتانسیل گرانشی به مینیمم برسد.

# سوال 3

با توجه به رابطه اوورشوت برای مقدار M=0.05، پارامتر برابر با 0.690 آمده و با توجه به رابطه زمان نشست به مدت 1 ثانیه، برابر با 6.67 به دست می­آید. حال ضریب Ki را فعلا صرف در نظر می­گیریم و کنترلر PD طراحی می­کنیم. برای این هدف از رابطه زیر استفاده می­کنیم:

با توجه به این مقادیر، به Kp=6.67 و Kd=0.246 می­شود. (با توجه به داده های مسئله، J=0.15 و B=1.135 می­باشد)

با توجه به اینکه این روابط برای کنترل زاویه موتور است و نه مفصل، باید ضریب r=50 در آن ضرب شود. سپس تیونینگ را با نرم افزار PID Tuner انجام می­دهیم و مقدار Ki را هم در نظر می­گیریم و در نهایت به مقادر زیر برای هر کدام از ضرایب PID می­رسیم(اندیس 1). این فرایند برای کنترلر موتور خطی نیز تکرار می­شود (اندیس 2).

Kd1=145; Ki1=10; Kp1=1963;

Kd2=97; Ki2=10; Kp2=2377; مقادیر نهایی

حال کنترلر ها به مفاصل متصل می­کنیم ( در model\_2)

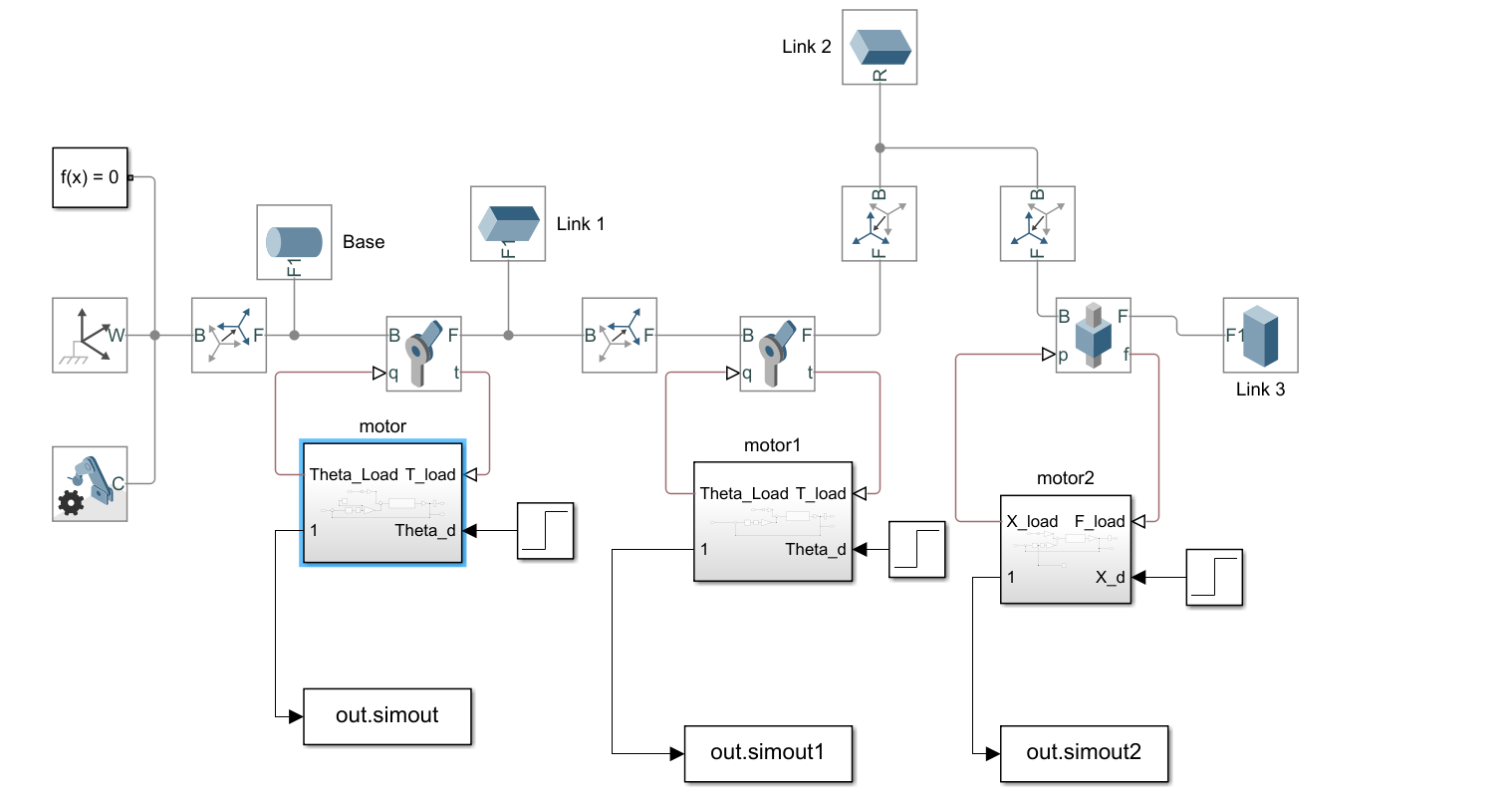
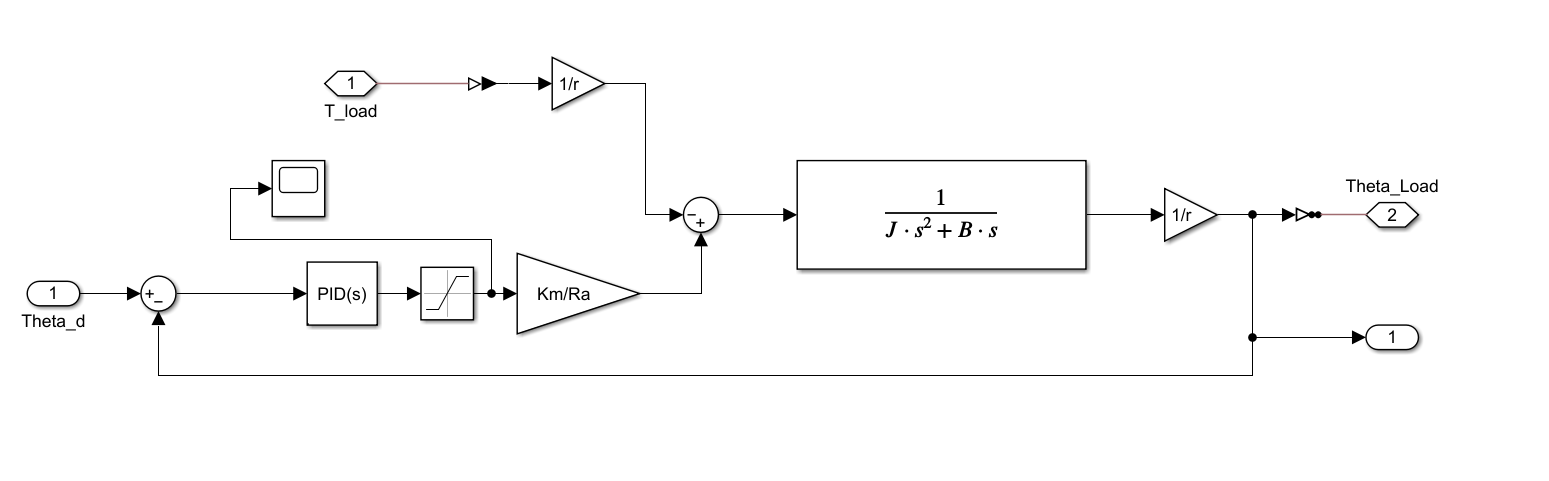


Figure 7: model\_2 diagram & motor block

ورودی های دلخواه به صورت پله، به بخش ورودی desired بلوک موتورها داده شده است. سپس کنترلر ولتاژ را طوری تنظیم می­کند تا مفاصل ربات به موقعیت های خواسته شده برسند.

در نهایت کد این بخش را اجرا کرده و بعد از آن مدل را ران می­کنیم. در نهایت با دستور plott ، نمودارهای مفاصل را رسم می­کنیم.

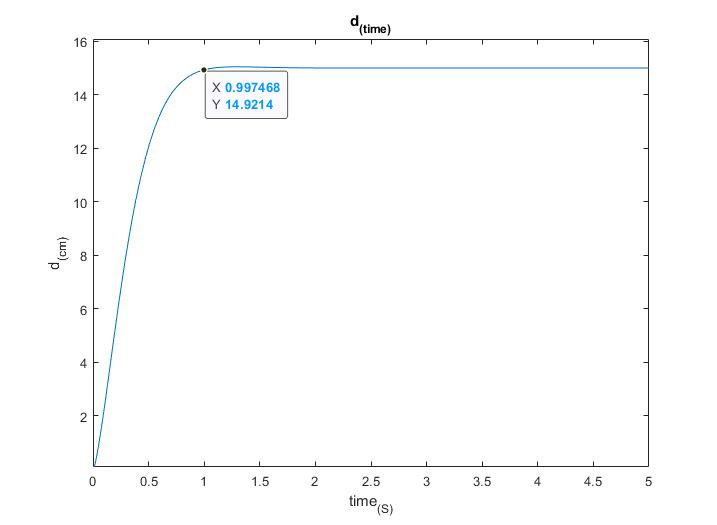
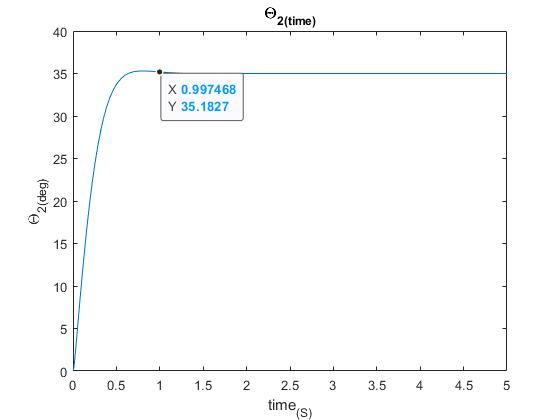
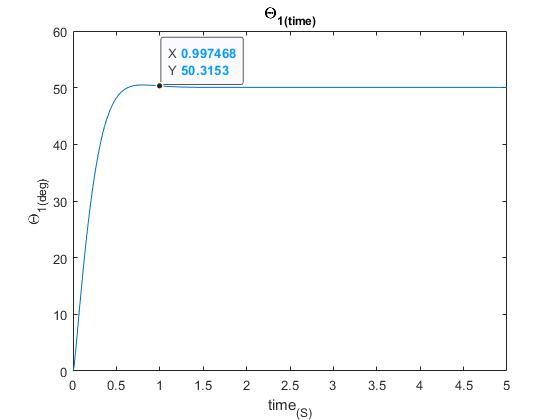
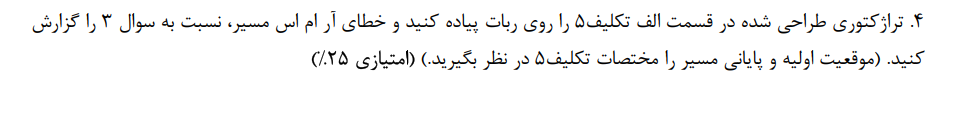
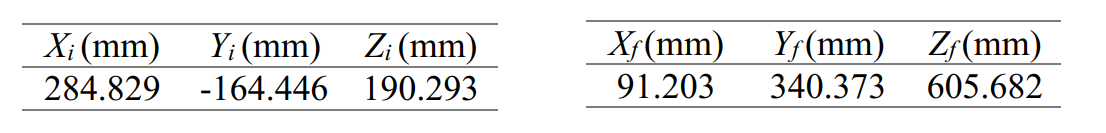


Figure 8: Joints diagram & settling times

همانطور که در نمودارها مشاهده می­شود، زمان نشست نزدیک به 1 ثانیه بوده و اووشوت کمتر از 5% است.



# سوال 4 (امتیازی)

برای این قسمت باید طی یک مسیر نرم با درجه جمله 3 نسبت به زمان، تراژکتوری خواسته شده بین دو نقطه زیر را ایجاد کنیم.

برای ایجاد این مسیر از کد تمرین 5 استفاده می­کنیم تا ورودی desired هر مفصل بر حسب زمان به دست آید.

%% Problem 4

% First run this

Kd1=145; Ki1=10; Kp1=1963;

Kd2=97; Ki2=10; Kp2=2377;

L1=310; %mm

L2=300; %mm

position=[284.829 -164.446 190.293 ;

91.203 340.373 605.682 ];

parameters=zeros(3,2);

a=zeros(3,4);

s=1; % in seconds

for k=1:2

x=position(k,1);

y=position(k,2);

z=position(k,3);

t1=atan2(y,x);

if abs(x)<1e-3

t2=atan( (z-L1)\*sin(t1)/y );

d=y/(sin(t1)\*cos(t2))-L2;

else

t2=atan( (z-L1)\*cos(t1)/x );

d=x/(cos(t1)\*cos(t2))-L2;

end

parameters(:,k)=[t1;t2;d];

end

%Now we calculate trajectory coefficient

a(:,1)=parameters(:,1);

%a(:,2) remain zero

a(:,4)=(-parameters(:,2)+a(:,1))/(0.5\*s^3);

a(:,3)= -1.5\*a(:,4)\*s;

%Now we generate trajactory inputs

t=0:0.01:1;

f=@(x) a(:,1)+a(:,2)\*x+a(:,3)\*x.^2+a(:,4)\*x.^3;

output=f(t); % outputs are in mm and rad

joints\_traj=[t',output(1,:)',t',output(2,:)',t',output(3,:)'];

سپس این ورودی ها را، در model\_3 در بلوک Ideal Trajectory به مفاصل می­دهیم تا مسیر ایده­آل را برای محاسبه خطای rms داشته باشیم.

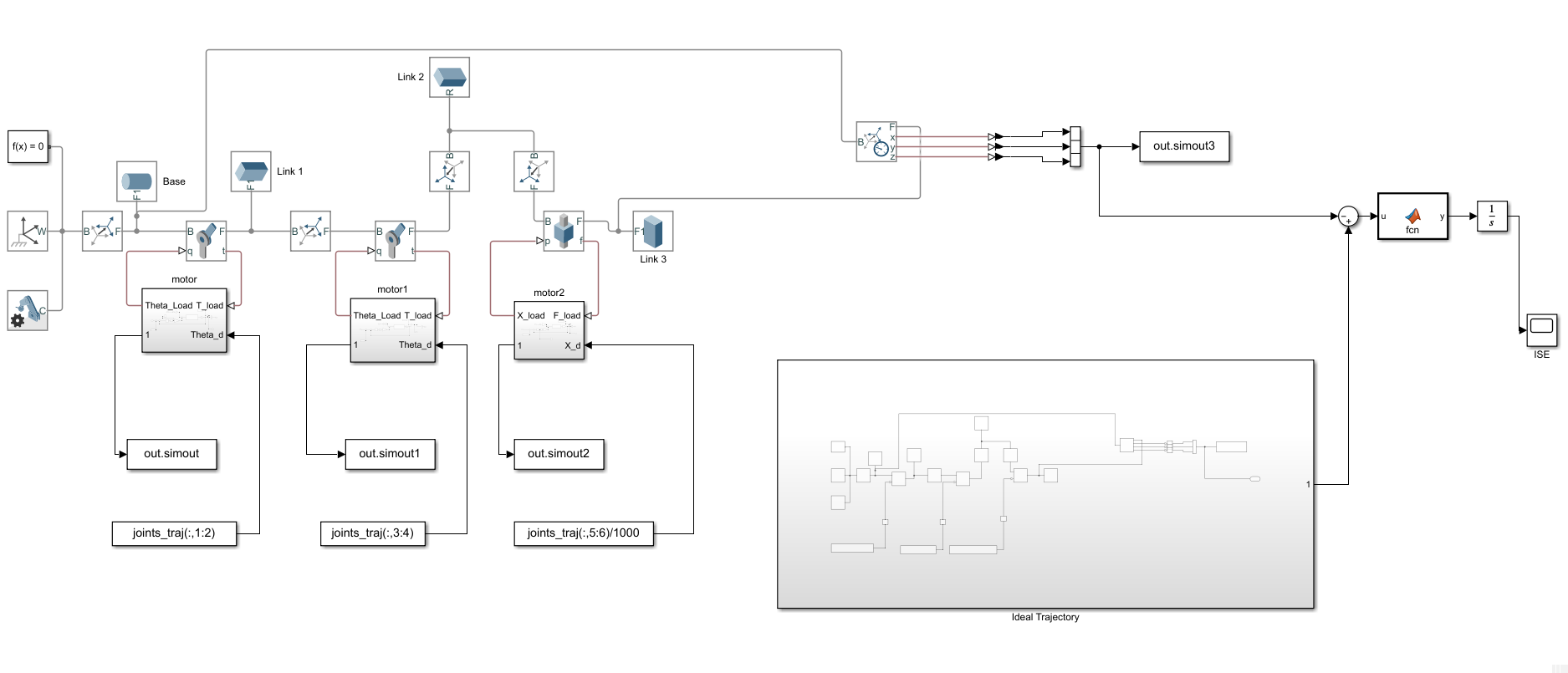


Figure 9:model\_3

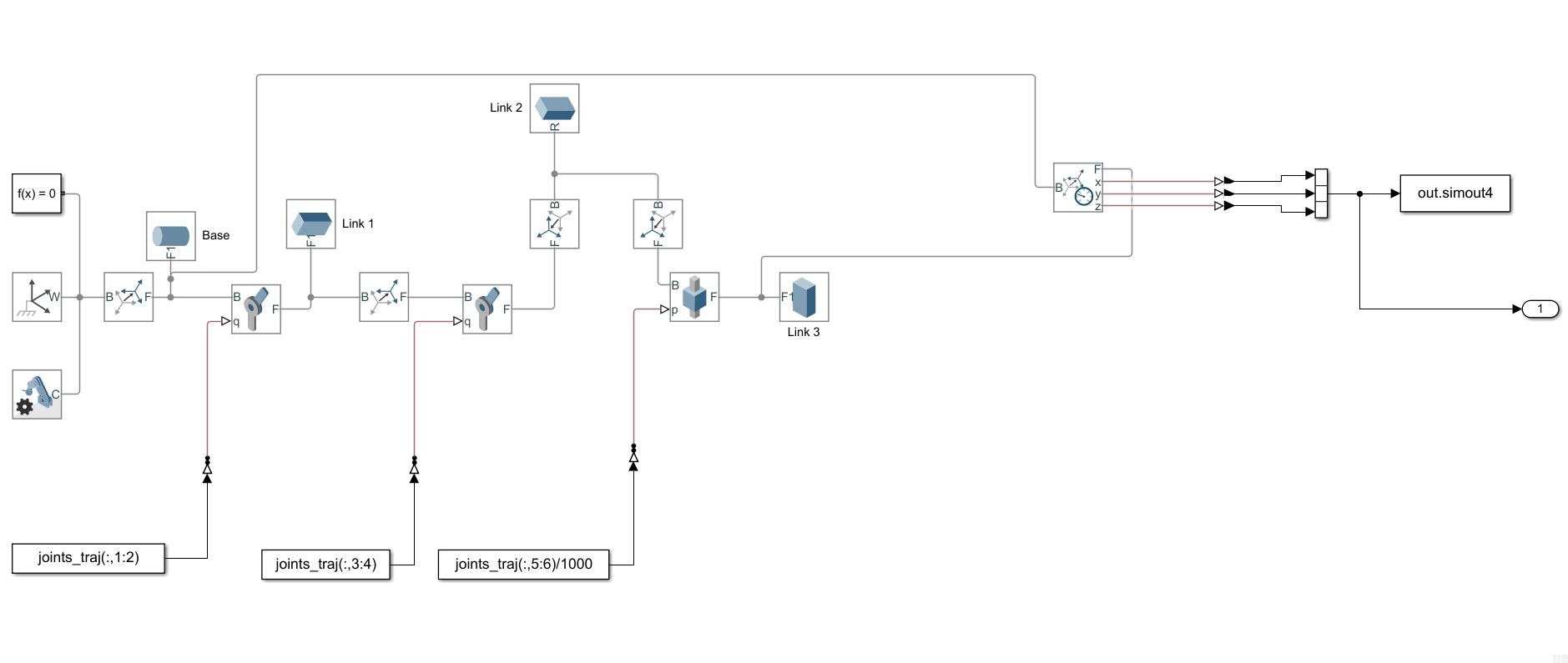
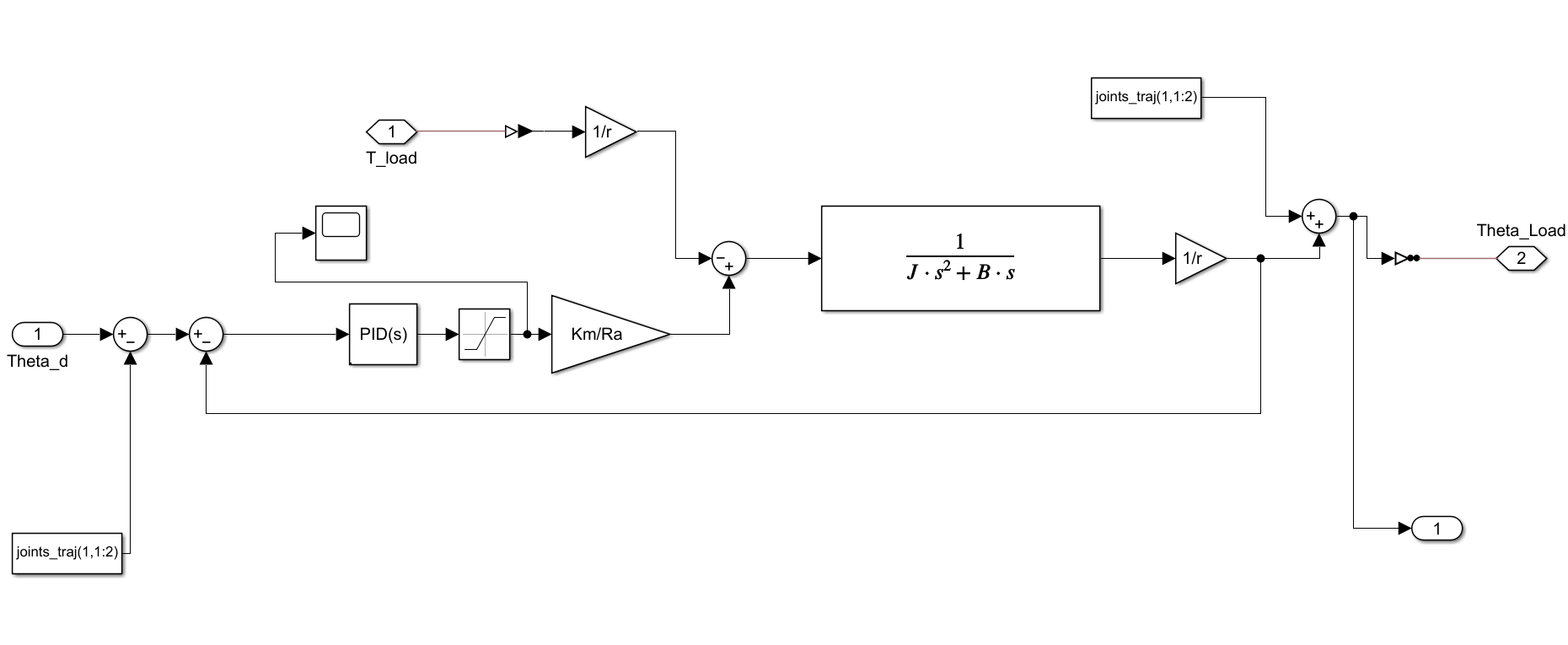


Figure 10: Ideal Trajectory block

برای اجرا کردن model\_3 از همان کنترلرهایی استفاده می­کنیم که در سوال 3 استفاده کردیم اما باید دقت کرد که موقعیت اولیه مفاصل صفر نخواهد بود، درحالی که کنترلر با فرض ورودی صفر طراحی می­شود. برای رفع این مشکل، در بلوک موتورهای کنترل شده به فرم زیر تغییراتی ایجاد می­کنیم.



پارامتر اولیه

Figure 11 : model\_3 motor block

در ابتدا فرض می­کنیم که موتور واقعا در موقعیت گفته شده طبق trajectory قرار دارد. در نتیجه زمانی که موتور، ورودی دریافت می­کند، از آن پارامتر موقعیت اولیه را کم می­کنیم. به این شکل اگر کاربر بخواهد ربات در موقعیت خود بماند، هیچ خطایی مشاهده نمی­شود. اما باید در انتها مقدار اولیه به خروجی سیستم کنترلری اضافه شود تا موقعیت واقعی به عنوان ورودی به مفصل داده شود. در نهایت نیز خروجی بلوک تراژکتوری ایده­آل و سیستم کنترل شده از یکدیگر کم می­شوند و اندازه بردار اختلاف بر حسب زمان، انتگرال گرفته می­شود.(تابع matlab تعریف شده در Simulink کار تابع norm() به توان 2 را انجام می­دهد)

در نتیجه نمودار انتگرال مجذور خطا، به شکل زیر خواهد بود:

می­توان مشاهده کرد که اندازه انتگرال حداکثر تا 0.036 پیش خواهد رفت. در نتیجه خطای rms برابر می شود با :

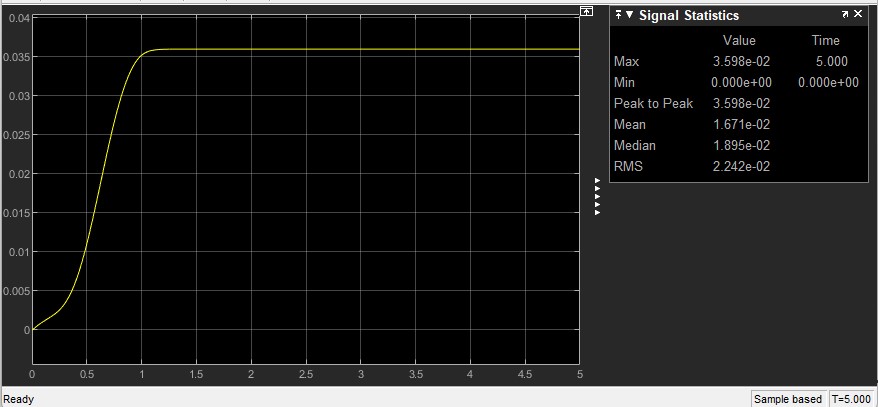
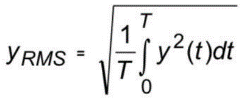
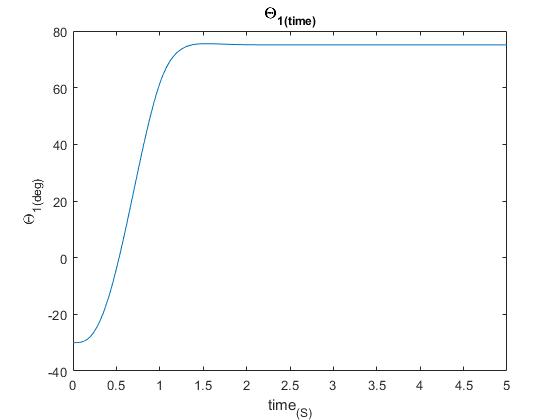


Figure 12: ISE diagram



چون زمان نشست T برابر با 1.5 ثانیه است، rms برابر با 0.155 متر خواهد بود. این به این معناست که کنترلر به طور متوسط در بازه نشست می­تواند به صورت محدودی تراژکتوری را دنبال کند اما با تاخیر در مرتبه زمان نشست، به آن می­رسد. در نهایت با اجرا plottt که مانند plott عمل می­کند ولی خطای تراژکتوری و مقدار واقعی موقعیت اندفکتور را نیز می­دهد، به نمودار های زیر می­رسیم.



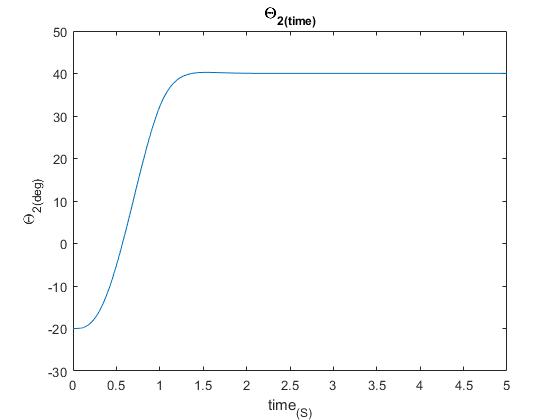
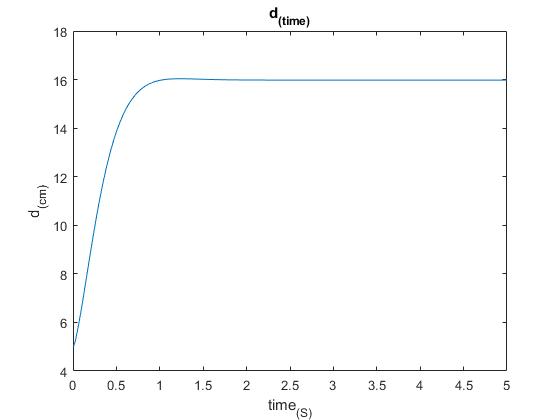


Figure 13: Joints parameters in model\_3

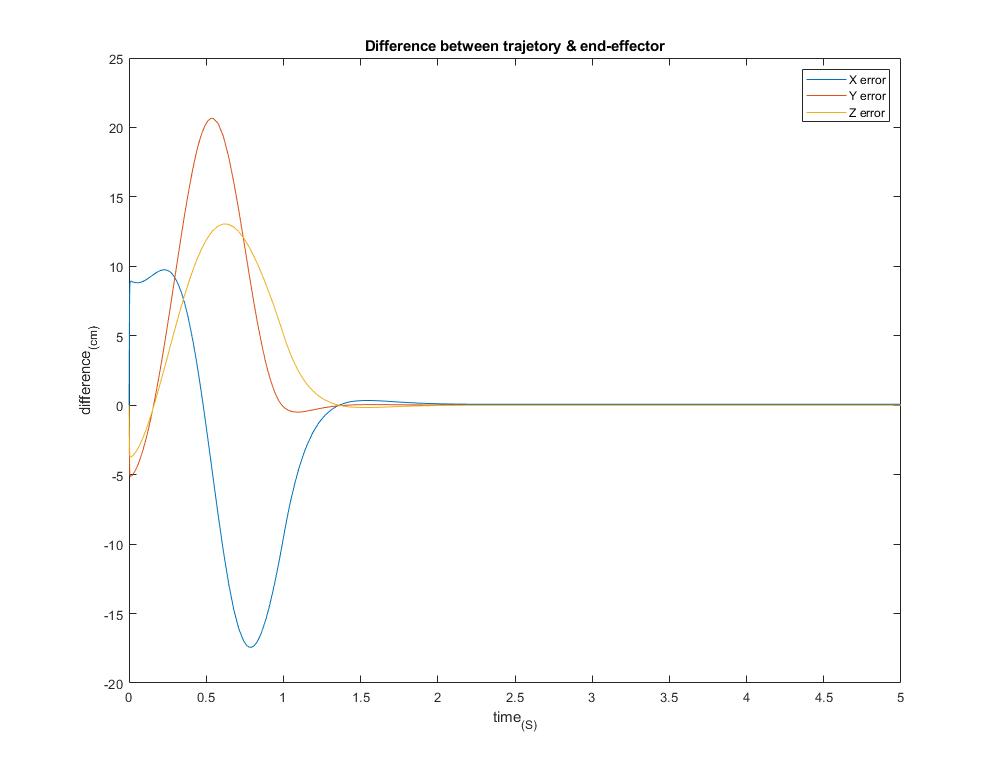
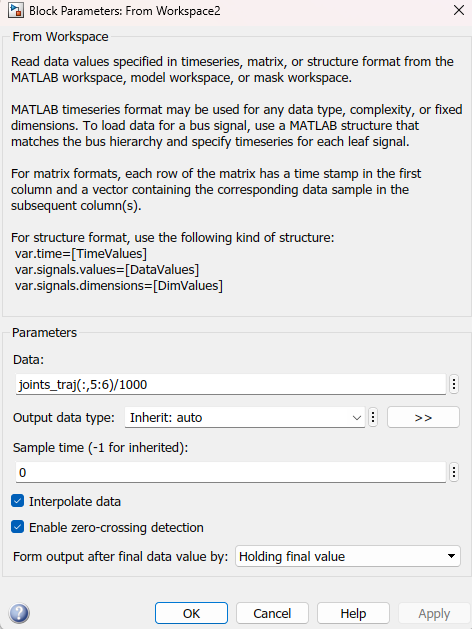


Figure 14: Trajectory & end-effector difference

همچنین باید توجه کرد که پس از 1 ثانیه، موقعیت desired همان موقعیت نهایی بماند. برای این کار، در بلوک from workspace باید این تنظیم را انجام دهیم.