Table of Contents


```
function [] = slider_plot()
% Plot different plots according to slider location.
% Junk data for comparison of adjusted curves in this example

S.fh = figure('units','normalized',...
    'Position', [0.515 0.025 0.415 0.87],... %%%%
              'name','slider_plot');


% Define inital parameter values for curve
S.a = 0;
S.b = 0;
S.c = 0;
S.d = 0;
S.e = 0;
S.f = 0;
S.g = 0;

update(S);

% 1st Slider:
S.aSlider = uicontrol('style','slider',...
               'unit','normalized',...
               'position',[0.2 0.3 0.3 0.01],...
               'min',-180,'max',180,'value', S.a,...
               'sliderstep',[0.003 0.03],...
               'callback', {@SliderCB, 'a'});
% Add a text uicontrol to label the slider.
txta = uicontrol('Style','text',...
     'unit','normalized',...
               'position',[0.2 0.31 0.3 0.02],...
     'String','Joint angle j1');

% 2nd Slider:
S.bSlider = uicontrol('style','slide',...
               'unit','normalized',...
               'position',[0.2 0.35 0.3 0.01],...
               'min',-180,'max',180,'value', S.b,...
               'sliderstep',[0.003 0.03],...
               'callback', {@SliderCB, 'b'});
% Add a text uicontrol to label the slider.
txtb = uicontrol('Style','text',...
```

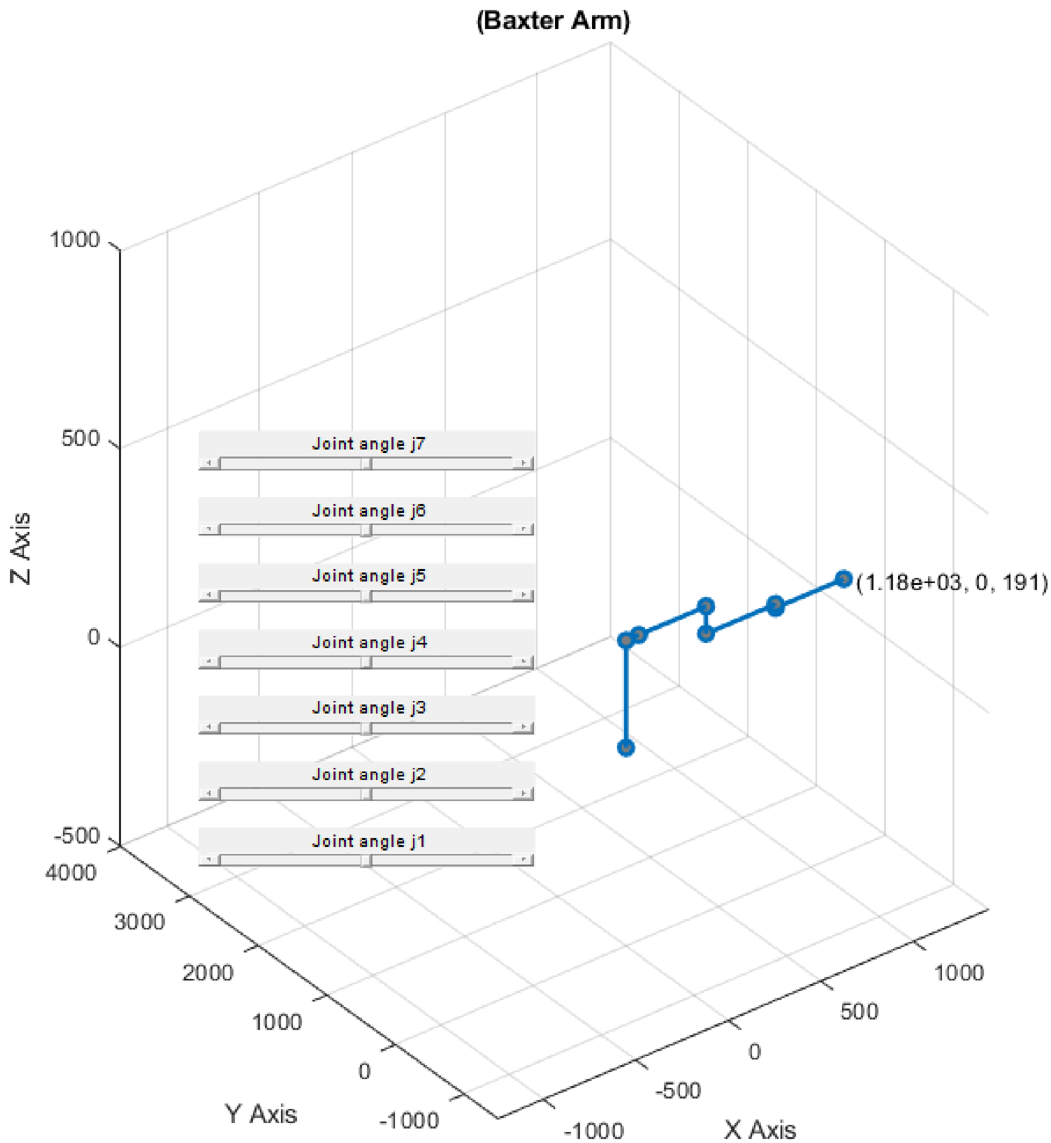
```
     'unit','normalized',...
               'position',[0.2 0.36 0.3 0.02],...
     'String','Joint angle j2');

% 3rd Slider:
S.cSlider = uicontrol('style','slide',...
               'unit','normalized',...
               'position',[0.2 0.4 0.3 0.01],...
               'min',-180,'max',180,'value', S.c,...
               'sliderstep',[0.003 0.03],...
               'callback', {@SliderCB, 'c'});
% Add a text uicontrol to label the slider.
txtc = uicontrol('Style','text',...
     'unit','normalized',...
               'position',[0.2 0.41 0.3 0.02],...
     'String','Joint angle j3');

% 4th Slider:
S.dSlider = uicontrol('style','slide',...
               'unit','normalized',...
               'position',[0.2 0.45 0.3 0.01],...
               'min',-180,'max',180,'value', S.d,...
               'sliderstep',[0.003 0.03],...
               'callback', {@SliderCB, 'd'});
% Add a text uicontrol to label the slider.
txtd = uicontrol('Style','text',...
     'unit','normalized',...
               'position',[0.2 0.46 0.3 0.02],...
     'String','Joint angle j4');

 % 5th Slider:
S.eSlider = uicontrol('style','slide',...
               'unit','normalized',...
               'position',[0.2 0.5 0.3 0.01],...
               'min',-180,'max',180,'value', S.e,...
               'sliderstep',[0.003 0.03],...
               'callback', {@SliderCB, 'e'});
% Add a text uicontrol to label the slider.
txtd = uicontrol('Style','text',...
     'unit','normalized',...
               'position',[0.2 0.51 0.3 0.02],...
     'String','Joint angle j5');

  % 6th Slider:
S.fSlider = uicontrol('style','slide',...
               'unit','normalized',...
               'position',[0.2 0.55 0.3 0.01],...
               'min',-180,'max',180,'value', S.f,...
               'sliderstep',[0.003 0.03],...
               'callback', {@SliderCB, 'f'});
% Add a text uicontrol to label the slider.
txtd = uicontrol('Style','text',...
     'unit','normalized',...
               'position',[0.2 0.56 0.3 0.02],...
```

```
     'String','Joint angle j6');

   % 7th Slider:
S.eSlider = uicontrol('style','slide',...
               'unit','normalized',...
               'position',[0.2 0.6 0.3 0.01],...
               'min',-180,'max',180,'value', S.g,...
               'sliderstep',[0.003 0.03],...
               'callback', {@SliderCB, 'g'});
% Add a text uicontrol to label the slider.
txtd = uicontrol('Style','text',...
     'unit','normalized',...
               'position',[0.2 0.61 0.3 0.02],...
     'String','Joint angle j7');


guidata(S.fh, S);  % Store S structure in the figure
end

% Callback for all sliders defined above
function SliderCB(aSlider, EventData, Param)
S = guidata(aSlider);  % Get S structure from the figure
S.(Param) = get(aSlider, 'Value');  % Any of the 'a', 'b', etc.
 defined
update(S);  % Update the plot values
guidata(aSlider, S);  % Store modified S in figure
end
% Plot update function, creates new y-vector for plot and replaces the
 plot
% S.p2 with new y-vector
function update(S)
h1=FKdraw(S.a,S.b,S.c,S.d,S.e,S.f,S.g)
end
```

```
h1 =

  Columns 1 through 6

         1           0           0           0           1
  0
         0           1           0           0           0
  0
         0           0           1         270           0
 -1
         0           0           0           1           0
  0

  Columns 7 through 12

         0          69           0           0           1
 69
         1           0           0           1           0
  0
```

```
         0         270          -1           0           0
270
         0           1           0           0           0
 1

 Columns 13 through 18

         0           0           1         433           0
-1
         0           1           0           0           0
 0
        -1           0           0         270          -1
 0
         0           0           0           1           0
 0

 Columns 19 through 24

         0         433           0           0           1
433
         1           0           0           1           0
 0
         0         201          -1           0           0
201
         0           1           0           0           0
 1

 Columns 25 through 30

         0           0           1         808           0
-1
         0           1           0           0           0
 0
        -1           0           0         201          -1
 0
         0           0           0           1           0
 0

 Columns 31 through 36

         0         808           0           0           1
808
         1           0           0           1           0
 0
         0         191          -1           0           0
191
         0           1           0           0           0
 1

 Columns 37 through 40

         0           0           1        1176
         0           1           0           0
        -1           0           0         191
```

```
0            0            0            1
```



Fk Draw Function (Plots the Links in 3D space)

```
function [ FK ] = FKdraw( j1,j2,j3,j4,j5,j6,j7)

%initial parameter
%[j0 j1 j2 j3;d0 d1 d2 d3;a0 a1 a2 a3;t0 t1 t2 t3]
```

```
% D-H Parameters
a1 = 69; % length of first arm
a2 = 0; % length of second arm
a3 = 69; % length of third arm
a4 = 0; % length of fourth arm
a5 = 10; % length of fifth arm
a6 = 0; % length of sixth arm
a7 = 0; % length of seventh arm
d1 = 270; % offset of first arm
d2 = 0; % offset of second arm
d3 = 364; % offset of third arm
d4 = 0; % offset of fourth arm
d5 = 375; % offset of fifth arm
d6 = 0; % offset of sixth arm
d7 = 368; % offset of seventh arm

j=[j1 0 j2+90 j3 0 j4 j5 0 j6 j7;d1 0 0 d3 0 0 d5 0 0 d7;0 a1 0 0 a3 0
 0 a5 0 0;0 -90 90 0 -90 90 0 -90 90 0];

FK=DHkine(j);
Q=XYZkine(FK);

h=plot3(Q(1,:),Q(2,:),Q(3,:),'-
o','LineWidth',2,'MarkerSize',6,'MarkerFaceColor',[0.5,0.5,0.5]);



grid on;clc
text(Q(1,11),Q(2,11),Q(3,11),['  (', num2str(Q(1,11),3), ', ',
 num2str(Q(2,11),3),', ', num2str(Q(3,11),3), ')']);
title('(Baxter Arm)');
xlabel('X Axis');
ylabel('Y Axis');
zlabel('Z Axis');
axis([-1250 1400 -1500 4000 -500 1000]);
h = rotate3d;
h.Enable = 'on';
h.ActionPostCallback = @mypostcallback;
assignin('base','FK',FK);

end

function mypostcallback(obj,evd)
%disp('A rotation is about to occur.');
ax_properties = get(gca);
assignin('base','pov',ax_properties.View);
end

%use evalin('base',a) to get variable a from workspace
%assignin('base','a_rms',a_rms) to write variable a_rms to workspace
```

DHkine Function (Computes the Transformation Matrices)

```
function [ FK ] = DHkine(j)
%collum 1=joint angle, 2=joint offset, 3=link lenght, 4=twist angle
T01=DHmatrix(j(1,1),j(2,1),j(3,1),j(4,1));
T12=DHmatrix(j(1,2),j(2,2),j(3,2),j(4,2));
T23=DHmatrix(j(1,3),j(2,3),j(3,3),j(4,3));
T34=DHmatrix(j(1,4),j(2,4),j(3,4),j(4,4));
T45=DHmatrix(j(1,5),j(2,5),j(3,5),j(4,5));
T56=DHmatrix(j(1,6),j(2,6),j(3,6),j(4,6));
T67=DHmatrix(j(1,7),j(2,7),j(3,7),j(4,7));
T78=DHmatrix(j(1,8),j(2,8),j(3,8),j(4,8));
T89=DHmatrix(j(1,9),j(2,9),j(3,9),j(4,9));
T910=DHmatrix(j(1,10),j(2,10),j(3,10),j(4,10));


T02=T01*T12;
T03=T02*T23;
T04=T03*T34;
T05=T04*T45;
T06=T05*T56;
T07=T06*T67;
T08=T07*T78;
T09=T08*T89;
T010=T09*T910;


FK=[T01 T02 T03 T04 T05 T06 T07 T08 T09 T010];
end
```

DH matrix Function (Contains the General DH Matrix)

```
function [Mdh] = DHmatrix(theta,d,a,alpha)
%DHMATRIX Summary of this function goes here
%   input DH parameter
Mdh=[cosd(theta) -sind(theta)*cosd(alpha) sind(theta)*sind(alpha)
 a*cosd(theta);
     sind(theta) cosd(theta)*cosd(alpha)  -cosd(theta)*sind(alpha)
 a*sind(theta);
     0,sind(alpha),cosd(alpha),d;
     0,0,0,1];
end
```

XYZkine (To compute the Q matrix to plot)

```
function [Q] = XYZkine(FK)
%DRAWKINE Summary of this function goes here

Q1=[0 FK(1,4) FK(1,8) FK(1,12) FK(1,16) FK(1,20) FK(1,24) FK(1,28)
 FK(1,32) FK(1,36) FK(1,40)];
Q2=[0 FK(2,4) FK(2,8) FK(2,12) FK(2,16) FK(2,20) FK(2,24) FK(2,28)
 FK(2,32) FK(2,36) FK(2,40)];
Q3=[0 FK(3,4) FK(3,8) FK(3,12) FK(3,16) FK(3,20) FK(3,24) FK(3,28)
 FK(3,32) FK(3,36) FK(3,40)];
Q=[Q1;Q2;Q3];
end
```

Published with MATLAB® R2019b