Example code for Lab 3

Henrik Andreasson, henrik.andreasson@oru.se

October 8, 2015

This document contains a set of Matlab script files (.m - files) which gives parts of the solutions in Lab 3.

1 Example code for Task 3

File: example_lab3task3.m

```
% Lab 3 task 3.

close all;
clear all;

S(1).x = 0;
S(1).y = 0;
S(1).z = 0;
S(1).alpha = 0;
S(1).beta = pi/6;
S(1).gamma = 0;

S(2).x = 2;
S(2).y = 0;
S(2).z = 0;
S(2).alpha = 0;
S(2).beta = 0;
S(2).gamma = 0;

% Plot the frames
plot_S(S)

% Function that rotates round the x-axis.
S(1) = rotx_S(S(1), pi/8);
plot_S(S)

% Rotate around S(2)
S(2) = rotx_S(S(2), pi/8);
plot_S(S)
```

```
% Translate along z-axis in S(1)
S(1) = translz_S(S(1), 0.2);
plot_S(S)

% Translate along z-axis in S(2)
S(2) = translz_S(S(2), 0.2);
plot_S(S)



% New plotting window
figure(2)
% Use the keys

stop=0; inc=0.01;
while ¬stop
    clf;
    plot_S(S)
    waitforbuttonpress;
    if strcmp(get(gcf,'currentcharacter'),'a');
        S(1) = rotx_S(S(1), inc);
    elseif strcmp(get(gcf,'currentcharacter'),'z');
        S(1) = rotx_S(S(1), -inc);
    elseif strcmp(get(gcf,'currentcharacter'),'s');
        S(2) = rotx_S(S(2), inc);
    elseif strcmp(get(gcf,'currentcharacter'),'x');
        S(2) = rotx_S(S(2), -inc);
    elseif strcmp(get(gcf,'currentcharacter'),'d');
        S(1) = translz_S(S(1), inc);
    elseif strcmp(get(gcf,'currentcharacter'),'c');
        S(1) = translz_S(S(1), -inc);
    elseif strcmp(get(gcf,'currentcharacter'),'f');
        S(2) = translz_S(S(2), inc);
    elseif strcmp(get(gcf,'currentcharacter'),'v');
        S(2) = translz_S(S(2), -inc);

    elseif strcmp(get(gcf,'currentcharacter'),'q');
    stop=1; disp('quit');
    end

end
```

1.1 Functions used for drawing

File: `S2T.m`

```
function T = S2T(S)
%
% forms a homogeneous matrix T given
%
% S - struct as given in the lab description
%

R = rx(S.alpha)*ry(S.beta)*rz(S.gamma);
t = [S.x S.y S.z]';
```

```
T = [          R, t;
    zeros(1,3),   1];

%%%EOF form_T
% ...
```

File: `plot_T.m`

```
function plot_T(T)
%
% plot the corresponding frame to a homogeneous matrix T
%
% T - homogeneous matrix
%
p = T(1:3,4);
r1 = T(1:3,1);
r2 = T(1:3,2);
r3 = T(1:3,3);
plot3([p(1) p(1)+r1(1)], [p(2) p(2)+r1(2)], [p(3) p(3)+r1(3)], 'r')
plot3([p(1) p(1)+r2(1)], [p(2) p(2)+r2(2)], [p(3) p(3)+r2(3)], 'g')
plot3([p(1) p(1)+r3(1)], [p(2) p(2)+r3(2)], [p(3) p(3)+r3(3)], 'b')
%%%EOF plot_T
% ...
```

File: `plot_S.m`

```
function plot_S(S)
%
% plot the struct S from the lab description
%
% S - struct as given in the lab description
% for simplicity this is hard coded to be of length 2.
%
% Compute the T matrices
T1_local = S2T(S(1));
T2_local = S2T(S(2));

T1_global = T1_local;
T2_global = T1_global*T2_local;

% Plot the frames
hold on
plot_T(T1_global)
plot_T(T2_global)
%%%EOF plot_S
% ...
```

1.2 Functions to perform rotations and translations

File: `rotx_S.m`

```
function S=rotx_S(S, angle)
%
% update S with rotation angle around it's local X coordinate
%
% S - struct as given in the lab description
% angle - angle to be rotated
%

R = rx(S.alpha)*ry(S.beta)*rz(S.gamma)*rx(angle);
rpy = R2rpy(R);
S.alpha = rpy(1);
S.beta = rpy(2);
S.gamma = rpy(3);

% This will not guarantee that we rotate around the x-axis only ...
    since the
% rotation could be dependent on S.beta and S.gamma.
%S.alpha = S.alpha + angle;

%%%EOF plot_S
% ...
```

File: `translz_S.m`

```
function S=translz_S(S, dist)
%
% update S with a translation along it's local Z coordinate
%
% S - struct as given in the lab description
% dist - distance to be translated
%

S.z = S.z + dist;
%%%EOF translz_S
% ...
```

2 Example code for Task 4

File: `example_lab3task4.m`

```
%
% Example1 (Lab_03)
%
```

```
clear;clc;cla

% constant parameters (system definition)
% ...
pos(:,1) = [1; 1; 0]; % position of frame 1 in frame 0
pos(:,2) = [1; 0; 0]; % position of frame 2 in frame 1
pos(:,3) = [1; 0; 0]; % position of frame 3 in frame 2
%pos(:,4) = [0; 1; 0]; % position of frame 4 in frame 3
% ...

% frame i is rotated w.r.t. frame i-1 using x->y->z Euler angles ... (current axis)
rot(:,1) = [0;0;0]; % frame 1 in world frame
rot(:,2) = [0;pi/2;0]; % frame 2 in frame 1
rot(:,3) = [0;0;0]; % frame 3 in frame 2
%rot(:,4) = [0;0;0]; % frame 4 in frame 3
% ...

e.pos = [1;0;0]; % position in last frame
e.rot = [0;0;0]; % x->y->z Euler angles (current frame) w.r.t ... last frame

n = size(pos,2);

% just testing
%pos = randn(3,n);
%rot = randn(3,n);
%e.pos = randn(3,1);
%e.rot = randn(3,1);

% variable parameters: q(i) is the angle around the local z-axis ... of frame i
% ...
q = zeros(n,1);
q(2) = pi/4;

% visualization
% ...
[pe, Re] = plot_chain(pos,rot,e,q); % plots one configuration

%plot 3 configuration
if 0
    for i = 1:3
        %plot_chain(pos,rot,e,q+[0.2*i;0;0]);
        %plot_chain(pos,rot,e,q+[0;0.2*i;0]);
        plot_chain(pos,rot,e,q+[0;0;0.2*i]);
    end
end

% the same system using the bMSd toolbox
% ...
if 0
    figure
    SP = model_bMSd(pos,rot,e);
    SV = System_Variables(SP);
    SV.q = q;

```

```
    % positions of CoM of links
    SV = calc_pos(SP,SV);

    % Position of the joints in the world frame
    pJ = fk_j(SP,SV,1:SP.n);

    % Position of the end-effector in the world frame
    [pe1,Re1] = fk_e(SP,SV,SP.bN,SP.bP,SP.bR);

    % visualize
    Draw_System(SP, SV, SP.bN, SP.bP,1:SP.n);

    disp('compare')
    pe-pe1
    Re-Re1
end

grid on; axis equal
%%%EOF
```

2.1 Functions used for drawing

File: `plot_chain.m`

```
function [pe, Re] = plot_chain(pos,rot,e,q)
%
% plots a chain of frames defined by
%
% pos(:,i)  - position of frame i as seen from frame i-1
%             (frame 0 is the world frame)
%
% rot(:,i)  - frame i is rotated w.r.t. frame i-1 using
%             x->y->z Euler angles (current axis)
%
% e.pos     - position of the end-effector in last frame
% e.pos     - x->y->z Euler angles (current frame) w.r.t last frame
%
% q(i)      - angles around the local z-axis
%

Tp = eye(4); % world frame (the parent frame of the first frame)

hold on;

% plot the world frame
plot_line([0;0;0],Tp(1:3,1),'b',2);
plot_line([0;0;0],Tp(1:3,2),'g',2);
plot_line([0;0;0],Tp(1:3,3),'r',2);

% Plot all frames
for i = 1:length(q)
    Tc = Tp*form_T(pos(:,i),rot(:,i),q(i)); % forward geometric ...
        model
    plot_T(Tp,Tc);
    Tp = Tc;
end
```

```
% handle the end-effector
Tc = Tp*form_T(e.pos,e.rot,0);
plot_T(Tp,Tc);
pe = Tc(1:3,4);
Re = Tc(1:3,1:3);
plot3(pe(1),pe(2),pe(3),'ro','MarkerFaceColor','r','markerSize',6)

%%%EOF plot_chain
% ...
```

```
function T = form_T(pos,rot,q)
%
% forms a homogeneous matrix T given
%
% pos - 3D position
% rot - Euler angles x->y->z (current axis)
% q   - joint angle around the local z-axis
%

R = rx(rot(1))*ry(rot(2))*rz(rot(3)+q);
T = [          R, pos;
    zeros(1,3),   1];

%%%EOF form_T
% ...
```

```
function plot_T(Tp,Tc)
%
% Tp and Tc are two homogeneous matrices defining the posture of two
% consecutive frames with respect to the world frame.
%
% This function plots the frame associated with the homogeneous ...
    matrix Tc
% as well as the vector from the origin of Tp to the origin of Tc
%

% position of the origin (in the world frame) of the frame ...
    associated with Tp
p1 = Tp(1:3,4);

% position of the origin (in the world frame) of the frame ...
    associated with Tc
p2 = Tc(1:3,4);

% rotation matrix (w.r.t the world frame) associated with Tc
R = Tc(1:3,1:3);

% plot vector from the origin of Tp to the origin of Tc
plot_line(p1,p2,'k');

scale = 0.5; % plot shorter coordinate axis (for better view)
plot_line(p2,p2+R(:,1)*scale,'b',2);
plot_line(p2,p2+R(:,2)*scale,'g',2);
plot_line(p2,p2+R(:,3)*scale,'r',2);

%%%EOF plot_T
% ...
```

```
function plot_line(p1,p2,color,lw)
```

```
%
% plots a line (in 3D) from point p1 to point p2 with
% color - color
% lw    - line width
%
% Example:
% --------
% plot_line([0;0;0],[1;1;1],'b',2)
%

if nargin < 4
    lw = 1;
end

plot3([p1(1) p2(1)], [p1(2) p2(2)], [p1(3) p2(3)], color, ...
    'LineWidth',lw)

%%%EOF plot_line
% ...

%%%EOF
```

2.2 Function to define the system in the bMSd simulator

File: `model_bMSd.m`

```
function SP = model_bMSd(pos,rot,e)
%
% this file is used in the example in Lab_03
%
% a system is defined using:
% ------------------------
% pos(:,i)  - position of frame i as seen from frame i-1
%             (frame 0 is the world frame)
%
% rot(:,i)  - frame i is rotated w.r.t. frame i-1 using
%             x->y->z Euler angles (current axis)
%
% e.pos     - position of the end-effector in last frame
% e.pos     - x->y->z Euler angles (current frame) w.r.t last frame
%

% definition of system structure
% ------------------------------------
SP.n = size(pos,2);
SP.C = 0:SP.n+1;
SP.mode = 1; % fixed base

% definition of joints
% I assume that the CoM of link i coincides with the input joint ...
    of link i
% ------------------------------------
for i = 1:SP.n
    SP.J(i).t    = pos(:,i);
    SP.J(i).f    = zeros(3,1); % CoM coincides with the joint
```

```
    SP.J(i).rpy  = rot(:,i);
    SP.J(i).type = 'R';
end

% definition of links
% I use trivial values here
% ---------------------------------------------------------
for i = 1:SP.n+1
    SP.L(i).m = 1;
    SP.L(i).I = eye(3);
end

% definition of end-effectors
% (only one end-effector)
% ---------------------------------------------------------
SP.bN = SP.n+1;
SP.bP = e.pos;
SP.bR = rpy2R(e.rot);

%%%EOF
```