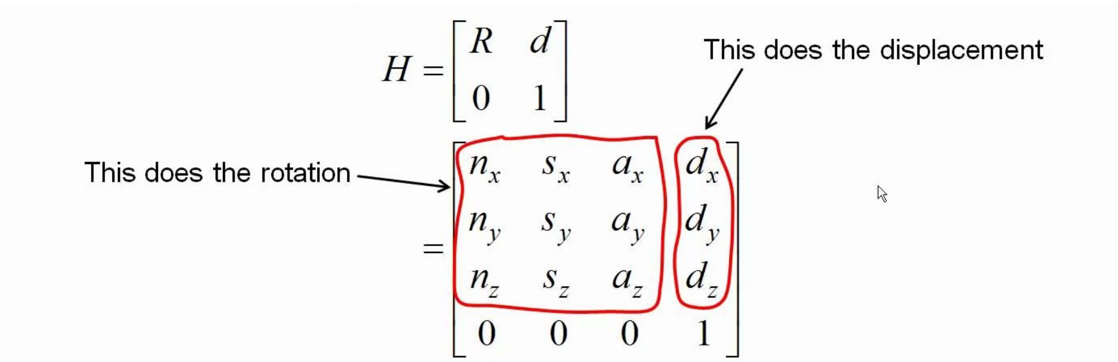
Kinematics on MATLAB for multi-link robots

Notebook:	Sirena Documentation		
Created:	4/6/2018 1:07 PM	**Updated:**	4/6/2018 1:21 PM
Author:	karthikram570@gmail.com		

Abstract :This document describes how make your own robot on MATLAB for kinematics analysis , for example figuring out forward and inverse kinematics. This can be helpful creating walk trajectories and visualizing the same on MATLAB. One can also attach mass and inertial properties to these models to do dynamic analysis.

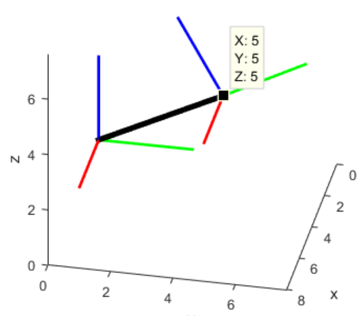
Step 1: Understanding Homogeneous Transformation Matrix
Homogeneous transformation matrices are used to represent transformation frames which encapsulates position and orientation data of an object in space.For a robot, it might be position and orientation of a joint in space.
The [dx, dy,dz] values are the x,y,z coordinates in space and [nx ny nz],[sx sy sz],[ax ay az] are normalized vectors emerging out of [dx,dy,dz]. A 4X4 identity matrix will be a frame located at origin with the x vector pointing on positive x direction with a magnitude of 1, similarly for y and z.



$$H = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & s_x & a_x & d_x \\ n_y & s_y & a_y & d_y \\ n_z & s_z & a_z & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Step 1: Understanding Homogeneous Transformation Matrix
Homogeneous transformation matrices are used to represent transformation frames which encapsulates position and orientation data of an object in space.For a robot, it might be position and orientation of a joint in space.
The [dx, dy,dz] values are the x,y,z coordinates in space and [nx ny nz],[sx sy sz],[ax ay az] are normalized vectors emerging out of [dx,dy,dz]. A 4X4 identity matrix will be a frame located at origin with the x vector pointing on positive x direction with a magnitude of 1, similarly for y and z.

```
F0 = T(0,0,0); %frame at origin
F1 = F0*T(5,5,5)*RX(0.1); %translation and rotation.
hold on;
plot_line(F0,F1); %this function plots a line between two frames
plot_transformation_frame(F0); %plots the transformation frame
plot_transformation_frame(F1); %plots the transformation frame
axis equal;
xlabel('x');
ylabel('y');
zlabel('z');
```



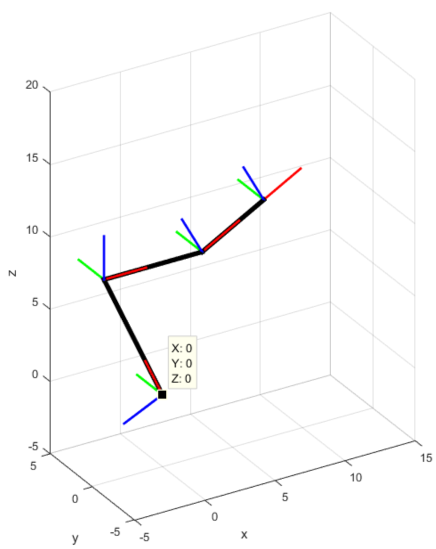

```
F0 =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

```
F1 =

    1.0000         0         0    5.0000
         0    0.8776   -0.4794    5.0000
         0    0.4794    0.8776    5.0000
         0         0         0    1.0000
```

Step 2: Creating Forward kinematic chains using HTMs

Now that you know how to initialize frames and plot them. Lets try making a 3 DOF robotic arm

```
%joint angles in radians
joint1_val = -2;
joint2_val = 2;
joint3_val = -0.5;

%Forward kinematics
F0 = T(0,0,0)*RY(joint1_val);
F1 = F0*T(10,0,0)*RY(joint2_val);
F2 = F1*T(7,0,0)*RY(joint3_val);
F3 = F2*T(5,0,0);

%plotting
hold on;
plot_line(F0,F1);
plot_line(F1,F2);
plot_line(F2,F3);

plot_transformation_frame(F0); %plots the transformation frame
plot_transformation_frame(F1); %plots the transformation frame
plot_transformation_frame(F2); %plots the transformation frame
plot_transformation_frame(F3); %plots the transformation frame

axis equal;
xlabel('x');
ylabel('y');
zlabel('z');
axis([-5 15 -5 5 -5 20])
grid on;
hold off
```




```
F0 =

   -0.6557         0   -0.7550         0
         0    1.0000         0         0
    0.7550         0   -0.6557         0
         0         0         0    1.0000

>> F1

F1 =

    0.9367         0   -0.3500   -6.5572
         0    1.0000         0         0
    0.3500         0    0.9367    7.5500
         0         0         0    1.0000
```

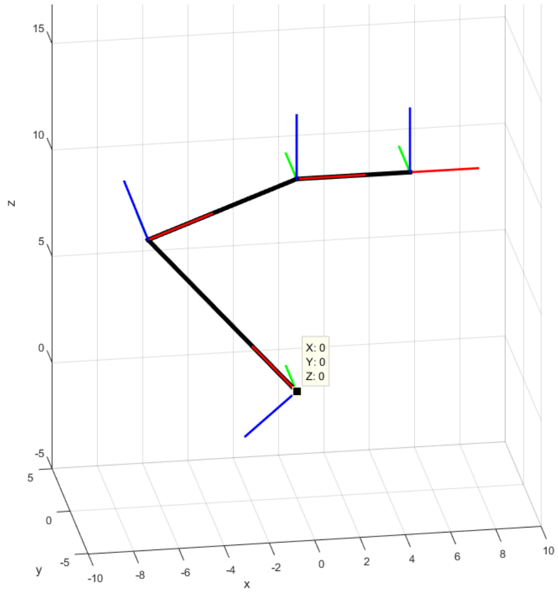
```
>> F2

F2 =

    1.0000         0    0.0000   -0.0000
         0    1.0000         0         0
    0.0000         0    1.0000   10.0000
         0         0         0    1.0000

>> F3

F3 =

    1.0000         0    0.0000    5.0000
         0    1.0000         0         0
    0.0000         0    1.0000   10.0000
         0         0         0    1.0000
```

Step 3: Applying Inverse Kinematics

To understand more about inverse kinematics, you can refer to this blog post. In this step we will implement inverse kinematics of the 3DOF arm we just created.
If you give a point outside the workspace of the arm, it will the inverse kinematic function will throw an error.

```
%joint values in radians
joint1_val = -2;
```

```
joint2_val = 2;
joint3_val = -0.5;

%Link lengths
L=[10,7,5];

%Inverse Kinematics *comment this section to perform only forward
%kinematics
x = 5;   %x position
z = 10;    %z position
phi = 0;  %end effector orientation
[JA] = R2_IK(x,z,L,phi,-1); %inverse kinematic function
offset=[0 0 0];             %offset to match up FK and IK
dmt = [-1 -1 -1];           %direction matrix to match up FK and IK
JA = JA+offset;
JA = JA.*dmt;
joint1_val = JA(1);
joint2_val = JA(2);
joint3_val = JA(3);

%Forward Kinematics
F0 = T(0,0,0)*RY(joint1_val);
F1 = F0*T(L(1),0,0)*RY(joint2_val);
F2 = F1*T(L(2),0,0)*RY(joint3_val);
F3 = F2*T(L(3),0,0);

%plotting
hold on;
plot_line(F0,F1);
plot_line(F1,F2);
plot_line(F2,F3);

plot_transformation_frame(F0); %plots the transformation frame
plot_transformation_frame(F1); %plots the transformation frame
plot_transformation_frame(F2); %plots the transformation frame
plot_transformation_frame(F3); %plots the transformation frame

axis equal;
xlabel('x');
ylabel('y');
zlabel('z');
axis([-10 10 -5 5 -5 20])
grid on;
hold off
```




```
F0 =

   -0.6557         0   -0.7550         0
         0    1.0000         0         0
    0.7550         0   -0.6557         0
         0         0         0    1.0000

>> F1

F1 =

    0.9367         0   -0.3500   -6.5572
         0    1.0000         0         0
    0.3500         0    0.9367    7.5500
         0         0         0    1.0000
```

```
>> F2

F2 =

    1.0000         0    0.0000   -0.0000
         0    1.0000         0         0
    0.0000         0    1.0000   10.0000
         0         0         0    1.0000

>> F3

F3 =

    1.0000         0    0.0000    5.0000
         0    1.0000         0         0
    0.0000         0    1.0000   10.0000
         0         0         0    1.0000
```

Step 4: Generating trajectories and animating the robot
In this section, we will create a trajectory (circle) and make the arm follow it.

```
%trajectroy generation - cicle with radius 5
radius=5;
theta=0:0.1:2*pi;
X=radius*cos(theta)+7;
Z=radius*sin(theta)+10;
pause(7)

%joint values in radians
joint1_val = -2;
joint2_val = 2;
joint3_val = -0.5;

%Link lengths
L=[10,7,5];


%executing trajectory


for i=1:1:length(X)
    %Inverse Kinematics *comment this section to perform only forward
    %kinematics
     x = X(i);    %x position
     z = Z(i);    %z position
     phi = 0;     %end effector orientation
     [JA] = R2_IK(x,z,L,phi,-1); %inverse kinematic function
     offset=[0 0 0];             %offset to match up FK and IK
     dmt = [-1 -1 -1];           %direction matrix to match up FK and IK
     JA = JA+offset;
     JA = JA.*dmt;

     joint1_val = JA(1);
     joint2_val = JA(2);
     joint3_val = JA(3);

    %Forward Kinematics
     F0 = T(0,0,0)*RY(joint1_val);
```

```
    F1 = F0*T(L(1),0,0)*RY(joint2_val);
    F2 = F1*T(L(2),0,0)*RY(joint3_val);
    F3 = F2*T(L(3),0,0)

    %plotting
    plot_line(F0,F1);
    hold on;

    plot_line(F1,F2);
    plot_line(F2,F3);

    plot_transformation_frame(F0); %plots the transformation frame
    plot_transformation_frame(F1); %plots the transformation frame
    plot_transformation_frame(F2); %plots the transformation frame
    plot_transformation_frame(F3); %plots the transformation frame

    plot3(X,zeros(1,length(X)),Z)
    axis equal;
    xlabel('x');
    ylabel('y');
    zlabel('z');
    axis([-10 15 -5 5 -5 20])
    grid on;

    pause(0.1)
    hold off;
end
```

The X-Z coordinates are extracted from the trajectory and are fed to the inverse kinematics functions, followed by forward kinematics and plotting.
The result of which...