Robotic Control System

Novint Falcon to SCARA with Spherical Wrist

EECE 487 – Final Project

University of British Columbia

March 28, 2013

Group Members:

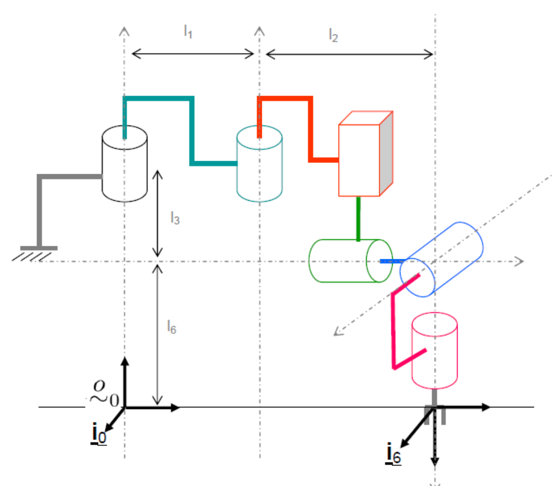
Po Liu 13623079
Timothy Lee 24543092
Changhoon Baek 45322096
Julian Zhang 25358094
James Zhan 15314073
Hua Chang 22837080

Introduction

The Novint Falcon controller is a haptic device that takes position and force measurements of its ball end-effector as inputs while providing touch feedback to the user. While primarily used for the gaming industry, it has potential applications for robotic control research due to its connectivity with scientific computing platforms.

The objective of this project was to build a teleoperation system that received position input from Novint Falcon and translate the input into controlling a virtual SCARA robot via MATLAB Simulink.

This report will discuss hardware and software tools used, the design decisions implemented, and joint variable approaches used. A diagram of the final SCARA arm with spherical wrist is shown to the right.

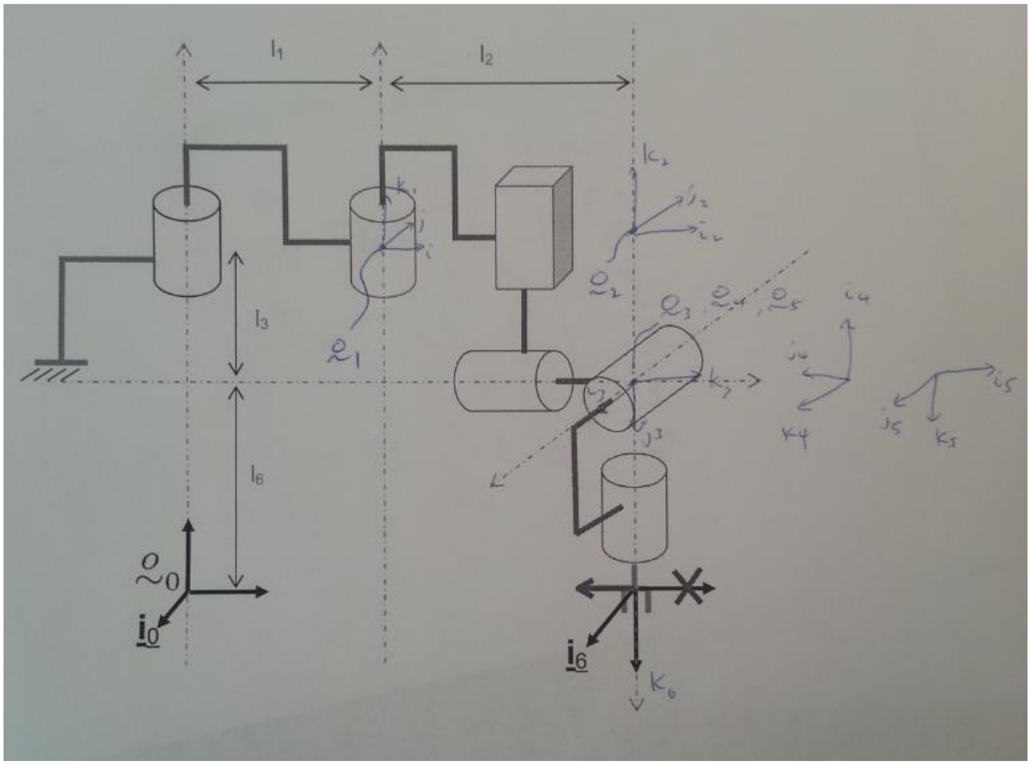


Exercise 1 - Kinematics of SCARA robot with spherical wrist

[Part A]

Before any operation of the robot can be performed, the frames for each of the joints must be established along with the rotational and translational relationship between the frames. It is thus important to determine the homogeneous transformation matrices between each frame as well as from the base frame to the end effector frame. These manipulator direct kinematics were solved by using the following series of steps:

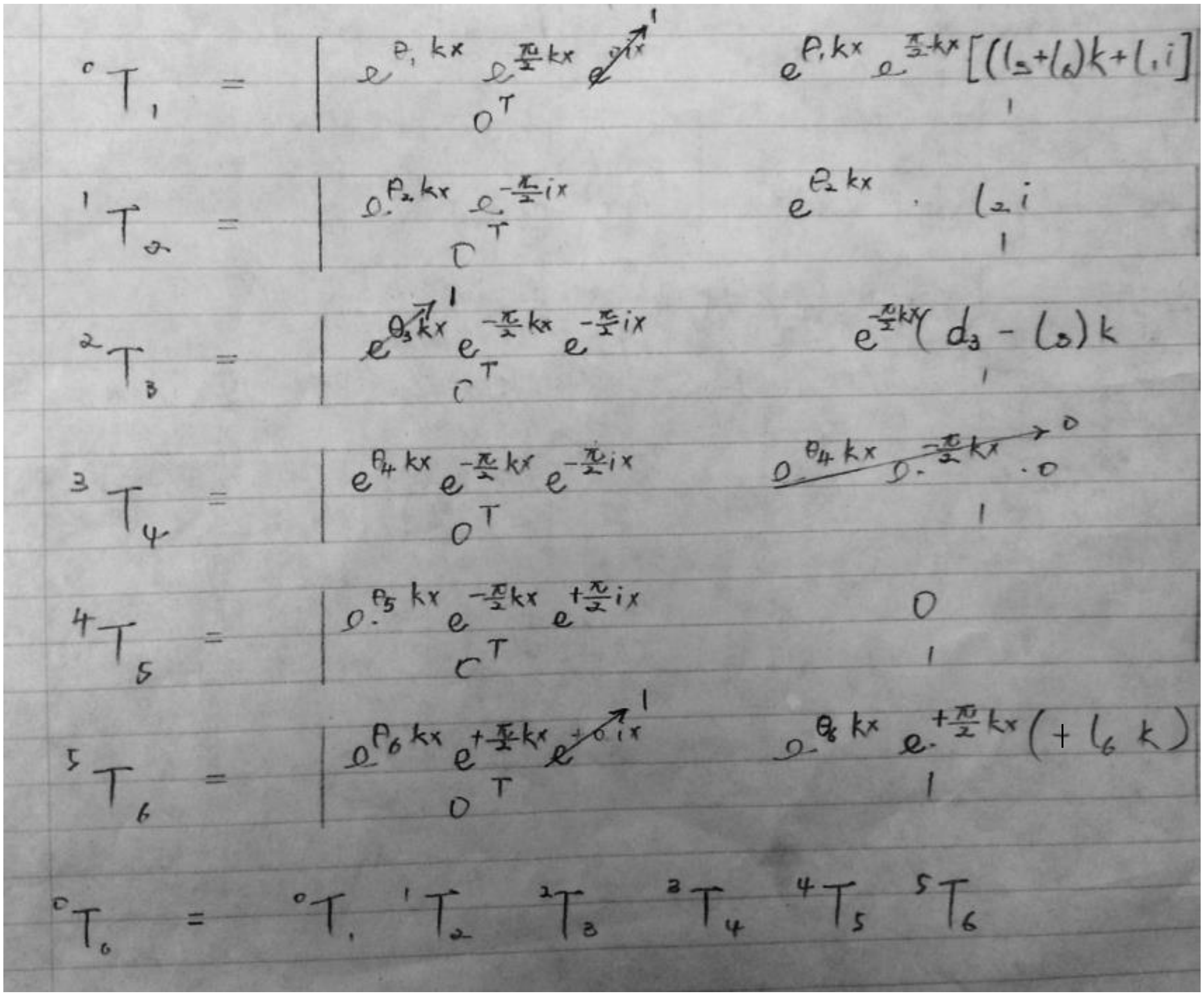
1. Assign coordinate frames to each link in the SCARA robot with spherical wrist.



2. Applying Denavit-Hartenberg convention, we determined for each link the angle, offset, length, and twist of each homogeneous transformation.

Link	Θ_i	d_i	a_i	α_i
1	$(\Theta_1) + \pi/2$	l3+l6	l1	0
2	(Θ_2)	0	l2	0
3	$-\pi/2$	-l3 + d3	0	$-\pi/2$
4	$(\Theta_4) - \pi/2$	0	0	$-\pi/2$
5	$(\Theta_5) - \pi/2$	0	0	$\pi/2$
6	$(\Theta_6) + \pi/2$	l6	0	0

3. Using the table of DH parameters shown above, build each of the transformation matrices. We can then multiple all of the matrices together in order to obtain the relationship between the base frame and the end-effector frame.



$$ {}^{0}T_{1} = \left[\begin{array}{c|c} e^{\theta_1 kx}\, e^{\frac{\pi}{2}kx}\, \cancel{e^{ix}}\ 1 & e^{\theta_1 kx}\, e^{\frac{\pi}{2}kx}\left[((l_3+l_6)k + l_1 i\right] \\ 0^T & 1 \end{array}\right] $$

$$ {}^{1}T_{2} = \left[\begin{array}{c|c} e^{\theta_2 kx}\, e^{-\frac{\pi}{2}ix} & e^{\theta_2 kx} \cdot l_2 i \\ 0^T & 1 \end{array}\right] $$

$$ {}^{2}T_{3} = \left[\begin{array}{c|c} \cancel{e^{\theta_3 kx}}\ 1\ e^{-\frac{\pi}{2}kx}\, e^{-\frac{\pi}{2}ix} & e^{-\frac{\pi}{2}kx}(d_3 - l_5)k \\ 0^T & 1 \end{array}\right] $$

$$ {}^{3}T_{4} = \left[\begin{array}{c|c} e^{\theta_4 kx}\, e^{-\frac{\pi}{2}kx}\, e^{-\frac{\pi}{2}ix} & \cancel{e^{\theta_4 kx}\, e^{-\frac{\pi}{2}kx} \cdot 0} \to 0 \\ 0^T & 1 \end{array}\right] $$

$$ {}^{4}T_{5} = \left[\begin{array}{c|c} e^{\theta_5 kx}\, e^{-\frac{\pi}{2}kx}\, e^{+\frac{\pi}{2}ix} & 0 \\ 0^T & 1 \end{array}\right] $$

$$ {}^{5}T_{6} = \left[\begin{array}{c|c} e^{\theta_6 kx}\, e^{+\frac{\pi}{2}kx}\, \cancel{e^{+0\,ix}}\ 1 & e^{\theta_6 kx}\, e^{+\frac{\pi}{2}kx}(+l_6 k) \\ 0^T & 1 \end{array}\right] $$

$$ {}^{0}T_{6} = {}^{0}T_{1}\ {}^{1}T_{2}\ {}^{2}T_{3}\ {}^{3}T_{4}\ {}^{4}T_{5}\ {}^{5}T_{6} $$

[Part B]

Required for robotic control, the Jacobian is a matrix of partial derivatives that linearly translate between the joint variable velocity into the end effector translational and angular velocities.

To determine the manipulator Jacobian symbolically, we write out the following initially:

$$ \underline{J} = \begin{bmatrix} \underline{k}_0 \times (\underset{\sim}{o}_6 - \underset{\sim}{o}_0) & \underline{k}_1 \times (\underset{\sim}{o}_6 - \underset{\sim}{o}_1) & \underline{k}_2 & \underline{k}_3 \times (\underset{\sim}{o}_6 - \underset{\sim}{o}_3) & \underline{k}_4 \times (\underset{\sim}{o}_6 - \underset{\sim}{o}_3) & \underline{k}_5 \times (\underset{\sim}{o}_6 - \underset{\sim}{o}_3) \\ \underline{k}_0 & \underline{k}_1 & 0 & \underline{k}_3 & \underline{k}_4 & \underline{k}_5 \end{bmatrix} $$

Because the SCALA manipulator has a spherical wrist, we are able to perform decoupling of singularities into arm and wrist singularities. Using elementary row operations we obtain:

$$
\begin{aligned}
\underline{J} &\sim \underbrace{\begin{bmatrix} \underline{k}_0 \times (\underset{\sim}{o}_3 - \underset{\sim}{o}_0) & \underline{k}_1 \times (\underset{\sim}{o}_3 - \underset{\sim}{o}_1) & \underline{k}_2 \\ \underline{k}_0 & \underline{k}_1 & 0 \end{bmatrix}}_{\text{arm singularities}} \underbrace{\begin{matrix} 0 & 0 & 0 \\ \underline{k}_3 & \underline{k}_4 & \underline{k}_5 \end{matrix}}_{\text{wrist singularities}} \\
&= \begin{bmatrix} \underline{J}_{11} & 0 \\ \underline{J}_{21} & \underline{J}_{22} \end{bmatrix}
\end{aligned}
$$

Wrist Singularities:

- *J22* is singular when k3, k4, k5 are coplanar. This can only happen when k3 and k5 are aligned, since k4 is perpendicular to both.

Arm Singularities:

- *J11* is singular when its three vectors are coplanar. This can only happen when axes k0 is parallel to (o3-o0) or when k0 x (o3-o0) = 0.
  - A visualization of this scenario would be when the wrist center is along the axis of rotation of the base frame.

- Another singularity occurs when k0 x k1 = 0; however, this situation is impossible given the joint configuration of the SCARA arm (k0 and k1 can never be parallel to each other).

[Part C]

There may arise situations when the desired end-effector location is known without knowledge of the joint parameters. Solving manipulator inverse kinematics involves first decoupling the problem into an inverse arm problem and an inverse wrist problem.

Since the SCARA arm with spherical wrist is a 6 degree-of-freedom robot, the following statements can be made:

$$
\begin{aligned}
\underset{\sim}{o}_3 &= \underset{\sim}{o}_4 = \underset{\sim}{o}_5 \\
\underline{C}_6 &= \underline{C}_5 e^{\theta_6 k \times} \\
\underset{\sim}{o}_6 &= \underset{\sim}{o}_5 + \underline{C}_5 e^{\theta_6 k \times}(d_6 k) \enspace .
\end{aligned}
$$

where d6 is the distance from the center of the spherical wrist to the origin of the end-effector. Therefore, o3 is a known position because

$$
\underset{\sim}{o}_3 = \underset{\sim}{o}_6 - d_6 \underline{k}_6 = \underset{\sim}{o}_d - d_6 \underline{C}_d k
$$

$$\underset{\sim}{O}_3 = \underset{\sim}{O}_4 = \underset{\sim}{O}_5$$

$$C_6 = C_5 \cdot e^{\theta_6 kx} e^{\frac{\pi}{2} ix}$$

$$\underset{\sim}{O}_6 = \underset{\sim}{O}_5 + e^{\theta_6 kx} e^{\frac{\pi}{2} kx} (l_6 k)$$

$$\underset{\sim}{O}_3 = \underset{\sim}{O}_5 = \underset{\sim}{O}_6 - l_6 C_6 k = \underset{\sim}{O}_6 - l_6 e^{\theta_6 kx} e^{\frac{\pi}{2} kx}$$

$$C_6 = e^{\theta_4 k_3 x} e^{-\frac{\pi}{2} k_3 x} e^{-\frac{\pi}{2} i_3 x} e^{\theta_5 i_3 x} e^{-\frac{\pi}{2} i_3 x} e^{\frac{\pi}{2} j_3 x} e^{\theta_6 j_3 x} e^{\frac{\pi}{2} i_3 x}$$

$$C_6 = C_3 \underline{e^{\theta_4 kx} e^{-\frac{\pi}{2} kx} e^{-\frac{\pi}{2} ix}}\ \underline{e^{\theta_5 ix} e^{-\frac{\pi}{2} ix} \cancel{e^{\frac{\pi}{2} jx}}}\ \underline{e^{\theta_6 jx} \cancel{e^{\frac{\pi}{2} jx}}}$$

$$e^{-\theta_6 k_6 x} C_6 = e^{\theta_4 k_3 x} e^{-\frac{\pi}{2} kx} C_3 e^{-\pi ix} e^{\theta_5 ix}$$

$$e^{-\theta_6 k_6 x} C_6 i = e^{\theta_4 k_3 x} C_3 i$$

$$e^{-\theta_6 k_6 x} i_6 = k_4 = e^{\theta_4 k_3 x} i_3$$

$$|\theta_4| = 2 \arctan \frac{|k_4 - i_3|}{|k_4 + i_3|}$$

$$\mathrm{sign}(\theta_4) = \cos^{-1} \frac{k_4^T i_3}{|k_4||i_3|}$$

$$|\theta_6| = 2 \arctan \frac{|k_4 - i_6|}{|k_4 + i_6|}$$

$$\mathrm{sign}(\theta_6) = \cos^{-1} \frac{k_4^T i_6}{|k_4||i_6|}$$

$$e^{-\theta_6 i_3 x} C_6 k = e^{(\theta_5 - \pi) i_3 x} C_3 k$$

$$k_6 = e^{(\theta_5 - \pi) i_3 x} k_3$$

$$|\theta_5| = 2 \arctan \frac{|k_6 - k_3|}{|k_6 + k_3|} - \frac{\pi}{2}$$

$$\mathrm{sign}(\theta_5) = \mathrm{sign}\left[k_3^T (i_3 \times k_6) \right]$$

Solving for the first three robot joints with geometric properties…

IK ARM

$$\cos\theta = \frac{dx^2+dy^2-L_1^2-L_2^2}{2\cdot L_1\cdot L_2}$$

$$\theta_2 = \pm\tan^{-1}\left(\sqrt{(1-\cos\theta^2)}/\cos\theta\right)$$

$$\theta_1 = \tan^{-1}\left(\frac{dy}{dx}\right) + \cos^{-1}\left(\frac{dx^2+dy^2+L_1^2-L_2^2}{2\cdot L_1\cdot\sqrt{dx^2+dy^2}}\right)$$

Thus,

Arm = $[\,2\theta_1 \quad 2\theta_2 \quad dz+L_6 \quad 0 \quad 0 \quad 0\,]$

Solving for the inverse kinematics to find the wrist angles Θ_4, Θ_5 , Θ_6 (renumbered below).

IK Grabber.

$$\theta_1 = \tan^{-1}\left(\frac{dy}{dx}\right)$$

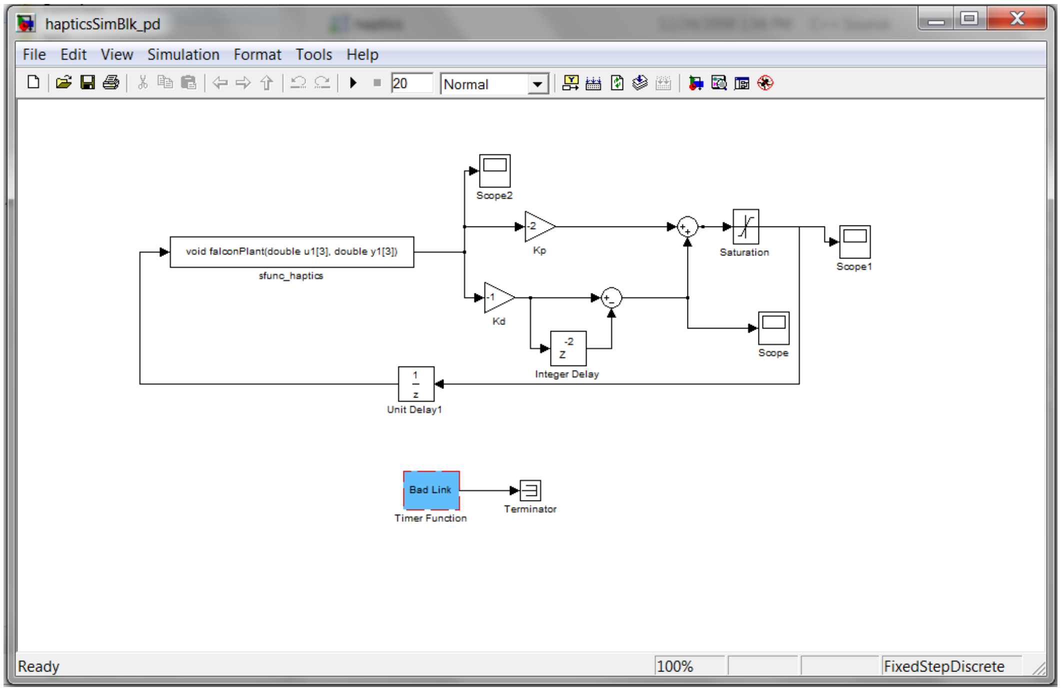
$$\theta_2 = \tan^{-1}\left(\frac{dz}{\sqrt{dx^2+dy^2}}\right)$$

θ_3 = unknown. 0 as default.

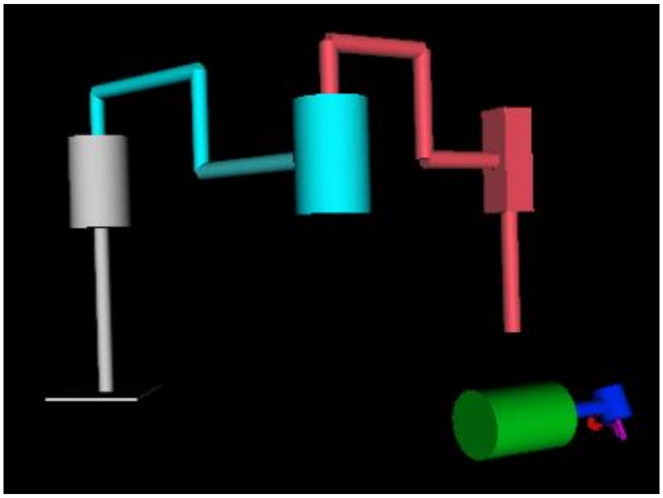
Wrist = $[\,0 \quad 0 \quad 0 \quad 2\theta_1 \quad 2\theta_2 \quad 0\,]$.

Exercise 2 – Teleoperation System of SCARA robot using Jacobian

The teleoperation system was setup by connecting a Novint Falcon controller via USB to an ECE lab computer preloaded with appropriate drivers. Novint input can be read using MATLAB's Simulink which is platform for 3D model simulation. The user interface is shown below with the provided block diagram for falconPlant:



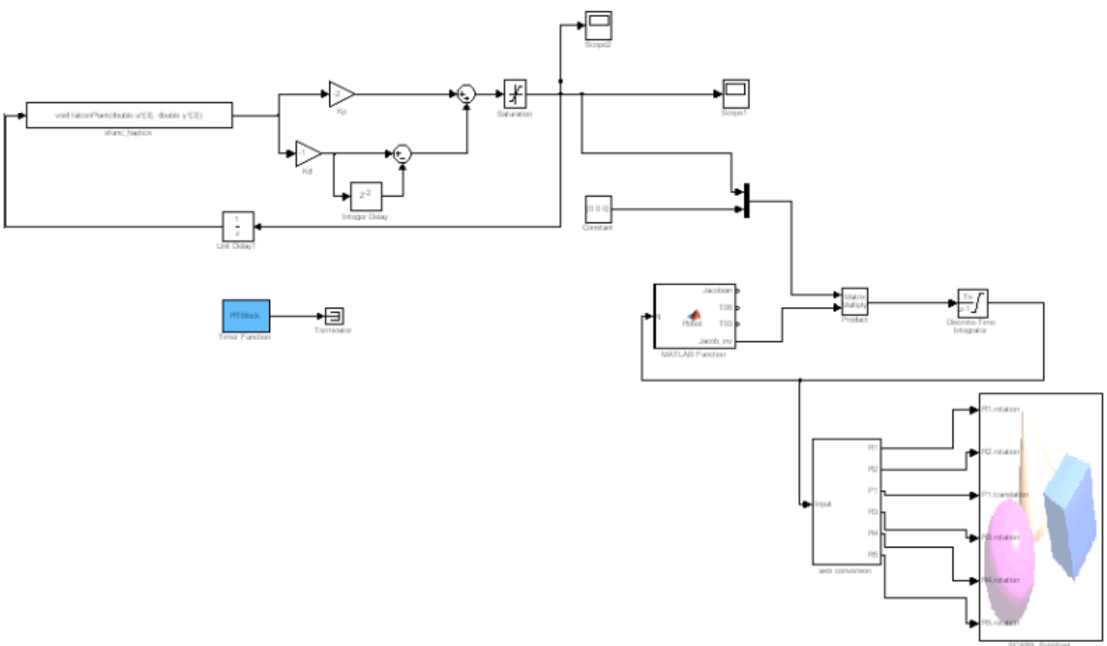
The SCARA robot with spherical wrist was implemented in a Virtual Reality Simulation block. Joints can be individually added as Transform nodes for each link with parameters that describe its location and geometry in 3D space. Additional parameters can be set to specify color, texture, and type (prismatic or rotational). The visual representation of the gripper is shown below.



The PD control mode was set with relatively low gains within kp and kd to be comfortable to the hand. Scopes were used for troubleshooting and debugging the input variables that the Novint displacement to SCARA joint variables function used

A dead band was created within the Discrete-Time Integrator block by setting the upper and lower saturation limits at +50 and -50. This prevented the Novint controller from moving erratically when the joystick was released or held softly.

The top-level block diagram for the SCARA robot control using Jacobian to solve for SCARA link variables is shown below.



MATLAB code for inverse Jacobian function

```
function [Jacobian,T06,T03,Jacob_inv]=Robot(q)
%Robot(1 2 3 4 5 6)
a1 = q(1);
a2 = q(2);
d3 = q(3);
a4 = q(4);
a5 = q(5);
a6 = q(6);

L1=1; L2=1; L3=0.5; L6=0.2;

        input = [pi/2+a1 L6+L3 L1   0;
                  a2    0    L2     0;
```

```
                 -pi/2 -L3+d3 0 -pi/2;
                 -pi/2+a4 0 0 -pi/2;
                 -pi/2+a5 0 0 pi/2;
                 -pi/2+a6 -L6 0 0];

   % function output = TRANS(input,i)

    T01 =    [cos(input(1,1)) -sin(input(1,1))*cos(input(1,4))
sin(input(1,1))*sin(input(1,4)) (input(1,3))*cos(input(1,1));
              sin(input(1,1)) cos(input(1,1))*cos(input(1,4)) -
cos(input(1,1))*sin(input(1,4)) (input(1,3))*sin(input(1,1));
                 0                  sin(input(1,4))
cos(input(1,4))            input(1,2)                       ;
                 0          0                           0
1            ];
    T12 =    [cos(input(2,1)) -sin(input(2,1))*cos(input(2,4))
sin(input(2,1))*sin(input(2,4)) (input(2,3))*cos(input(2,1));
              sin(input(2,1)) cos(input(2,1))*cos(input(2,4)) -
cos(input(2,1))*sin(input(2,4)) (input(2,3))*sin(input(2,1));
                 0                  sin(input(2,4))
cos(input(2,4))            input(2,2)                       ;
                 0          0                           0
1            ];
    T23 =    [cos(input(3,1)) -sin(input(3,1))*cos(input(3,4))
sin(input(3,1))*sin(input(3,4)) (input(3,3))*cos(input(3,1));
              sin(input(3,1)) cos(input(3,1))*cos(input(3,4)) -
cos(input(3,1))*sin(input(3,4)) (input(3,3))*sin(input(3,1));
                 0                  sin(input(3,4))
cos(input(3,4))            input(3,2)                       ;
                 0          0                           0
1            ];
    T34 =    [cos(input(4,1)) -sin(input(4,1))*cos(input(4,4))
sin(input(4,1))*sin(input(4,4)) (input(4,3))*cos(input(4,1));
              sin(input(4,1)) cos(input(4,1))*cos(input(4,4)) -
cos(input(4,1))*sin(input(4,4)) (input(4,3))*sin(input(4,1));
                 0                  sin(input(4,4))
cos(input(4,4))            input(4,2)                       ;
                 0          0                           0
1            ];
    T45 =    [cos(input(5,1)) -sin(input(5,1))*cos(input(5,4))
sin(input(5,1))*sin(input(5,4)) (input(5,3))*cos(input(5,1));
              sin(input(5,1)) cos(input(5,1))*cos(input(5,4)) -
cos(input(5,1))*sin(input(5,4)) (input(5,3))*sin(input(5,1));
                 0                  sin(input(5,4))
cos(input(5,4))            input(5,2)                       ;
                 0          0                           0
1            ];
    T56=     [cos(input(6,1)) -sin(input(6,1))*cos(input(6,4))
sin(input(6,1))*sin(input(6,4)) (input(6,3))*cos(input(6,1));
              sin(input(6,1)) cos(input(6,1))*cos(input(6,4)) -
cos(input(6,1))*sin(input(6,4)) (input(6,3))*sin(input(6,1));
                 0                  sin(input(6,4))
cos(input(6,4))            input(6,2)                       ;
                 0          0                           0
1            ];
```

```
    T0  = eye(4);    O_0 = T0(1:3,4);      k0 = [0 0 1]';
    T01 = T0*T01;    O_1 = T01(1:3,4);     k1 = [T01(1:3,3)];
    T02 = T01*T12;   O_2 = T02(1:3,4);     k2 = [T02(1:3,3)];
    T03 = T02*T23;   O_3 = T03(1:3,4);     k3 = [T03(1:3,3)];
    T04 = T03*T34;   O_4 = T04(1:3,4);     k4 = [T04(1:3,3)];
    T05 = T04*T45;   O_5 = T05(1:3,4);     k5 = [T05(1:3,3)];
    T06 = T05*T56;   O_6 = T06(1:3,4);     k6 = [T06(1:3,3)];

    Jacobian=[cross(k0,(O_6-O_0)), cross(k1,(O_6-O_1)), k2, cross(k3,(O_6-
O_3)), cross(k4,(O_6-O_4)), cross(k5,(O_6-O_5));
                k0, k1, zeros(3,1), k3, k4, k5];

    Jacob_inv = inv(Jacobian + eye*1E-1)

end
```

Mathematically, to solve for the joint variables:

Let x_m be the Novint joint deflection.

$$x_m = [a_m \;\; b_m \;\; c_m]$$

$$\text{set } v = [x_m^T \; 0 \; 0 \; 0]^T = [a_m \; b_m \; c_m \; 0 \; 0 \; 0]^T$$

$$\text{solve for } \dot{q} = [\dot{\theta}_1 \; \dot{\theta}_2 \; \dot{d}_3 \; \dot{\theta}_4 \; \dot{\theta}_5 \; \dot{\theta}_6]^T$$

$$v = J(q)\dot{q}$$

Therefore…

$$J^{-1}(q)v = \dot{q}$$
$$\int \dot{q} = q$$

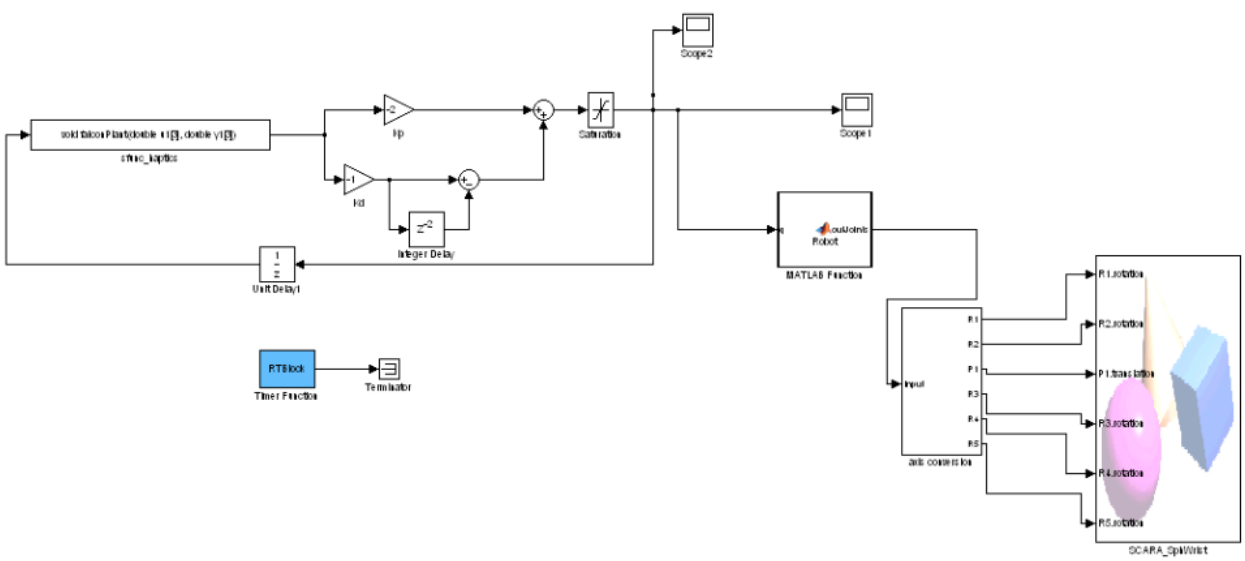
Discussion on Singular Configurations

- When the robot is moved to a singular configuration, the joint variables are immediately reset because they are undefined at the boundary. Singularities occur because of the division of the determinant within the inverse Jacobian. When the determinant is zero, a divide by zero occurs and all joint variables cannot be defined by the algorithm.

Exercise 3 – Inverse Kinematics-based Resolved Motion Control

The identical teleoperation setup for the Novint controller was used with the inverse kinematics approach to calculate joint variables. The elbow-arm was obtained from drawing a 2D diagram of the first three links from the origin to the prismatic joints. Making use of the given end location of *o3*, Θ_1 and Θ_2 were solved using the Cosine Law. The spherical joints can be described by three Euler rotations. The directions of Θ_4 to Θ_6 are then found by relating the coordinate vectors to DH conventions and the known position of coordinate frame of the third joint.

Alternations were made to the block diagram which is shown below. A new function that translates the Novint displacement to joint variables directly was implemented using inverse kinematics while the discrete-time integrator was removed.



MATLAB code for inverse kinematics function

```
% choose one of the following function to perform inverse kinematics for
% either arm or grabber

% function y=invKarm(input)
% dx=input(1)/2;
% dy=input(2)/2;
% dz=input(3);
% L1=1;L2=1;L6=0.2;
%
%  ca2=(dx^2+dy^2-L1^2-L2^2)/(2*L1*L2);
%
%  if dx>=0 && dy>=0
%  a2=atan(sqrt(1-ca2^2)/ca2);
```

```
%   else
%   a2=atan(-sqrt(1-ca2^2)/ca2);
%   end
%
%
%   B=acos((dx^2+dy^2+L1^2-L2^2)/(2*L1*sqrt(dx^2+dy^2)));
%   a1=atan(dy/dx)+B;
%   y=[2*a1 2*a2 dz+L6 0 0 0];
%
% end

function y=invKg(input)
ax=input(1);
ay=input(2);
az=input(3)-1.5;

a1=atan(ay/ax);

a2=atan(az/(sqrt(ax^2+ay^2)));

a3=0;
    y=[0 0 0 2*a1 2*a2 a3];
end
```

Discussion on the relative advantages of the two approaches

- As can be seen from the block diagram above, the inverse kinematics approach requires far less components and logic. In addition, this approach will circumvent the issue with singularities that plagued the inverse Jacobian method. The reason why singularities are not an issue is because inverse kinematics forces a specification of the joint variables for every position in space. The inverse Jacobian has undefined behavior at these points. The inverse calculations must specify a sign from the trigonometric functions and therefore the approach will "choose" a specific joint configuration at the singularity.

  Since only the position of coordinate frame 6 is provided by the ball controller, a disadvantage would be that no angle is specified for Θ_6.

  - Observations:
    - The SCARA stops right at the desired location that is proportional to the movement of the ball controller.

- The inverse Jacobian approach has the advantage of being far easier to establish because it has a general form in which DH parameters can be specified. The Jacobian matrix can be found through iterative manipulations of the coordinate frames and methodically writing out the homogenous transformation matrices. On the other hand, the inverse kinematics approach requires detailed trigonometric gymnastics that are heavily reliant on the unique setup of the joints in the robot. The inverse kinematics approach is therefore far more difficult to calculate if a greater number of joints were involved – especially if a spherical wrist was not a component.

- Observations
    - The SCARA keeps moving in the same direction as the ball controller at a speed that is relative to the magnitude of the displacement of ball until it reaches the preset upper limit.