Space Station Remote Manipulator System (SSRMS) Kinematics

John Karasinski

March 20, 2018

Contents

1 Introduction

The Space Station Remote Manipulator System (SSRMS) was designed and built to construct the International Space Station (ISS) and grapple with visiting space vehicles. The SSRMS is a seven degree of freedom (DoF) manipulator consisting entirely of revolute joints. The arm is symmetric, consisting of a 3DoF (roll, yaw, pitch) "shoulder", an "elbow" pitch joint, and a 3DoF (pitch, yaw, roll) "wrist". Due to this symmetric structure, the arm has the ability to "walk" along the station, greatly increasing it's available working space. The arm can lock the wrist to a grapple fixture, then disconnect the shoulder (which becomes the new wrist) to walk along the station. See Figure 1 for a picture of the arm grappling a visiting Cygnus vehicle.

The SSRMS is operating from one of the two Robotic Work Stations (RWS) located in either the Cupola or the Destiny module on the ISS. While operating the arm from the RWS, astronauts commonly lock one of the shoulder joints, which allows for more predictable movement of the arm. The shoulder roll is the most commonly locked joint during training and operation of the arm [4]. For this reason, we will consider the shoulder roll joint (the first joint of the arm) to be locked at a fixed angle for the majority of this report. We will begin by defining the D-H parameters and the transformation matrices for each joint, solve for the inverse kinematics, and then work through two different Jacobians.

Figure 1: Orbital ATK's Cygnus cargo spacecraft is captured using the Canadarm2 robotic arm on the International Space Station [3].

2 Finite Kinematic Analysis

2.1 Denavit-Hartenberg Parameters

The Denavit-Hartenberg parameters form a minimal representation of a kinematic linkage of a robotic arm. These four parameters are the joint angle, θ, the link twist angle, α, the link length, a, and the joint offset, d. These parameters are identified by inspection, and are based off the coordinates from and lengths defined in Figure 2. The resulting D-H parameters are presented in Table 1. The parameters are plugged into the generic D-H transformation matrix, see Equation 1. This equation transforms positions and rotations from the i^{th} to the $i+1^{th}$ coordinates frames.

$$T_{i,i+1} = \begin{bmatrix} \cos(\phi) & -\cos(\alpha)\sin(\phi) & \sin(\alpha)\sin(\phi) & a\cos(\phi) \\ \sin(\phi) & \cos(\alpha)\cos(\phi) & -\sin(\alpha)\cos(\phi) & a\sin(\phi) \\ 0 & \sin(\alpha) & \cos(\alpha) & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

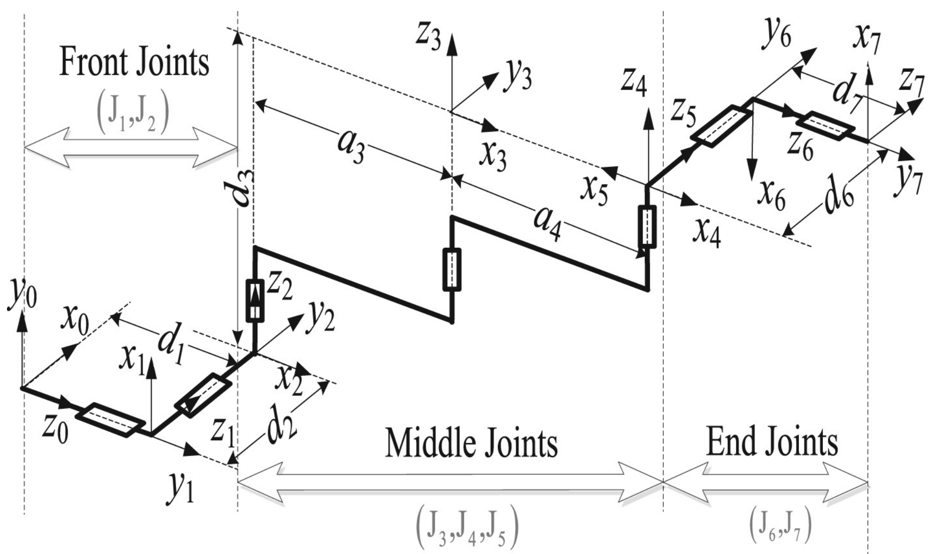

Figure 2: The Denavit-Hartenberg (D-H) parameters for the Space Station Remote Manipulator System (SSRMS) [6].

i	θ_i	α_i	a_i	d_i
1	90	90	0	d_1
2	90	90	0	d_2
3	0	0	a_3	d_3
4	0	0	a_4	0
5	180	90	0	0
6	-90	90	0	d_6
7	180	90	0	d_7

Table 1: The Denavit-Hartenberg parameters for the SSRMS. These parameters are the joint angle, θ, the link twist angle, α, the link length, a, and the joint offset, d. These θ_is give the initial or "zero-displacement" configuration, but each θ_i is modeled as an individual variable below.

The resulting seven matrices are therefore

$$T_{01} = \begin{bmatrix} \cos(\theta_1) & 0 & \sin(\theta_1) & 0 \\ \sin(\theta_1) & 0 & -\cos(\theta_1) & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_{12} = \begin{bmatrix} \cos(\theta_2) & 0 & \sin(\theta_2) & 0 \\ \sin(\theta_2) & 0 & -\cos(\theta_2) & 0 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{23} = \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & a_3\cos(\theta_3) \\ \sin(\theta_3) & \cos(\theta_3) & 0 & a_3\sin(\theta_3) \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_{34} = \begin{bmatrix} \cos(\theta_4) & -\sin(\theta_4) & 0 & a_4\cos(\theta_4) \\ \sin(\theta_4) & \cos(\theta_4) & 0 & a_4\sin(\theta_4) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{45} = \begin{bmatrix} \cos(\theta_5) & 0 & \sin(\theta_5) & 0 \\ \sin(\theta_5) & 0 & -\cos(\theta_5) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_{56} = \begin{bmatrix} \cos(\theta_6) & 0 & \sin(\theta_6) & 0 \\ \sin(\theta_6) & 0 & -\cos(\theta_6) & 0 \\ 0 & 1 & 0 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{67} = \begin{bmatrix} \cos(\theta_7) & 0 & \sin(\theta_7) & 0 \\ \sin(\theta_7) & 0 & -\cos(\theta_7) & 0 \\ 0 & 1 & 0 & d_7 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.1.1 Direct Kinematics

Once these seven matrices are defined, it is often desirable to be able to translate directly from the initial coordinate frame to the final end effector frame. This is easily found by multiplying the successive matrices together to form T_{07}, see

Equation 2. Multiplying these matrices together yields

$$
\begin{aligned}
T_{07} =& T_{01}T_{12}T_{23}T_{34}T_{45}T_{56}T_{67} \qquad (2)\\
T_{07}[1,1] = u_x =& \left(-s_1c_{345} + s_{345}c_1c_2\right)s_7 + \left(s_1s_{345}c_6 + s_2s_6c_1 + c_1c_2c_6c_{345}\right)c_7\\
T_{07}[2,1] = u_y =& \left(s_1s_{345}c_2 + c_1c_{345}\right)s_7 + \left(s_1s_2s_6 + s_1c_2c_6c_{345} - s_{345}c_1c_6\right)c_7\\
T_{07}[3,1] = u_z =& \left(s_2c_6c_{345} - s_6c_2\right)c_7 + s_2s_7s_{345}\\
T_{07}[4,1] = 0 =& 0\\
T_{07}[1,2] = v_x =& s_1s_6s_{345} - s_2c_1c_6 + s_6c_1c_2c_{345}\\
T_{07}[2,2] = v_y =& -s_1s_2c_6 + s_1s_6c_2c_{345} - s_6s_{345}c_1\\
T_{07}[3,2] = v_z =& s_2s_6c_{345} + c_2c_6\\
T_{07}[4,2] = 0 =& 0\\
T_{07}[2,3] = w_x =& -\left(s_1s_{345}c_2 + c_1c_{345}\right)c_7 + \left(s_1s_2s_6 + s_1c_2c_6c_{345} - s_{345}c_1c_6\right)s_7\\
T_{07}[1,3] = w_y =& \left(s_1c_{345} - s_{345}c_1c_2\right)c_7 + \left(s_1s_{345}c_6 + s_2s_6c_1 + c_1c_2c_6c_{345}\right)s_7\\
T_{07}[3,3] = w_z =& \left(s_2c_6c_{345} - s_6c_2\right)s_7 - s_2s_{345}c_7\\
T_{07}[4,3] = 0 =& 0\\
T_{07}[1,4] = p_x =& a_3s_1s_3 + a_3c_1c_2c_3 + a_4s_1s_{34} + a_4c_1c_2c_{34} + d_2s_1 + d_3s_2c_1 - d_6s_1c_{345} + d_6s_{345}c_1c_2\\
&+ d_7s_1s_6s_{345} - d_7s_2c_1c_6 + d_7s_6c_1c_2c_{345}\\
T_{07}[2,4] = p_y =& a_3s_1c_2c_3 - a_3s_3c_1 + a_4s_1c_2c_{34} - a_4s_{34}c_1 - d_2c_1 + d_3s_1s_2 + d_6s_1s_{345}c_2 + d_6c_1c_{345}\\
&- d_7s_1s_2c_6 + d_7s_1s_6c_2c_{345} - d_7s_6s_{345}c_1\\
T_{07}[3,4] = p_z =& a_3s_2c_3 + a_4s_2c_{34} + d_1 - d_3c_2 + d_6s_2s_{345} + d_7s_2s_6c_{345} + d_7c_2c_6\\
T_{07}[4,4] = 1 =& 1
\end{aligned}
$$

2.2 Joint/Shape Matrices

We can similarly use joint and shape matrices to arrive at these T matrices. Shape matrices allow for a more general approach compared to D-H matrices, and "avoid the difficulties that sometimes arise in the use of D-H matrices" [5]. For easier readability, we relabel the joints from $1-7$ to $A-G$. All of the joints of the SSRMS are revolute. A general revolute joint, h, can be modeled with the joint matrix of

$$\Phi_h\left(\phi_h\right) = \begin{bmatrix} \cos\left(\phi_h\right) & -\sin\left(\phi_h\right) & 0 & 0\\ \sin\left(\phi_h\right) & \cos\left(\phi_h\right) & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We can construct the seven T matrices by

$$\begin{aligned}
T_{i,i+1} &= S_{i,j} J_j S_{i+1,j}^{-1} \\
T_{01} &= S_{0A} J_A S_{1A}^{-1} \\
T_{12} &= S_{1B} J_B S_{2B}^{-1} \\
T_{23} &= S_{2C} J_C S_{3C}^{-1} \\
T_{34} &= S_{3D} J_D S_{4D}^{-1} \\
T_{45} &= S_{4E} J_E S_{5E}^{-1} \\
T_{56} &= S_{5F} J_F S_{6F}^{-1} \\
T_{67} &= S_{6G} J_G S_{7G}^{-1}
\end{aligned} \tag{3}$$

For joints $J_A, J_B, J_C, J_D, J_E, J_F$, and J_G, we also define two shape matrices.

$$S_{0A} = I, S_{1A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -d_1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$S_{1B} = I, S_{2B} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -d_2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad S_{2C} = I, S_{3C} = \begin{bmatrix} 1 & 0 & 0 & -a_3 \\ 0 & 0 & 1 & -d_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$S_{3D} = I, S_{4D} = \begin{bmatrix} 1 & 0 & 0 & -a_3 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad S_{4E} = I, S_{5E} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$S_{5F} = I, S_{6F} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -d_6 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad S_{6G} = I, S_{7G} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -d_7 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplying out these shape and joint matrices according toe Equation 3 yields the same T matrices as obtained using the Denavit-Hartenberg parameters.

2.3 Inverse Kinematics

2.3.1 Method

The SSRMS has eight solutions of joint angles for any such given pose [6]. In their 2014 paper, Xu et al. show how to to solve for these configurations using a series of flags labeled "SHOULDER", "WRIST", and "ELBOW". After locking the first rotary joint at a known angle, there are a pair of solutions for the second joint known as the "SHOULDER" configuration. With the first two joints known, it is then possible to solve for the final two joints, giving the "WRIST" configuration. Finally, the middle three joints can be solved for, giving the "ELBOW" configuration.

This technique was inspired by a 1984 paper by Lee and Ziegler which used this geometric approach to solve for the inverse kinematics of PUMA robots [2]. In their paper, Lee and Ziegler define 4 "indicators" ("ARM", "ELBOW", "WRIST", and 'FLIP") based off the geometric configuration of the PUMA robot. These indicators were used to provide consistent solutions for the PUMA robots during movement through their workspace. Lee and Ziegler presented an algorithmic approach which was programmed to show how their method could be used in practice. An example of two of these indicators in shown in Figure 3.

For the SSRMS solution presented below, the SHOULDER, ELBOW, and WRIST configuration indicators take on the follow values [6]

$$\text{SHOULDER} = \begin{cases} +1, & \text{right shoulder.} \\ -1, & \text{left shoulder.} \end{cases}$$

$$\text{ELBOW} = \begin{cases} +1, & \text{outside elbow.} \\ -1, & \text{inside elbow.} \end{cases}$$

$$\text{WRIST} = \begin{cases} +1, & \text{wrist down.} \\ -1, & \text{wrist up.} \end{cases}$$

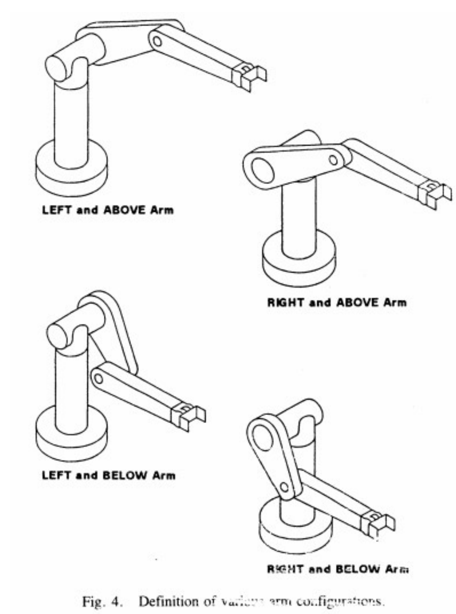

Figure 3: Definition of two of the PUMA robotic arm configurations, taken from Lee and Ziegler [2].

2.3.2 SSRMS Solution

In general, for a known end effector pose, we can define [1]

$$\begin{aligned} T_{07} &= \begin{bmatrix} u_x & v_x & w_x & p_x \\ u_y & v_y & w_y & p_y \\ u_z & v_z & w_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= T_{01}T_{12}T_{23}T_{34}T_{45}T_{56}T_{67} \end{aligned}$$

Premultiplying both sides by T_{01}^{-1} yields,

$$T_{01}^{-1}T_{07} = T_{12}T_{23}T_{34}T_{45}T_{56}T_{67}$$

Equating each element (i, j) on both the left and right hand sides yields:

$$u_x c_1 + u_y s_1 = (s_2 s_6 + c_2 c_6 c_{345})\, c_7 + s_7 s_{345} c_2 \tag{4}$$
$$v_x c_1 + v_y s_1 = -s_2 c_6 + s_6 c_2 c_{345} \tag{5}$$
$$w_x c_1 + w_y s_1 = (s_2 s_6 + c_2 c_6 c_{345})\, s_7 - s_{345} c_2 c_7 \tag{6}$$
$$p_x c_1 + p_y s_1 = a_3 c_2 c_3 + a_4 c_2 c_{34} + d_3 s_2 + d_6 s_{345} c_2 - d_7 s_2 c_6 + d_7 s_6 c_2 c_{345} \tag{7}$$
$$u_z = (s_2 c_6 c_{345} - s_6 c_2)\, c_7 + s_2 s_7 s_{345} \tag{8}$$
$$v_z = s_2 s_6 c_{345} + c_2 c_6 \tag{9}$$
$$w_z = (s_2 c_6 c_{345} - s_6 c_2)\, s_7 - s_2 s_{345} c_7 \tag{10}$$
$$-d_1 + p_z = a_3 s_2 c_3 + a_4 s_2 c_{34} - d_3 c_2 + d_6 s_2 s_{345} + d_7 s_2 s_6 c_{345} + d_7 c_2 c_6 \tag{11}$$
$$u_x s_1 - u_y c_1 = -s_7 c_{345} + s_{345} c_6 c_7 \tag{12}$$
$$v_x s_1 - v_y c_1 = s_6 s_{345} \tag{13}$$
$$w_x s_1 - w_y c_1 = s_7 s_{345} c_6 + c_7 c_{345} \tag{14}$$
$$p_x s_1 - p_y c_1 = a_3 s_3 + a_4 s_{34} + d_2 - d_6 c_{345} + d_7 s_6 s_{345} \tag{15}$$
$$0 = 0 \tag{16}$$
$$0 = 0 \tag{17}$$
$$0 = 0 \tag{18}$$
$$1 = 1 \tag{19}$$

where we have defined $s_i = \sin i, c_i = \cos i, s_{ij} = \sin(i+j), c_{ij} = \cos(i+j), s_{ijk} = \sin(i+j+k)$ and $c_{ijk} = \cos(i+j+k)$. Manipulating the equations, we take $(Eq.\ 4)\, s_2 - (Eq.\ 8)\, c_2$ and simplify, producing

$$\big(u_x c_1 + u_y s_1\big)\, s_2 - u_z c_2 = s_6 c_7 \tag{20}$$

Similarly, we can do $(Eq.\ 7)\, s_2 - (Eq.\ 11)\, c_2$ and simplify, which results in

$$\big(p_x c_1 + p_y s_1\big)\, s_2 - (-d_1 + p_z)\, c_2 = d_3 - c_6 d_7 \tag{21}$$

We can also subtract $(Eq.\ 9)\, c_2 - (Eq.\ 5)\, s_2$

$$v_z c_2 - \big(v_x c_1 + v_y s_1\big)\, s_2 = c_6 \tag{22}$$

Finally, we can also subtract $(Eq.\ 6)\, s_2 - (Eq.\ 10)\, c_2$

$$\left(w_x c_1 + w_y s_1\right) s_2 - w_z c_2 = s_6 s_7 \tag{23}$$

Rearranging Equations 21 and 22 to be equal to c_6 and equating the two yields

$$-d_3 = \Big(\left(v_x d_7 - p_x\right) c_1 + \left(v_y d_7 - p_y\right) s_1\Big)\, s_2 + \left(-v_z d_7 - d_1 + p_z\right) c_2 \tag{24}$$

Locking the shoulder roll angle to a known angle, $\boxed{\theta_1 = \beta}$, we can solve for θ_2,

$$\boxed{\theta_2 = \text{SHOULDER} \cdot \operatorname{acos}\left(\frac{d_3}{\sqrt{h_1^2 + q_1^2}}\right) + \operatorname{atan2}(q_1, h_1)} \tag{25}$$

where

$$h_1 = \left(-v_z d_7 - d_1 + p_z\right) \tag{26}$$

$$q_1 = \Big(\left(v_x d_7 - p_x\right) c_\beta + \left(v_y d_7 - p_y\right) s_\beta\Big) \tag{27}$$

With θ_1 and θ_2 now known, θ_6 can be solved using Equation 22,

$$\boxed{\theta_6 = \text{WRIST} \cdot \operatorname{acos}\Big(v_z c_2 - \left(v_x c_1 + v_y s_1\right) s_2\Big)} \tag{28}$$

And we can then combine Equations 20 and 23, yielding

$$\boxed{\theta_7 = \operatorname{atan2}\left(\frac{\left(u_x c_1 + u_y s_1\right) s_2 - u_z c_2}{s_6}, \frac{\left(w_x c_1 + w_y s_1\right) s_2 - w_z c_2}{s_6}\right)} \tag{29}$$

With the shoulder and wrist joints resolved, we can now solve for the middle joints. We now take

$$\left(T_{12}^{-1}\right)\left(T_{17}\right)\left(T_{67}^{-1}\right)\left(T_{56}^{-1}\right) = \left(T_{23}\right)\left(T_{34}\right)\left(T_{45}\right)$$

Taking the left and right hand side $(1,4)$ and $(2,4)$ elements from the resulting matrix yields

$$\begin{aligned} a_3 c_3 + a_4 c_{34} = {} & d_6 \Big(w_z s_2 + c_2 \left(w_x c_1 + w_y s_1\right)\Big) c_7 \\ & - d_6 \Big(u_z s_2 + c_2 \left(u_x c_1 + u_y s_1\right)\Big) s_7 \\ & - d_7 \Big(v_z s_2 + c_2 \left(v_x c_1 + v_y s_1\right)\Big) \\ & + \left(-d_1 + p_z\right) s_2 + c_2 \left(p_x c_1 + p_y s_1\right) \end{aligned} \tag{30}$$

$$\begin{aligned} a_3 s_3 + a_4 s_{34} = {} & -d_2 + d_6 \left(w_x s_1 - w_y c_1\right) c_7 - d_6 \left(u_x s_1 - u_y c_1\right) s_7 \\ & - d_7 \left(v_x s_1 - v_y c_1\right) + p_x s_1 - p_y c_1 \end{aligned} \tag{31}$$

θ_4 is then solved by combining the above two equations, resulting in

$$\boxed{\theta_4 = \text{ELBOW} \cdot \text{acos}\left(\frac{X^2 + Y^2 - a_3^2 - a_4^2}{2a_3a_4}\right)} \tag{32}$$

where

$$\begin{aligned}
X &= d_6\left(\left(w_z s_2 + c_2\left(w_x c_1 + w_y s_1\right)\right)c_7 - \left(u_z s_2 + c_2\left(u_x c_1 + u_y s_1\right)\right)s_7\right) \\
&\quad - d_7\left(v_z s_2 + c_2\left(v_x c_1 + v_y s_1\right)\right) + \left(-d_1 + p_z\right)s_2 + c_2\left(p_x c_1 + p_y s_1\right) \\
Y &= -d_2 + d_6\left(w_x s_1 - w_y c_1\right)c_7 - d_6\left(u_x s_1 - u_y c_1\right)s_7 - d_7\left(v_x s_1 - v_y c_1\right) + p_x s_1 - p_y c_1
\end{aligned}$$

Substituting the solution into θ_4 and Equations 30 and 31 and combining yields

$$\boxed{\theta_3 = \text{atan2}\left(Y\left(a_3 + a_4c_4\right) - Xa_4s_4, X\left(a_3 + a_4c_4\right) + Ya_4s_4\right)}$$

Subtracting $(Eq.\ 14)c_7$ and $(Eq.\ 12)s_7$ yields

$$c_{345} = \left(w_x s_1 - w_y c_1\right)c_7 - \left(u_x s_1 - u_y c_1\right)s_7$$

And from Equation 13 we have

$$s_{345} = \frac{v_x s_1 - v_y c_1}{s_6}$$

which we can combine to solve for the last joint

$$\begin{gathered}
\theta_5 = \left(\theta_3 + \theta_4 + \theta_5\right) - \left(\theta_3 + \theta_4\right) \\
\boxed{\theta_5 = \text{atan2}\left(s_{345}, c_{345}\right) - \left(\theta_3 + \theta_4\right)}
\end{gathered}$$

2.4 Numerical Example

For practical purposes, the link length and offset values can be set to

$$a_3 = 2.30, a_4 = 2.30, d_1 = 0.65, d_2 = 0.30, d_3 = 0.90, d_6 = 0.30, d_7 = 0.65$$

Note that $a_3 = a_4, d_1 = d_7,$ and $d_2 = d_6$, as the arm is symmetric. As an example, plugging in these values and the initial angles given in Table 1 into Equation 2 yields

$$T_{07} = \begin{bmatrix} 0 & 0 & 1 & 0.6 \\ 1 & 0 & 0 & 0.9 \\ 0 & 1 & 0 & 5.9 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

As another example, given the end effector pose

$$T_{07} = \begin{bmatrix} 0.8021 & 0.1217 & 0.5846 & 2.4790 \\ -0.5859 & 0.3495 & 0.7311 & -2.4734 \\ -0.1154 & 0.9290 & 0.3517 & -0.4927 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and locking the first joint variable $\theta_1 = \beta = 60°$, we can solve for the 8 possible configurations of the arm, see Table 2. The equations used to solve for these values are taken from above, and the code used to generate these results are in the Appendex.

S	E	W	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7
1	1	1	60.000	-20.268	64.074	79.722	-149.770	138.205	-77.426
1	1	-1	60.000	-20.268	58.153	99.444	16.428	-138.205	102.573
1	-1	1	60.000	-20.268	143.797	-79.722	-70.048	138.205	-77.426
-1	1	1	60.000	-109.087	35.576	79.140	-119.938	49.659	-85.275
1	-1	-1	60.000	-20.268	157.598	-99.444	115.872	-138.205	102.573
-1	1	-1	60.000	-109.087	23.189	100.025	51.565	-49.659	94.724
-1	-1	1	60.000	-109.087	114.717	-79.140	-40.797	49.659	-85.275
-1	-1	-1	60.000	-109.087	123.214	-100.025	151.590	-49.659	94.724

Table 2: The eight possible configurations for locking the first joint, where S, E, and W stand for "SHOULDER", "ELBOW", and "WRIST", respectively.

3 Differential Kinematic Analysis

3.1 Method 1: Kinematic Jacobian

The first Jacobian is based of a kinematic approach. Where $\hat{z}_i$ is taken from the last column of T_{0i}, and can be defined [1]

$$T_{0i} = \begin{bmatrix} & \underline{\underline{\Theta}}_i & & \vdots & a_i \\ & \cdots & & & \cdots \\ 0 & 0 & 0 & \vdots & 1 \end{bmatrix}$$

$$\Theta_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}$$

$$\hat{z}_i = \left(\prod_{i=0}^{n} \Theta_i \right) z_i$$

and $\vec{r}_i$ is defined

$$\vec{r}_i = \sum_{i=0}^{n} \vec{a}_i$$

With these definitions, we can find the Jacobian via

$$\begin{aligned}
\dot{\vec{P}} &= \sum_{i=0}^{n} \left(\hat{z}_i \times \vec{r}_i\right) \dot{\theta}_i \\
\vec{w} &= \sum_{i=0}^{n} \dot{\theta}_i \hat{z}_i \\
J_K \dot{q} &= \begin{bmatrix} \underline{\dot{P}} \\ \underline{\vec{w}} \end{bmatrix} \\
\begin{bmatrix} \hat{z}_1 \times \vec{r}_1 & \hat{z}_2 \times \vec{r}_2 & \cdots & \hat{z}_7 \times \vec{r}_7 \\ \hat{z}_1 & \hat{z}_2 & \cdots & \hat{z}_7 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \vdots \\ \dot{\theta}_7 \end{bmatrix} &= \begin{bmatrix} \underline{\dot{P}}_{EE} \\ \underline{w}_{EE} \end{bmatrix}
\end{aligned}$$

The results of these equations yield in the kinematic Jacobian,

$$\begin{aligned}
J_{11} &= -a_3 s_1 c_2 c_3 + a_3 s_3 c_1 - a_4 s_1 c_2 c_{34} + a_4 s_{34} c_1 + d_2 c_1 - d_3 s_1 s_2 \\
&\quad - d_6 s_1 s_{345} c_2 - d_6 c_1 c_{345} + d_7 s_1 s_2 c_6 - d_7 s_1 s_6 c_2 c_{345} + d_7 s_6 s_{345} c_1 \\
J_{21} &= a_3 s_1 s_3 + a_3 c_1 c_2 c_3 + a_4 s_1 s_{34} + a_4 c_1 c_2 c_{34} + d_2 s_1 + d_3 s_2 c_1 \\
&\quad - d_6 s_1 c_{345} + d_6 s_{345} c_1 c_2 + d_7 s_1 s_6 s_{345} - d_7 s_2 c_1 c_6 + d_7 s_6 c_1 c_2 c_{345} \\
J_{31} &= 0 \\
J_{41} &= 0 \\
J_{51} &= 0 \\
J_{61} &= 1
\end{aligned}$$

$$\begin{aligned}
J_{12} &= -\left(a_3 s_2 c_3 + a_4 s_2 c_{34} - d_3 c_2 + d_6 s_2 s_{345} + d_7 s_2 s_6 c_{345} + d_7 c_2 c_6\right) c_1 \\
J_{22} &= -\left(a_3 s_2 c_3 + a_4 s_2 c_{34} - d_3 c_2 + d_6 s_2 s_{345} + d_7 s_2 s_6 c_{345} + d_7 c_2 c_6\right) s_1 \\
J_{32} &= a_3 c_2 c_3 + a_4 c_2 c_{34} + d_3 s_2 + d_6 s_{345} c_2 - d_7 s_2 c_6 + d_7 s_6 c_2 c_{345} \\
J_{42} &= s_1 \\
J_{52} &= -c_1 \\
J_{62} &= 0
\end{aligned}$$

$$
\begin{aligned}
J_{13} &= a_3 s_1 c_3 - a_3 s_3 c_1 c_2 + a_4 s_1 c_{34} - a_4 s_{34} c_1 c_2 + d_6 s_1 s_{345} \\
&\quad + d_6 c_1 c_2 c_{345} + d_7 s_1 s_6 c_{345} - d_7 s_6 s_{345} c_1 c_2 \\
J_{23} &= -a_3 s_1 s_3 c_2 - a_3 c_1 c_3 - a_4 s_1 s_{34} c_2 - a_4 c_1 c_{34} + d_6 s_1 c_2 c_{345} \\
&\quad - d_6 s_{345} c_1 - d_7 s_1 s_6 s_{345} c_2 - d_7 s_6 c_1 c_{345} \\
J_{33} &= (-a_3 s_3 - a_4 s_{34} + d_6 c_{345} - d_7 s_6 s_{345})\, s_2 \\
J_{43} &= s_2 c_1 \\
J_{53} &= s_1 s_2 \\
J_{63} &= -c_2
\end{aligned}
$$

$$
\begin{aligned}
J_{14} &= a_4 s_1 c_{34} - a_4 s_{34} c_1 c_2 + d_6 s_1 s_{345} + d_6 c_1 c_2 c_{345} + d_7 s_1 s_6 c_{345} - d_7 s_6 s_{345} c_1 c_2 \\
J_{24} &= -a_4 s_1 s_{34} c_2 - a_4 c_1 c_{34} + d_6 s_1 c_2 c_{345} - d_6 s_{345} c_1 - d_7 s_1 s_6 s_{345} c_2 - d_7 s_6 c_1 c_{345} \\
J_{34} &= (-a_4 s_{34} + d_6 c_{345} - d_7 s_6 s_{345})\, s_2 \\
J_{44} &= s_2 c_1 \\
J_{54} &= s_1 s_2 \\
J_{64} &= -c_2
\end{aligned}
$$

$$
\begin{aligned}
J_{15} &= d_6 s_1 s_{345} + d_6 c_1 c_2 c_{345} + d_7 s_1 s_6 c_{345} - d_7 s_6 s_{345} c_1 c_2 \\
J_{25} &= d_6 s_1 c_2 c_{345} - d_6 s_{345} c_1 - d_7 s_1 s_6 s_{345} c_2 - d_7 s_6 c_1 c_{345} \\
J_{35} &= (d_6 c_{345} - d_7 s_6 s_{345})\, s_2 \\
J_{45} &= s_2 c_1 \\
J_{55} &= s_1 s_2 \\
J_{65} &= -c_2
\end{aligned}
$$

$$
\begin{aligned}
J_{16} &= d_7 (s_1 s_{345} c_6 + s_2 s_6 c_1 + c_1 c_2 c_6 c_{345}) \\
J_{26} &= d_7 (s_1 s_2 s_6 + s_1 c_2 c_6 c_{345} - s_{345} c_1 c_6) \\
J_{36} &= d_7 (s_2 c_6 c_{345} - s_6 c_2) \\
J_{46} &= -s_1 c_{345} + s_{345} c_1 c_2 \\
J_{56} &= s_1 s_{345} c_2 + c_1 c_{345} \\
J_{66} &= s_2 s_{345}
\end{aligned}
$$

$$
\begin{aligned}
J_{17} &= 0 \\
J_{27} &= 0 \\
J_{37} &= 0 \\
J_{47} &= (s_1 s_{345} + c_1 c_2 c_{345})\, s_6 - s_2 c_1 c_6 \\
J_{57} &= (s_1 c_2 c_{345} - s_{345} c_1)\, s_6 - s_1 s_2 c_6 \\
J_{67} &= s_2 s_6 c_{345} + c_2 c_6
\end{aligned}
$$

3.2 Method 2: Geometric Jacobian

The geometric Jacobian is formed from a linearized form of the kinematic equations [1]. We first form our ${}^iD$ matrices from

$$ {}^iD = T_{i-1} Q_i T_{i-1}^{-1} $$

And $T_0 = I, T_i = \prod_0^n A_n$, where A_n are the matrices formed by the D-H parameters. Where, as all our joints are revolute, the derivative operator matrix is

$$ Q = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} $$

Selecting elements from these ${}^iD$ matrices, we form the Jacobian via

$$ J_G = \begin{bmatrix} {}^0D_{14} & {}^1D_{14} & {}^2D_{14} & {}^3D_{14} & {}^4D_{14} & {}^5D_{14} & {}^6D_{14} \\ {}^0D_{24} & {}^1D_{24} & {}^2D_{24} & {}^3D_{24} & {}^4D_{24} & {}^5D_{24} & {}^6D_{24} \\ {}^0D_{34} & {}^1D_{34} & {}^2D_{34} & {}^3D_{34} & {}^4D_{34} & {}^5D_{34} & {}^6D_{34} \\ {}^0D_{32} & {}^1D_{32} & {}^2D_{32} & {}^3D_{32} & {}^4D_{32} & {}^5D_{32} & {}^6D_{32} \\ {}^0D_{13} & {}^1D_{13} & {}^2D_{13} & {}^3D_{13} & {}^4D_{13} & {}^5D_{13} & {}^6D_{13} \\ {}^0D_{21} & {}^1D_{21} & {}^2D_{21} & {}^3D_{21} & {}^4D_{21} & {}^5D_{21} & {}^6D_{21} \end{bmatrix} $$

And

$$ \dot{x} = J_G \dot{q} $$

$$ \begin{bmatrix} \mu_x \\ \mu_y \\ \mu_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} {}^0D_{14} & {}^1D_{14} & {}^2D_{14} & {}^3D_{14} & {}^4D_{14} & {}^5D_{14} & {}^6D_{14} \\ {}^0D_{24} & {}^1D_{24} & {}^2D_{24} & {}^3D_{24} & {}^4D_{24} & {}^5D_{24} & {}^6D_{24} \\ {}^0D_{34} & {}^1D_{34} & {}^2D_{34} & {}^3D_{34} & {}^4D_{34} & {}^5D_{34} & {}^6D_{34} \\ {}^0D_{32} & {}^1D_{32} & {}^2D_{32} & {}^3D_{32} & {}^4D_{32} & {}^5D_{32} & {}^6D_{32} \\ {}^0D_{13} & {}^1D_{13} & {}^2D_{13} & {}^3D_{13} & {}^4D_{13} & {}^5D_{13} & {}^6D_{13} \\ {}^0D_{21} & {}^1D_{21} & {}^2D_{21} & {}^3D_{21} & {}^4D_{21} & {}^5D_{21} & {}^6D_{21} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \\ \dot{\theta}_5 \\ \dot{\theta}_6 \\ \dot{\theta}_7 \end{bmatrix} $$

Solving for these ${}^iD$ matrices and selecting the identified elements results in

$$ \begin{aligned} J_{11} &= 0 & J_{12} &= d_1 c_1 & J_{13} &= & -d_1 s_1 s_2 + d_2 c_1 c_2 \\ J_{21} &= 0 & J_{22} &= d_1 s_1 & J_{23} &= & d_1 s_2 c_1 + d_2 s_1 c_2 \\ J_{31} &= 0 & J_{32} &= 0 & J_{33} &= & d_2 s_2 \\ J_{41} &= 0 & J_{42} &= s_1 & J_{43} &= & s_2 c_1 \\ J_{51} &= 0 & J_{52} &= -c_1 & J_{53} &= & s_1 s_2 \\ J_{61} &= 1 & J_{62} &= 0 & J_{63} &= & -c_2 \end{aligned} $$

$$J_{14} = -a_3 s_1 c_3 + a_3 s_3 c_1 c_2 - d_1 s_1 s_2 + d_2 c_1 c_2$$
$$J_{24} = a_3 s_1 s_3 c_2 + a_3 c_1 c_3 + d_1 s_2 c_1 + d_2 s_1 c_2$$
$$J_{34} = (a_3 s_3 + d_2)\, s_2$$
$$J_{44} = s_2 c_1$$
$$J_{54} = s_1 s_2$$
$$J_{64} = -c_2$$

$$J_{15} = -a_3 s_1 c_3 + a_3 s_3 c_1 c_2 - a_4 s_1 c_{34} + a_4 s_{34} c_1 c_2 - d_1 s_1 s_2 + d_2 c_1 c_2$$
$$J_{25} = a_3 s_1 s_3 c_2 + a_3 c_1 c_3 + a_4 s_1 s_{34} c_2 + a_4 c_1 c_{34} + d_1 s_2 c_1 + d_2 s_1 c_2$$
$$J_{35} = (a_3 s_3 + a_4 s_{34} + d_2)\, s_2$$
$$J_{45} = s_2 c_1$$
$$J_{55} = s_1 s_2$$
$$J_{65} = -c_2$$

$$J_{16} = -(d_1 c_2 - d_3)(s_1 s_{345} + c_1 c_2 c_{345}) - (a_3 c_{45} + a_4 c_5 + d_1 s_2 c_{345} + d_2 s_{345})\, s_2 c_1$$
$$J_{26} = -(d_1 c_2 - d_3)(s_1 c_2 c_{345} - s_{345} c_1) - (a_3 c_{45} + a_4 c_5 + d_1 s_2 c_{345} + d_2 s_{345})\, s_1 s_2$$
$$J_{36} = a_3 c_2 c_{45} + a_4 c_2 c_5 + d_2 s_{345} c_2 + d_3 s_2 c_{345}$$
$$J_{46} = -s_1 c_{345} + s_{345} c_1 c_2$$
$$J_{56} = s_1 s_{345} c_2 + c_1 c_{345}$$
$$J_{66} = s_2 s_{345}$$

$$\begin{aligned}
J_{17} &= \big((s_1 s_{345} + c_1 c_2 c_{345})\, c_6 + s_2 s_6 c_1\big)(a_3 s_{45} + a_4 s_5 + d_1 s_2 s_{345} - d_2 c_{345} + d_6) \\
&\quad + (s_1 c_{345} - s_{345} c_1 c_2)(a_3 c_6 c_{45} + a_4 c_5 c_6 + d_1 s_2 c_6 c_{345} - d_1 s_6 c_2 + d_2 s_{345} c_6 + d_3 s_6) \\
J_{27} &= \big((s_1 c_2 c_{345} - s_{345} c_1)\, c_6 + s_1 s_2 s_6\big)(a_3 s_{45} + a_4 s_5 + d_1 s_2 s_{345} - d_2 c_{345} + d_6) \\
&\quad - (s_1 s_{345} c_2 + c_1 c_{345})(a_3 c_6 c_{45} + a_4 c_5 c_6 + d_1 s_2 c_6 c_{345} - d_1 s_6 c_2 + d_2 s_{345} c_6 + d_3 s_6) \\
J_{37} &= (s_2 c_6 c_{345} - s_6 c_2)(a_3 s_{45} + a_4 s_5 + d_1 s_2 s_{345} - d_2 c_{345} + d_6) - \\
&\quad (a_3 c_6 c_{45} + a_4 c_5 c_6 + d_1 s_2 c_6 c_{345} - d_1 s_6 c_2 + d_2 s_{345} c_6 + d_3 s_6)\, s_2 s_{345} \\
J_{47} &= s_1 s_6 s_{345} - s_2 c_1 c_6 + s_6 c_1 c_2 c_{345} \\
J_{57} &= -s_1 s_2 c_6 + s_1 s_6 c_2 c_{345} - s_6 s_{345} c_1 \\
J_{67} &= s_2 s_6 c_{345} + c_2 c_6
\end{aligned}$$

4 Conclusions

In this report we have completed a kinematic analysis of the Space Station Robotic Manipulator with a locked shoulder roll joint. Transformation matrices were found using both Denavit-Hartenberg parameters and shape matrices. The direct kinematic solution was identified using the seven transformation matrices. The inverse kinematic solution found by Xu et al. was verified and several inaccuracies were fixed [6]. Using the resulting inverse kinematic solution along with three configuration indicators, a Python program was written to generate eight solutions for an example pose, matching the solution found by Xu et al.

A differential kinematic analysis was conducted, resulting in two Jacobians. The kinematic Jacobian, J_K, was generated using $\hat{z}$ and $\vec{r}$ derived from the Denavit-Hartenberg matrices. A geometric Jacobian, J_G, was calculated using a different approach involving derivative matrices. A symbolic Python program was written to calculate and compare the numerical efficiency of these two Jacobians. J_K required a total of 4833 operations per calculation, or

```
269*ADD + 1210*COS + 1754*MUL + 100*NEG + 1106*SIN + 394*SUB
```

while J_G required a total of 1138 operations per calculation, or

```
207*ADD + 225*COS + 351*MUL + 33*NEG + 4*POW + 207*SIN + 111*SUB
```

Based off this simple analysis, it appears that the geometric Jacobian technique produced a more efficient Jacobian in this case.

References

[1] John Karasinski. Lecture notes in Spatial Kinematics from Professor Bahram Ravani, 2018.

[2] CSG Lee and M Ziegler. Geometric approach in solving inverse kinematics of puma robots. *IEEE Transactions on Aerospace and Electronic Systems*, (6):695–706, 1984.

[3] NASA. "Orbital ATK's Cygnus cargo spacecraft is captured using the Candarm2 robotic arm". 10/23/2016, CC BY-NC-ND 2.0.

[4] Stephen Robinson and Terry Virts. "private communication", 2018.

[5] John J Uicker, Bahram Ravani, and Pradip N Sheth. *Matrix methods in the design analysis of mechanisms and multibody systems*. Cambridge University Press, 2013.

[6] Wenfu Xu, Yu She, and Yangsheng Xu. Analytical and semi-analytical inverse kinematics of SSRMS-type manipulators with single joint locked failure. *Acta Astronautica*, 105(1):201–217, 2014.

A Appendix: Code

To confirm that the algorithms written here were correct, I used the Stanford Arm examples in class. I verified that the results produced with the Stanford Arm D-H parameters were correct according the results presented in lecture[1].

```
from sympy import *
from sympy.matrices import *

class Transform:
    def __init__(self, phi, alpha, a, d):
        self.phi = phi
        self.alpha = alpha
        self.a = a
        self.d = d
        self.DHMatrix()

    def DHMatrix(self):
        # phi = rad(self.phi)
        phi = self.phi
        alpha = rad(self.alpha)
        a = self.a
        d = self.d

        dh = Matrix([[cos(phi), -sin(phi)*cos(alpha)  ,sin(phi)*sin(alpha), a*cos(phi)],
                     [sin(phi),  cos(phi)*cos(alpha), -cos(phi)*sin(alpha), a*sin(phi)],
                     [       0,            sin(alpha),           cos(alpha),          d],
                     [       0,                     0,                    0,          1]])
        self.A = dh

        self.R = dh[:3, :3]
        self.vecA = dh[:3, 3]

# Let's do an analysis with the SSRMS with the shoulder roll joint locked at an angle beta
beta, theta1, theta2, theta3, theta4, theta5, theta6, theta7 = symbols("beta theta1 theta2 theta3 theta4 theta5 theta6 theta7")
d1, d2, d3, d6, d7, a3, a4 = symbols('d1 d2 d3 d6 d7 a3 a4')

# SSRMS
configs = [[theta1, 90,  0, d1],
           [theta2, 90,  0, d2],
           [theta3,  0, a3, d3],
           [theta4,  0, a4,  0],
           [theta5, 90,  0,  0],
           [theta6, 90,  0, d6],
           [theta7, 90,  0, d7]]

# Stanford Arm Example from Class
```

[1] Note that the Stanford Arm D-H parameters presented on 3/6 are different than those presented on 3/13. On 3/6 the Stanford Arm had $d_1 = 0$.

```
# configs = [[theta1, -90, 0, d1],
#            [theta2,  90, 0, d2],
#            [   -90,   0, 0, d3],
#            [theta4, -90, 0,  0],
#            [theta5,  90, 0,  0],
#            [theta6,   0, 0,  0]]

# Build our robot
T = []
for config in configs:
    T.append(Transform(*config))

def KJ():
  # Kinematic Jacobian
  # The z_i's need to be WRT the base coordinate system
  z = Matrix([[0], [0], [1]])

  # Calculate all our z vectors
  for i, _ in enumerate(T):
      rot = Identity(3)
      for j in range(i):
          rot *= T[j].R
      T[i].z = rot * z

  # Loop through backwards to calculate the r vectors
  for i, _ in enumerate(T):
      i = len(T) - (i+1)
      rot = Identity(3)
      for j in range(i):
          rot *= T[j].R
      rot *= T[i].vecA
      if (i+1) < len(T):
          rot += T[i+1].r
      T[i].r = rot

  # Calculate the z skew matrices and compute cross products
  for i, _ in enumerate(T):
      T[i].z_skew = Matrix([[          0, -T[i].z[2],  T[i].z[1]],
                            [ T[i].z[2],           0, -T[i].z[0]],
                            [-T[i].z[1],  T[i].z[0],           0]])
      T[i].crossed = T[i].z_skew * T[i].r

  # Combine elements to form Jacobian
  top, bottom = Matrix(), Matrix()
  for i, _ in enumerate(T):
      top = top.row_join(T[i].crossed)
      bottom = bottom.row_join(T[i].z)
  J = top.col_join(bottom)

  return J

def GJ():
  # Geometric Jacobian
  Q_rev = Matrix([[0, -1, 0, 0],
                  [1,  0, 0, 0],
                  [0,  0, 0, 0],
                  [0,  0, 0, 0]])
```

```python

  for i, _ in enumerate(T):
    print('i ', i)
    if i > 0:
      T[i].Ti = T[i-1].Ti * T[i-1].A
    if i == 0:
      T[i].Ti = Matrix(Identity(4))
    T[i].Ti.simplify()

    T[i].D = simplify(T[i].Ti * Q_rev) * simplify(T[i].Ti.inv())

  Jg = Matrix([[T[0].D[0, 3], T[1].D[0, 3], T[2].D[0, 3], T[3].D[0, 3], T[4].D[0, 3], T[5].D[0, 3], T[6].D[0, 3]],
               [T[0].D[1, 3], T[1].D[1, 3], T[2].D[1, 3], T[3].D[1, 3], T[4].D[1, 3], T[5].D[1, 3], T[6].D[1, 3]],
               [T[0].D[2, 3], T[1].D[2, 3], T[2].D[2, 3], T[3].D[2, 3], T[4].D[2, 3], T[5].D[2, 3], T[6].D[2, 3]],
               [T[0].D[2, 1], T[1].D[2, 1], T[2].D[2, 1], T[3].D[2, 1], T[4].D[2, 1], T[5].D[2, 1], T[6].D[2, 1]],
               [T[0].D[0, 2], T[1].D[0, 2], T[2].D[0, 2], T[3].D[0, 2], T[4].D[0, 2], T[5].D[0, 2], T[6].D[0, 2]],
               [T[0].D[1, 0], T[1].D[1, 0], T[2].D[1, 0], T[3].D[1, 0], T[4].D[1, 0], T[5].D[1, 0], T[6].D[1, 0]]])
  return Jg

# Inverse Kinematic Solution
# T07 = T[0].A * T[1].A * T[2].A * T[3].A * T[4].A * T[5].A * T[6].A

# We know that
# u_x, u_y, u_z, v_x, v_y, v_z, w_x, w_y, w_z, p_x, p_y, p_z = symbols('u_x u_y u_z v_x v_y v_z w_x w_y w_z p_x p_y p_z')
# T07 = Matrix([[u_x, v_x, w_x, p_x],
#               [u_y, v_y, w_y, p_y],
#               [u_z, v_z, w_z, p_z],
#               [  0,   0,   0,   1]])

# And setting T01^-1 * T07 = T12 T23 T34 T45 T56 T67
# print(latex(T[0].A.inv() * T07))
# print(latex(T[1].A * T[2].A * T[3].A * T[4].A * T[5].A * T[6].A))

def IK(SHOULDER=1, WRIST=1, ELBOW=1,
       a3=2.3, a4=2.3, d1=0.65, d2=0.3,
       d3=0.9, d6=0.3, d7=0.65,
       beta=rad(60),
       u_x=0.8021, u_y=-0.5859, u_z=-0.1154,
       v_x=0.1217, v_y= 0.3495, v_z=-0.9290,
       w_x=0.5846, w_y= 0.7311, w_z=0.3517,
       p_x=2.4790, p_y=-2.4734, p_z=-0.4927):
  '''
  Return the IK for a given pose, initial locked angle, and arm characteristics.
  '''

  theta1 = beta

  h1 = -v_z * d7 - d1 + p_z
```

```
  q1 = (v_x * d7 - p_x)*cos(beta) + (v_y * d7 - p_y)*sin(beta)

  theta2 = SHOULDER * acos(d3/(sqrt(h1**2 + q1**2))) + atan2(q1, h1) - pi

  theta6 = WRIST * acos(v_z*cos(theta2)-(v_x*cos(theta1) + v_y*sin(theta1))*sin(theta2))

  theta7 = -atan2(((u_x*cos(theta1) + u_y*sin(theta1))*sin(theta2) - u_z*cos(theta2))/sin(theta6),
                  ((w_x*cos(theta1) + w_y*sin(theta1))*sin(theta2) - w_z*cos(theta2))/sin(theta6)) + pi/2

  X = d6 * ((w_z * sin(theta2) + cos(theta2) * (w_x * cos(theta1) + w_y * sin(theta1))) * cos(theta7) - (u_z * sin(theta2) + cos(theta2) * (u_x * cos(theta1) + u_y * sin(theta1))) * sin(theta7)) - d7 * (v_z * sin(theta2) + cos(theta2) * (v_x * cos(theta1) + v_y * sin(theta1))) + (-d1 + p_z) * sin(theta2) + cos(theta2) * (p_x * cos(theta1) + p_y * sin(theta1))
  Y = -d2 + d6 * ((w_x * sin(theta1) - w_y * cos(theta1)) * cos(theta7) - (u_x * sin(theta1) - u_y * cos(theta1)) * sin(theta7)) - d7 * (v_x * sin(theta1) - v_y * cos(theta1)) + p_x * sin(theta1) - p_y*cos(theta1)

  theta4 = ELBOW * acos((X**2 + Y**2 - a3**2 - a4**2)/(2*a3*a4))

  theta3 = atan2(Y*(a3 + a4*cos(theta4)) - X*a4*sin(theta4),
                 X*(a3 + a4*cos(theta4)) + Y*a4*sin(theta4))

  theta5 = atan2((v_x*sin(theta1)-v_y*cos(theta1))/(sin(theta6)),
                 (w_x*sin(theta1)-w_y*cos(theta1))*cos(theta7)-(u_x*sin(theta1)-u_y*cos(theta1))*sin(theta7)) - (theta3 + theta4)

  thetas = []
  for theta in [theta1, theta2, theta3, theta4, theta5, theta6, theta7]:
    thetas.append(deg(theta).evalf())

  thetas = Matrix(thetas)
  return thetas

def IK_sols():
  '''
  Enumerate through the eight possible solutions for a given pose.
  '''

  c = [[ 1,  1,  1],
       [ 1,  1, -1],
       [ 1, -1,  1],
       [-1,  1,  1],
       [ 1, -1, -1],
       [-1,  1, -1],
       [-1, -1,  1],
       [-1, -1, -1]]

  res = []
  for S, E, W in c:
```

```
        print(S, E, W)
        t = IK(SHOULDER=S, ELBOW=E, WRIST=W)
        res.append(t)

    stacked = np.zeros((7, 1))
    for row in res:
        stacked = np.hstack((stacked, np.array(row)))

    return stacked[:, 1:].T
```