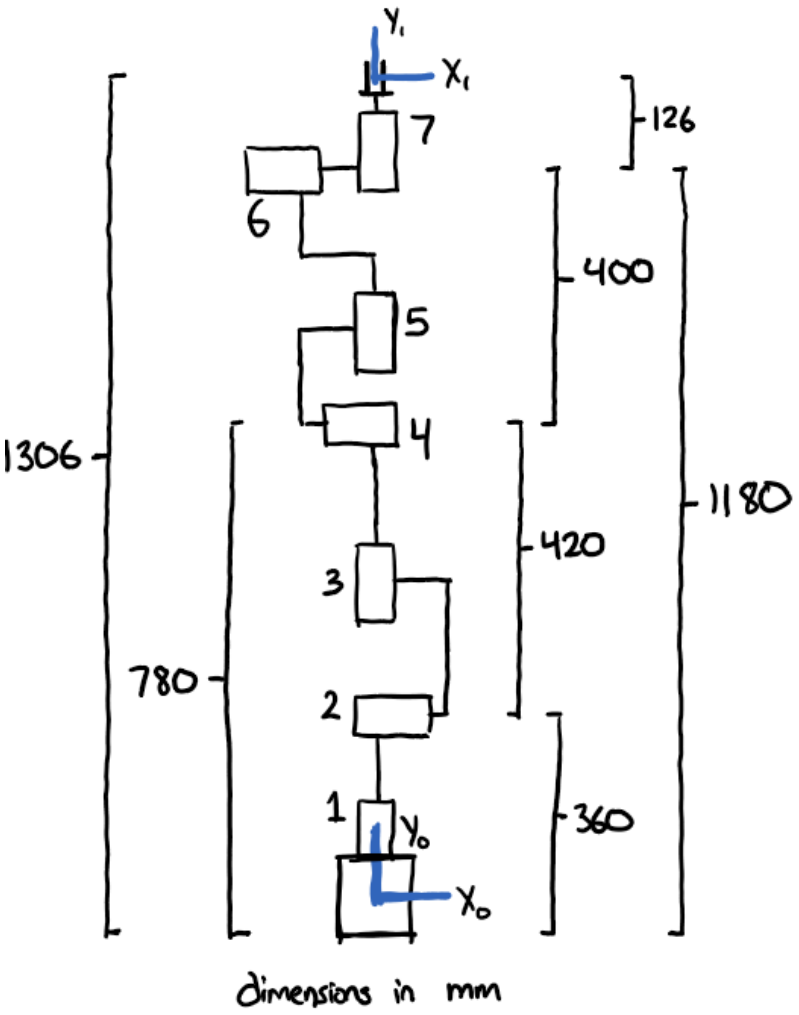
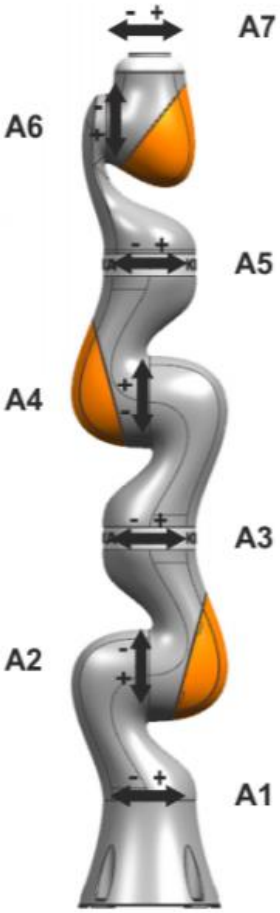
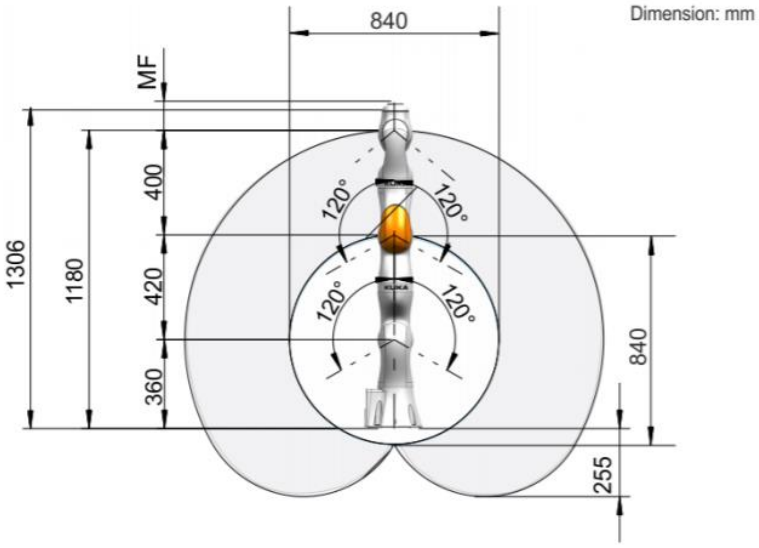
Schematic and Derivation of Forward Kinematics







Schematic and Derivation of Forward Kinematics

Derived spatial screw, end of tool relative to base

```
[[ 0.     1.     0.    -1.     0.     1.     0. ]
 [ 1.     0.     1.     0.     1.     0.     1. ]
 [ 0.     0.     0.     0.     0.     0.     0. ]
 [ 0.     0.     0.     0.     0.     0.     0. ]
 [ 0.     0.     0.     0.     0.     0.     0. ]
 [ 0.    -0.36   0.     0.78   0.    -1.18   0. ]]
```

Code used to derive forward kinematics:

```python
import numpy as np
import scipy.linalg as syl

def skew_sym(arr):
    return np.array([[0,-arr[2],arr[1]],[arr[2],0,-arr[0]],[-arr[1],arr[0],0]])

def matrix_rep(arr):
    matrix = np.zeros((4,4))
    matrix[0:3,0:3] = skew_sym(arr[0:3,0:1])
    matrix[0:3,3:4] = arr[3:6,0:1]
    return matrix

def prism_screw(a,q):
    S = np.zeros((6,1))
    S[3:6,0:1] = a
    return S

def rotat_screw(a,q):
    S = np.zeros((6,1))
    S[0:3,0:1] = a
    S[3:6,0:1] = -skew_sym(a) @ q
    return S
```

Schematic and Derivation of Forward Kinematics

```
# Kuka Forward Kinematics
# All measurements are in meters and radians

# Joint:          1  2  3  4  5  6  7
theta = np.array([0, 0, 0, 0, 0, 0, 0])


R = np.array([[1,0,0],[0,1,0],[0,0,1]])
p = np.array([[0],[1.306],[0]])
M = np.zeros((4,4))
M[0:3, 0:3] = R
M[0:3, 3:4] = p
M[3, 3] = 1


a_1 = np.array([[0],[1],[0]])
q_1 = np.array([[0],[0],[0]])

a_2 = np.array([[1],[0],[0]])
q_2 = np.array([[0],[0.360],[0]])

a_3 = np.array([[0],[1],[0]])
q_3 = np.array([[0],[0],[0]])

a_4 = np.array([[-1],[0],[0]])
q_4 = np.array([[0],[0.780],[0]])

a_5 = np.array([[0],[1],[0]])
q_5 = np.array([[0],[0],[0]])

a_6 = np.array([[1],[0],[0]])
q_6 = np.array([[0],[1.180],[0]])

a_7 = np.array([[0],[1],[0]])
q_7 = np.array([[0],[0],[0]])

S = np.zeros((6,7))
S[0:6,0:1] = rotat_screw(a_1,q_1)
S[0:6,1:2] = rotat_screw(a_2,q_2)
S[0:6,2:3] = rotat_screw(a_3,q_3)
S[0:6,3:4] = rotat_screw(a_4,q_4)
S[0:6,4:5] = rotat_screw(a_5,q_5)
S[0:6,5:6] = rotat_screw(a_6,q_6)
S[0:6,6:7] = rotat_screw(a_7,q_7)


T = (syl.expm(matrix_rep(S[0:6,0:1])*theta[0]) @
     syl.expm(matrix_rep(S[0:6,1:2])*theta[1]) @
     syl.expm(matrix_rep(S[0:6,2:3])*theta[2]) @
     syl.expm(matrix_rep(S[0:6,3:4])*theta[3]) @
     syl.expm(matrix_rep(S[0:6,4:5])*theta[4]) @
     syl.expm(matrix_rep(S[0:6,5:6])*theta[5]) @
     syl.expm(matrix_rep(S[0:6,6:7])*theta[6]) @ M)

print(S)
print(repr(T))
```

We predict robot's final frame position based on theta array of joint angles we pass it. In this case, theta is initialized to all 0.