Lab 3 Exercises

1 Derive the DH parameters <u>for each of the manipulators provided</u>. Use these to generate a model of the manipulator using the Robot Toolbox in MATLAB.

1.1 Work out the DH Parameters.

	θ_j	d_j	a_j	α_j
Link 1	θ_1	d_1	a_1	α_1
⋮	⋮	⋮	⋮	⋮
Link n	θ_n	d_n	a_n	α_n

1.2 In MATLAB, generate the robot model with the Robot Toolbox. Check that it matches the example provided.

```
L1 = Link('d',___,'a',___,'alpha',___,'offset',___,'qlim', [__,__]);
robot = SerialLink([L1 ... Ln],'name','myRobot');
q = zeros(1,n);   % This creates a vector of n joint angles at 0.
workspace = [-x +x -y +y -z +z];
scale = 1;
robot.plot(q,'workspace',workspace,'scale',scale);
```

1.3 You can manually play around with the robot.

```
robot.teach();
```

1.4 Get the current joint angles based on the position in the model.

```
q = robot.getpos()
```

1.5 The forward transformation matrix at the current joint configuration is given by:

```
T = robot.fkine(q)
```

1.6 We can also reverse this, given the current transformation T, and finding the joint angles q.

```
q = robot.ikine(T); % N.B. DOES NOT WORK FOR 3DOF MANIPULATORS
```

1.7 Get the Jacobian matrix. This maps joint velocities to end-effector velocities.

```
J = robot.jacob0(q);
J = J(1:3,1:3);   % For the 3-Link robots, we only need the first 3 rows. Ignore
                  this line for 6DOF robots.
```

1.8 Try and invert the Jacobian.

```
inv(J)
```

1.9 At certain joint configurations like this below the Jacobian can't be inverted. Are there other configurations where this happens?

```
q = zeros(1,n)
J = robot.jacob0(q)
inv(J)
```

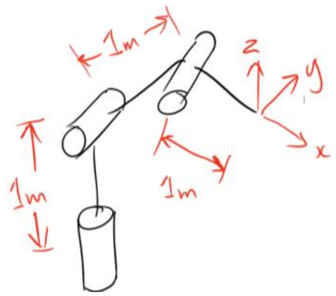
1.10 You can visualise how fast the end-effector can move in Cartesian space with the following command:

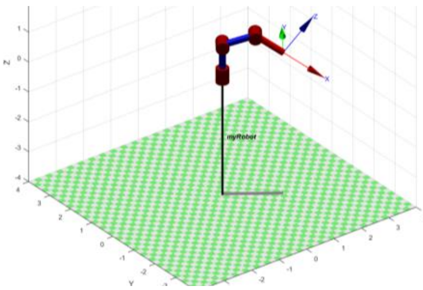
```
robot.vellipse(q);
```

1.11 Show what the velocity ellipse look like when the Jacobian "J" can't be inverted?

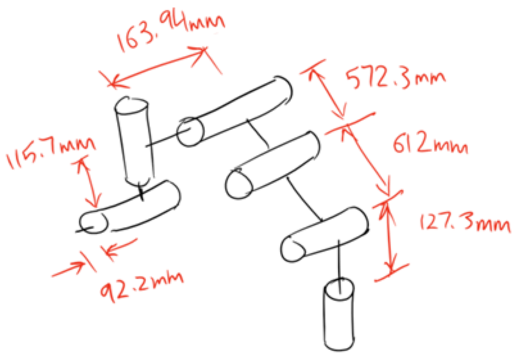
ROBOT 1: 3-Link 3D Robot

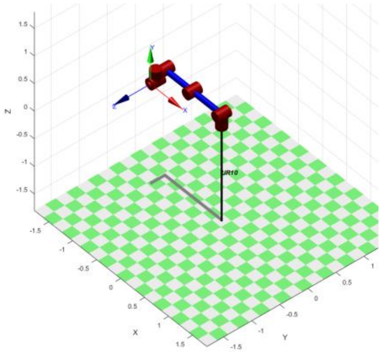
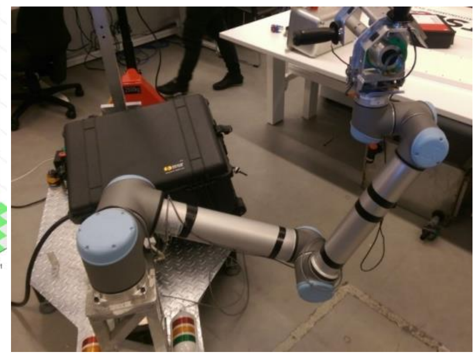
Note the orientation of the end-effector frame.





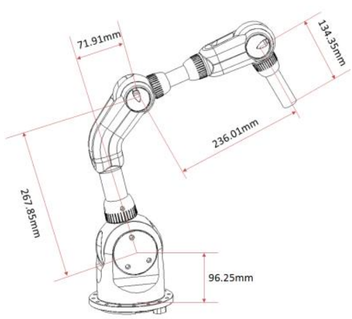
ROBOT 2: AAN-BOT (Assistance As Needed BOT) (6DOF)[1][2]

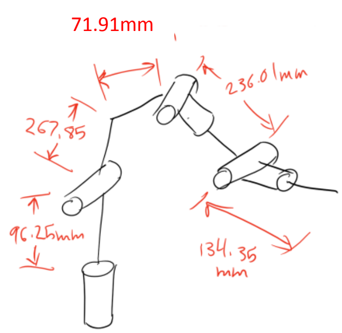


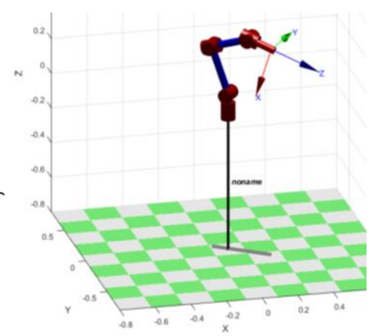



ROBOT 3: Submerged Pile Inspection Robot (SPIR)[3]

Note: this arm has 2DOF joints.







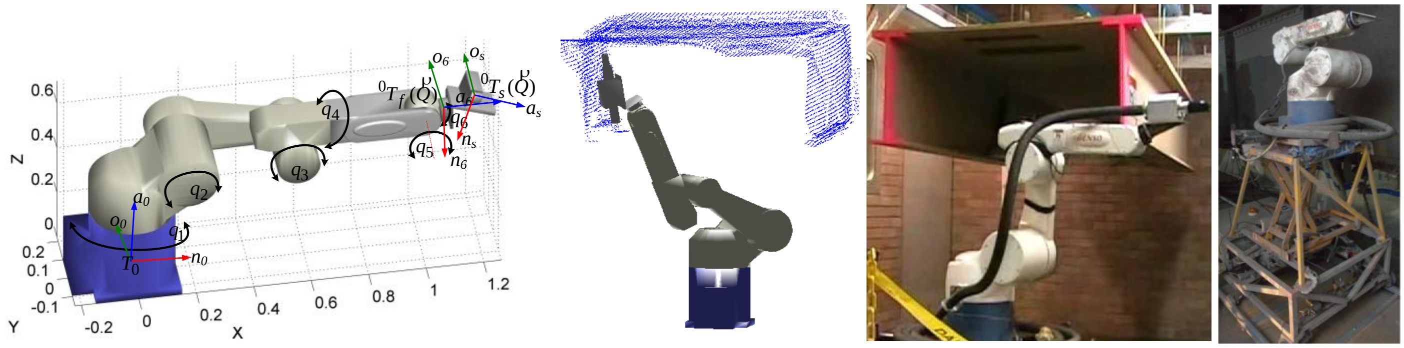

(Bonus) ROBOT 4: Sawyer Robot (7DOF)

These are brand new arms in CAS. We don't have any DH Parameters for them yet. Try and work them out for us!



[1] https://www.uts.edu.au/research-and-teaching/our-research/centre-autonomous-systems/cas-research/projects/assistive-robotic
[2] https://www.uts.edu.au/research-and-teaching/our-research/centre-autonomous-systems/cas-research/projects/assistive-robot-0
[3] https://www.uts.edu.au/research-and-teaching/our-research/centre-autonomous-systems/cas-research/projects/intelligent-0

2 Consider a Denso VM-6083D-W industrial robot shown in the PDF available on UTSOnline[4]



(a) (b) (c) (d)

Fig 1. (a)[5] Simulated robot model: joints and coordinate frames (i.e. robot base, end-effector and sensor), (b)[6] Blasting of a simulated environment constructed from exploration data, (c)[5] (d[7]) Grit-blasting system: a 6DOF manipulator on 2DOF base.

2.1 Download the PDF of the robot from UTSOnline

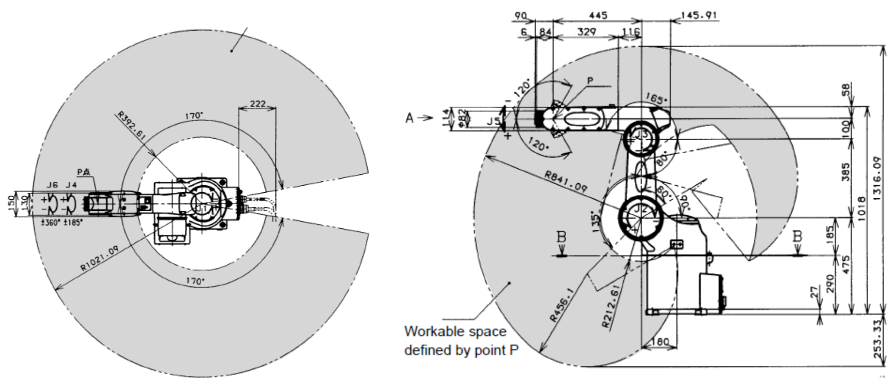
2.2 Determine the D&H parameters based upon the link measurements on the PDF

2.3 Determine and include the joint limits in your model

2.4 Sample the joint angles within the joint limits at 30 degree increments between each of the joint limits (note: from *jacob0* or *teach* you see that joint 6 is irrelevant to the position so don't iterate through it).

2.5 Use fkine to determine the point in space for each of these poses, so that you end up with a big list of points

2.6 Create a 3D model showing where the end effector can be over all these samples. (hint: it should look similar to the top and side workspace view on the PDF)



2.7 (Bonus) Get the start of the blast using fkine (to get end effector transform)

2.8 (Bonus) Get the end of the stream with TR * transl (blast stream length (i.e. 1m along z)

2.9 (Bonus) Project a line out of the end effector (i.e. a mock grit-blasting stream)

2.10 (Bonus) Create a surface plane that goes through [1.5,0,0] with a normal [-1,0,0]

```
[Y,Z] = meshgrid(-2:0.1:2,-2:0.1:2);
X = repmat(1.5,size(Y,1),size(Y,2));
surf(X,Y,Z);
```

2.11 (Bonus) Determine if and where the blast stream intersects with the surface plane

---

[4] https://online.uts.edu.au/bbcswebdav/pid-1370187-dt-content-rid-7998363_1/xid-7998363_1

[5] http://dx.doi.org/10.1016/j.autcon.2012.08.007

[6] http://hdl.handle.net/10453/9096

[7] http://hdl.handle.net/10453/16384