**7.1 · Describing a Robot Arm**

Within the Toolbox we represent a robot link with a Link object which is created by

>> L = Link([0, 0.1, 0.2, pi/2, 0])

L =

theta=q, d=0.1, a=0.2, alpha=1.571 (R,stdDH)

where the elements of the input vector are given in the order θ*k*, *dj*, *aj*, α*j*. The optional fifth element σ*j* indicates whether the joint is revolute (σ*i*=0) or prismatic (σ*i*= 0). If not specified a revolute joint is assumed.

The displayed values of the Link object show its kinematic parameters as well as other status such the fact that it is a revolute joint (the tag R) and that the standard Denavit-Hartenberg parameter convention is used (the tag stdDH).



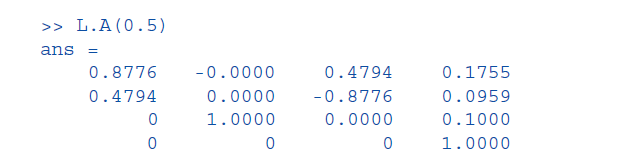
Although a value was given for θ it is not displayed – that value simply served as a placeholder in the list of kinematic parameters. θ is substituted by the joint variable *q* and its value in the Link object is ignored. The value will be managed by the Link object.

A Link object has many parameters and methods which are described in the online documentation, but the most common ones are illustrated by the following examples.

A slightly different notation, *modified Denavit-Hartenberg* notation is discussed in Sect. 7.5.3.

The link transform Eq. 7.2 for *q* =0.5 rad is





is a 4 ×4 homogeneous transformation. Various link parameters can be read or altered, for example

>> L.RP

ans = R

indicates that the link is prismatic and

>> L.a

ans =

0.2000

returns the kinematic parameter *a*. Finally a link can contain an offset

>> L.offset = 0.5;

>> L.A(0)

ans =

0.8776 -0.0000 0.4794 0.1755

0.4794 0.0000 -0.8776 0.0959

0 1.0000 0.0000 0.1000

0 0 0 1.0000

which is added to the joint variable before computing the link transform and will be discussed in more detail in Sect. 7.5.1.

**7.2 lForward Kinematics**

**7.2.1 lA 2-Link Robot**

The first robot that we will discuss is the two-link planar manipulator shown in Fig. 7.3.

It has the following Denavit-Hartenberg parameters which we use to create a vector of Link objects



>> L(1) = Link([0 0 1 0]);

>> L(2) = Link([0 0 1 0]);

>> L

L =

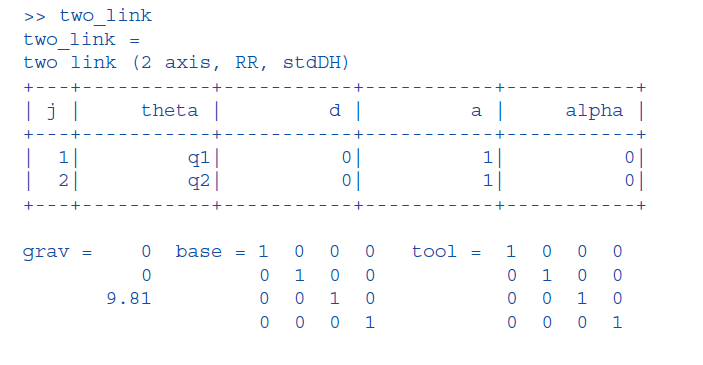
theta=q1, d=0, a=1, alpha=0 (R,stdDH)

theta=q2, d=0, a=1, alpha=0 (R,stdDH)

which are passed to the constructor SerialLink

>> two\_link = SerialLink(L, 'name', 'two link');

which returns a SerialLink object that we can display



This provides a concise description of the robot. We see that it has 2 revolute joints as indicated by the structure string 'RR', it is defined in terms of standard Denavit-Hartenberg parameters, that gravity is acting in the default *z*-direction.\_

The kinematic parameters of the link objects are also listed and the joint variables are shown as variables q1 and q2. We have also assigned a name to the robot which will be shown whenever the robot is displayed graphically. The script

>> mdl\_twolink

performs the above steps and defines this robot in the MATLAB® workspace, creating a SerialLink object named twolink.

The SerialLink object is key to the operation of the Robotics Toolbox. It has a great many methods and properties which are illustrated in the rest of this part, and described in detail in the online documentation.

Some simple examples are

>> twolink.n

ans =

2

which returns the number of joints, and

>> links = twolink.links

L =

theta=q1, d=0, a=1, alpha=0 (R,stdDH)

theta=q2, d=0, a=1, alpha=0 (R,stdDH)

which returns a vector of Link objects comprising the robot.\_ We can also make a copy of a SerialLink object

>> clone = SerialLink(twolink, 'name', 'bob')

clone =

bob (2 axis, RR, stdDH)

+---+-----------+-----------+-----------+-----------+

| j | theta | d | a | alpha |

+---+-----------+-----------+-----------+-----------+

| 1| q1| 0| 1| 0|

| 2| q2| 0| 1| 0|

+---+-----------+-----------+-----------+-----------+

grav = 0 base = 1 0 0 0 tool = 1 0 0 0

0 0 1 0 0 0 1 0 0

9.81 0 0 1 0 0 0 1 0

0 0 0 1 0 0 0 1

which has the name 'bob'.\_

Now we can put the robot arm to work. The forward kinematics are computed using the fkine method. For the case where *q*1=*q*2= 0

>> twolink.fkine([0 0])

ans =

1 0 0 2

0 1 0 0

0 0 1 0

0 0 0 1

the method returns the homogenous transform that represents the pose of the second link coordinate frame of the robot, ***T***2. For a different configuration the tool pose is

>> twolink.fkine([pi/4 -pi/4])

ans =

1.0000 0 0 1.7071

0 1.0000 0 0.7071

0 0 1.0000 0

0 0 0 1.0000

By convention, joint coordinates with the Toolbox are row vectors.

The robot can be visualized graphically

>> twolink.plot([0 0])

>> twolink.plot([pi/4 -pi/4])

as stick figures shown in Fig. 7.4. The graphical representation includes the robot’s name, the final-link coordinate frame, ***T***2 in this case, the joints and their axes, and a shadow on the ground plane. Additional features of the plot method such as multiple views and multiple robots are described in Sect. 7.8 with additional details in the

online documentation.

The simple two-link robot introduced above is limited in the poses that it can achieve since it operates entirely within the *xy*-plane The robot can be visualized graphically

>> twolink.plot([0 0])

>> twolink.plot([pi/4 -pi/4])

as stick figures shown in Fig. 7.4. The graphical representation includes the robot’s name, the final-link coordinate frame, ***T***2 in this case, the joints and their axes, and a shadow on the ground plane. Additional features of the plot method such as multiple views and multiple robots are described in Sect. 7.8 with additional details in the

online documentation.

The simple two-link robot introduced above is limited in the poses that it can achieve since it operates entirely within the *xy*-plane.