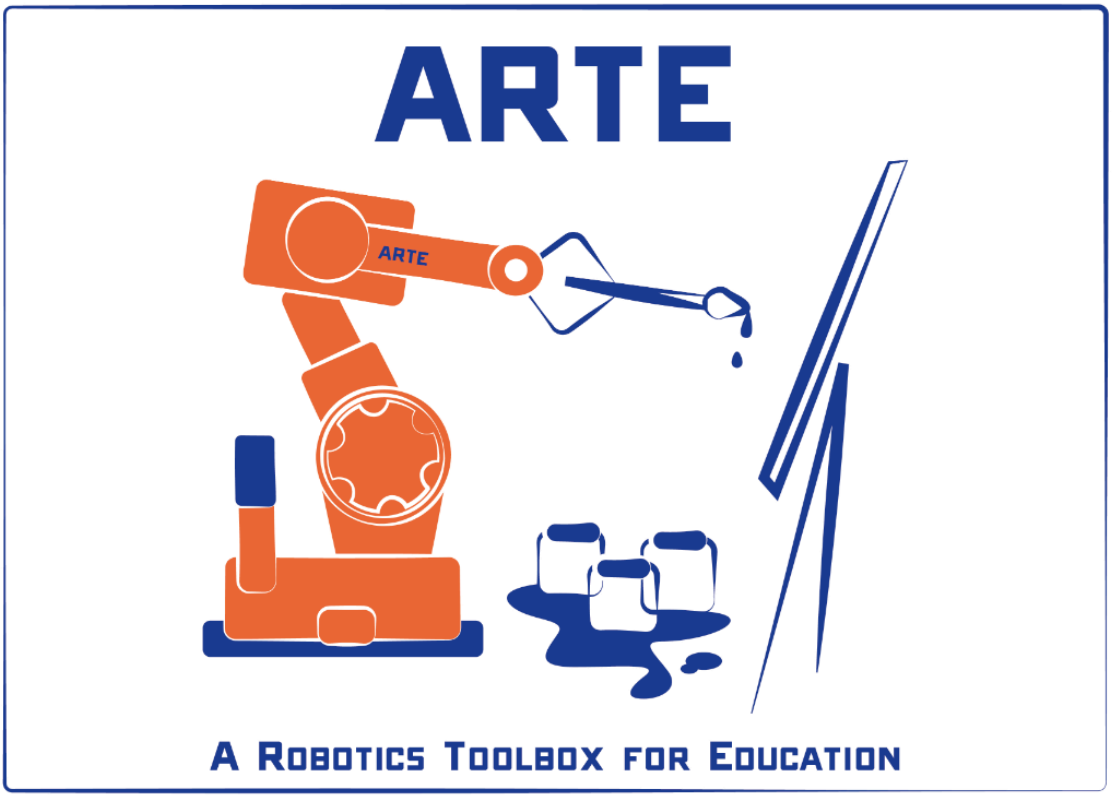
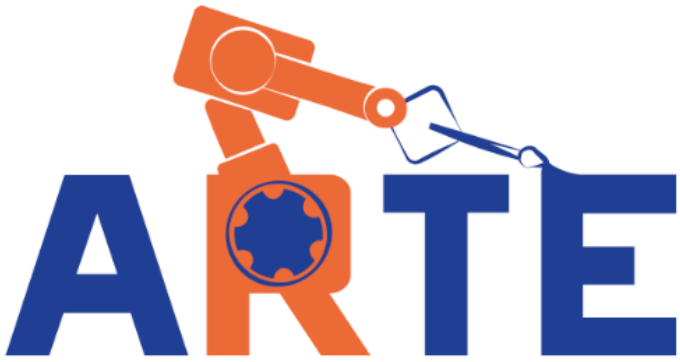


PRACTICAL SESSION 0: INTRODUCTION TO ARTE

Arturo Gil Aparicio

arturo.gil@umh.es



Index

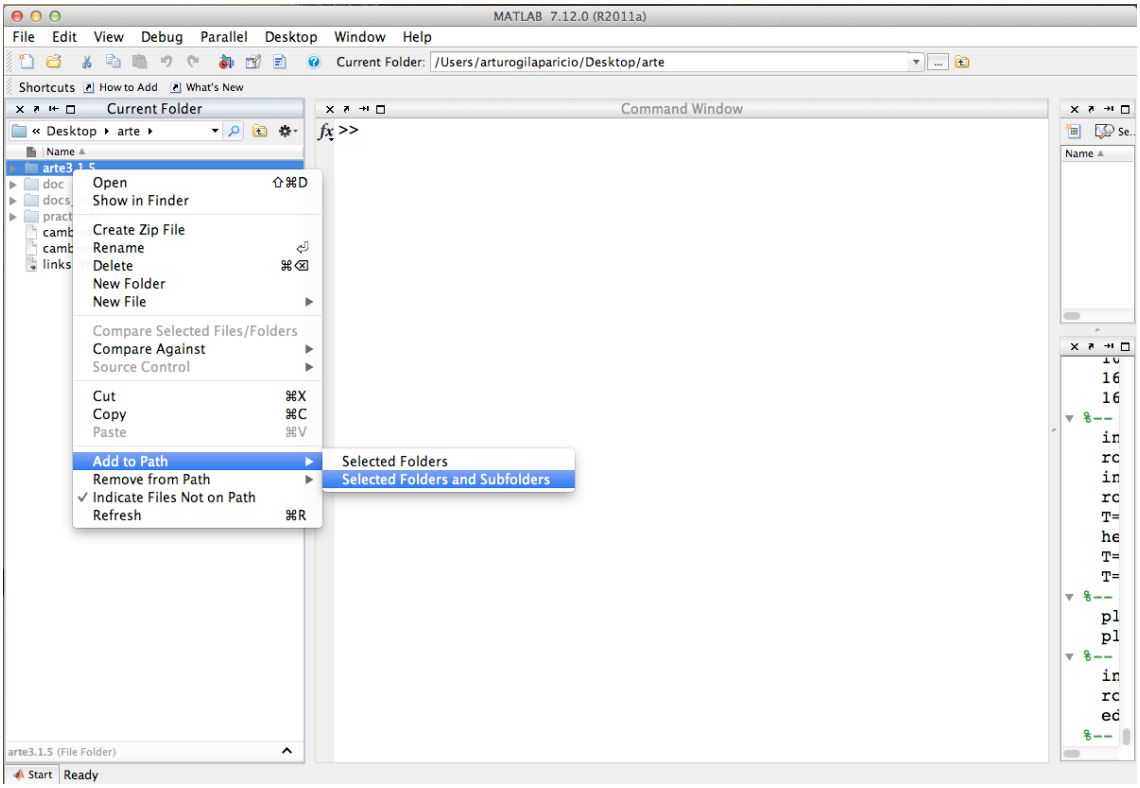
1 Install the library

Installing the library under Matlab is easy, just follow the steps:

a) Download and unzip the library from http://arvc.umh.es/arte

b) Copy the directory containing all the .m files in a directory named, for example, arte3.1.5. For example, copy it to the Desktop.

c) Add the whole directory and sub-directories to the Matlab PATH. To do this, click on File→Set Path→ Include the path where you placed the library directory. Alternatively, you can also use the file explorer window in Matlab. Right click on the library's directory and select *Add to Path→ Selected Folders and Subfolders*.

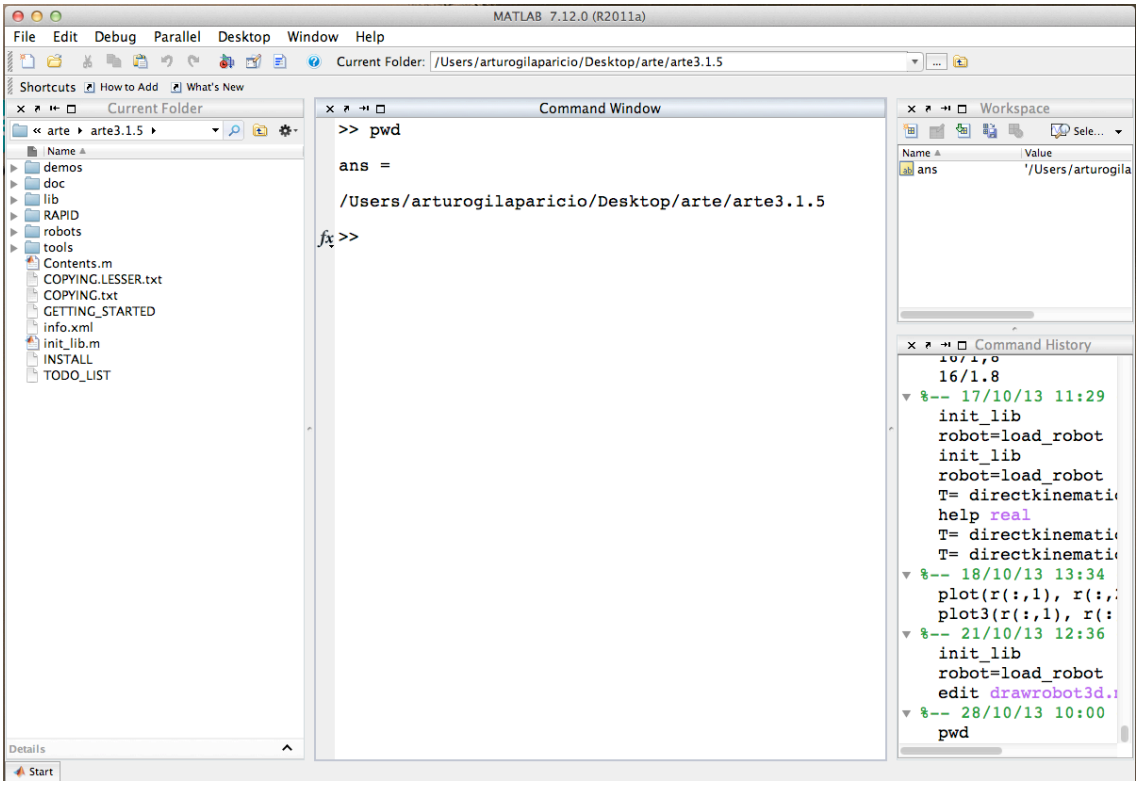


d) To initialize the library inside Matlab, you should be placed in the directory where the library is installed, for example, if you copied the library to the Desktop, you should check the following (in a Linux-based system):

```
>> pwd

ans =

/Users/arturo/Desktop/arte3.1.4
```

Windows users should have:

```
>> pwd

ans =

C:\Documents and Settings\arturo\Desktop\arte3.1.5
```



e) Next, in the Matlab command prompt type:

```
>> init_lib

%     ARTE (A Robotics Toolbox for Education)
%     Copyright (C) 2012  Arturo Gil Aparicio, arturo.gil@umh.es
%     http://arvc.umh.es/arte
%
%     This program is free software: you can redistribute it and/or modify
%     it under the terms of the GNU Lesser General Public License as published by
%     the Free Software Foundation, either version 3 of the License, or
%     any later version.
%
%     This program is distributed in the hope that it will be useful,
%     but WITHOUT ANY WARRANTY; without even the implied warranty of
%     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%     GNU  Lesser General Public License for more details.
%
%     You should have received a copy of the GNU Lesser General Public License
%     along with this program.  If not, see  <http://www.gnu.org/licenses/>.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%     To begin with, try loading a robot:
%     >> robot = load_robot('ABB','IRB140');
%
%     Next, draw it on its zero position:
%     >> drawrobot3d(robot,[0 0 0 0 0 0])
%
%     Next, try a different pose:
%     >> drawrobot3d(robot,[0 pi/2 -pi/2 0 0 0])
%
%     Finally, use the teach pendant to move the robot:
%     >> teach
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

To view all functions and demos type:

```
>>help Contents
```

If you want to have a fast view of the main capabilities of the library, type:

```
>> demos
```

An introduction to the library can be viewed here:

http://www.youtube.com/watch?v=rpHAOjYYJRU&list=PLClKgnzRFYe72qDYmj5CRpR9ICNnQehup

In spanish:

http://www.youtube.com/watch?v=s8QQydJ9PwI&list=PLClKgnzRFYe72qDYmj5CRpR9ICNnQehup&index=18

2 Load a robot

The library is based on the robot structure. This is a special data type defined in Matlab to store all the necessary parameters that define a robot. Several robot parameters are already included in the library. In addition, it is easy to add your own robots to the library. In order to load a robot, use the load_robot function.

You can type the following commands at the Matlab prompt:

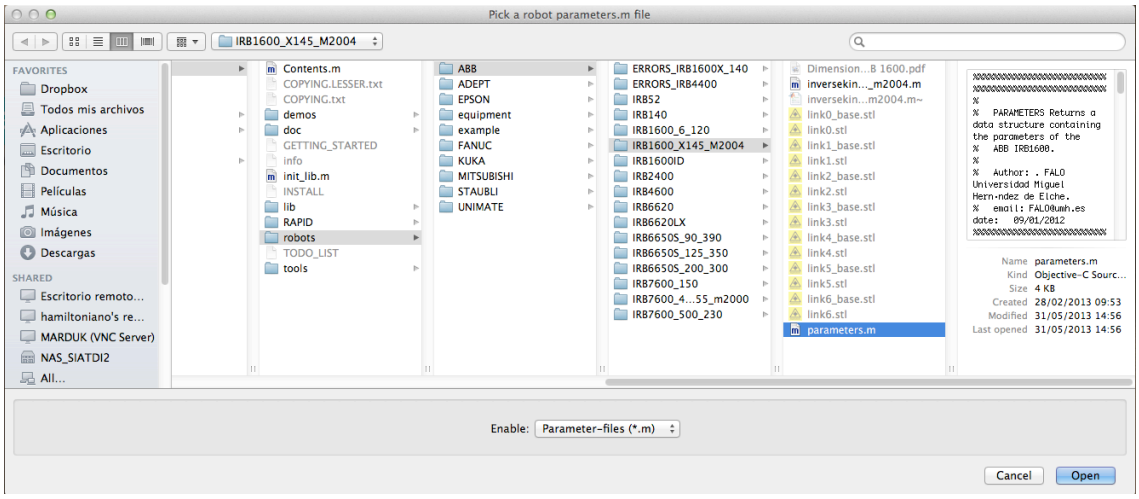
```
>> robot=load_robot('abb','irb140')

ans =

/Users/arturogilaparicio/Desktop/arte/arte3.1.5/robots/ABB/IRB140


Reading link 0
 /Users/arturogilaparicio/Desktop/arte/arte3.1.5/robots/ABB/IRB140/link0.stl
EndOfFile found...
Reading link 1
 /Users/arturogilaparicio/Desktop/arte/arte3.1.5/robots/ABB/IRB140/link1.stl
EndOfFile found...
Reading link 2
 /Users/arturogilaparicio/Desktop/arte/arte3.1.5/robots/ABB/IRB140/link2.stl
EndOfFile found...
Reading link 3
 /Users/arturogilaparicio/Desktop/arte/arte3.1.5/robots/ABB/IRB140/link3.stl
EndOfFile found...
Reading link 4
 /Users/arturogilaparicio/Desktop/arte/arte3.1.5/robots/ABB/IRB140/link4.stl
EndOfFile found...
Reading link 5
 /Users/arturogilaparicio/Desktop/arte/arte3.1.5/robots/ABB/IRB140/link5.stl
EndOfFile found...
Reading link 6
 /Users/arturogilaparicio/Desktop/arte/arte3.1.5/robots/ABB/IRB140/link6.stl
EndOfFile found...
robot =

                    name: 'ABB_IRB140_M2000'
                      DH: [1x1 struct]
                       J: []
    inversekinematic_fn: [1x33 char]
     directkinematic_fn: [1x25 char]
                     DOF: 6
```

```
                  kind: 'RRRRRR'
              maxangle: [6x2 double]
                velmax: [6x1 double]
              accelmax: [6x1 double]
         linear_velmax: 2.5000
                    T0: [4x4 double]
                 debug: 0
                     q: [6x1 double]
                    qd: [6x1 double]
                   qdd: [6x1 double]
                  time: []
              q_vector: []
             qd_vector: []
            qdd_vector: []
           last_target: [4x4 double]
        last_zone_data: 'fine'
                 tool0: [1x19 double]
                 wobj0: []
        tool_activated: 0
                  path: [1x65 char]
            graphical: [1x1 struct]
                  axis: [1x6 double]
          has_dynamics: 1
              dynamics: [1x1 struct]
                motors: [1x1 struct]
```

The load_robot function can also be called without parameters:

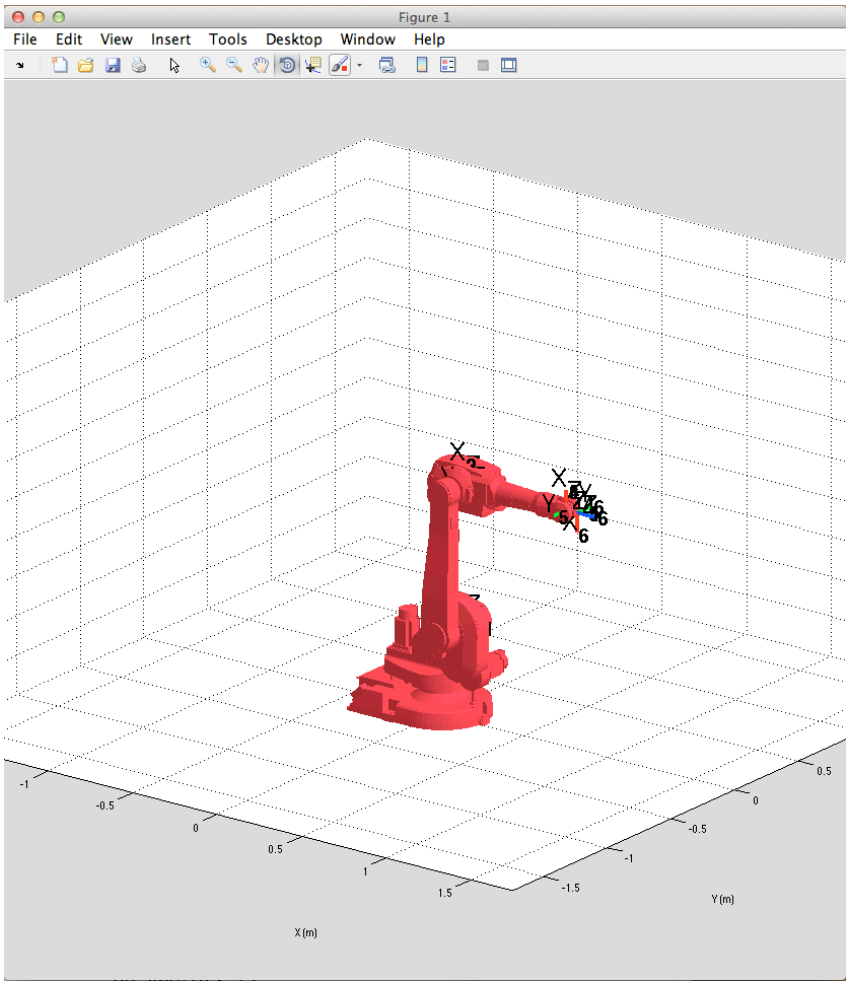
```
>> robot=load_robot
```

In this case, you should navigate inside the directory of any robot that you want to load in the library and click on the **parameters.m** file. If everything is correct, a plot of the current robot should appear in a Matlab Figure.




```
>> robot=load_robot

ans =

/Users/arturogilaparicio/Desktop/arte/arte3.1.5/robots/ABB/IRB1600_X145_M2004
```

```
Reading link 0

/Users/arturogilaparicio/Desktop/arte/arte3.1.5/robots/ABB/IRB1600_X145_M2004/
link0.stl
EndOfFile found...
Reading link 1

/Users/arturogilaparicio/Desktop/arte/arte3.1.5/robots/ABB/IRB1600_X145_M2004/
link1.stl
EndOfFile found...
Reading link 2

/Users/arturogilaparicio/Desktop/arte/arte3.1.5/robots/ABB/IRB1600_X145_M2004/
link2.stl
EndOfFile found...
Reading link 3

/Users/arturogilaparicio/Desktop/arte/arte3.1.5/robots/ABB/IRB1600_X145_M2004/
link3.stl
EndOfFile found...
Reading link 4

/Users/arturogilaparicio/Desktop/arte/arte3.1.5/robots/ABB/IRB1600_X145_M2004/
link4.stl
EndOfFile found...
Reading link 5

/Users/arturogilaparicio/Desktop/arte/arte3.1.5/robots/ABB/IRB1600_X145_M2004/
link5.stl
EndOfFile found...
Reading link 6

/Users/arturogilaparicio/Desktop/arte/arte3.1.5/robots/ABB/IRB1600_X145_M2004/
link6.stl
EndOfFile found...
robot =

                  name: 'abb_irb1600_X145_m2004'
                    DH: [1x1 struct]
                     J: []
    inversekinematic_fn: [1x49 char]
                   DOF: 6
                  kind: 'RRRRRR'
              maxangle: [6x2 double]
                velmax: [6x1 double]
              accelmax: [6x1 double]
         linear_velmax: 2.5000
                    T0: [4x4 double]
                 debug: 0
                     q: [6x1 double]
                    qd: [6x1 double]
                   qdd: [6x1 double]
                  time: []
              q_vector: []
             qd_vector: []
            qdd_vector: []
           last_target: [4x4 double]
         last_zone_data: 'fine'
                 tool0: [1x19 double]
                 wobj0: []
         tool_activated: 0
                   path: [1x77 char]
             graphical: [1x1 struct]
                  axis: [1x6 double]
          has_dynamics: 1
              dynamics: [1x1 struct]
                motors: [1x1 struct]
```



After typing:

```
>> robot=load_robot
```

And selecting a robot in the library, the robot parameters are stored in a Matlab variable named `robot`. This variable may have any other different names, for example:

```
>> abb=load_robot('abb','irb140')
```

After that, you can use the abb variable for any subsequent call to any function in the library:

```
>> directkinematic(abb, [0 0 0 0 0 0])
>> drawrobot3d(abb)
```

However, the robot variable name is reserved and very useful in the library. It is actually declared as a global variable, meaning that any function in the library has

free access to the variable contents. This is useful, for example, because the teach function uses this variable to simulate the robot easily:

```
>> robot=load_robot
>> teach
```

3 Use the GUI

At the Matlab's prompt, type:

```
>> robot=load_robot
>> teach
```

You may load any parameters.m file existing in the library. You can use the `teach` function to move a robot interactively. Type:

```
>> init_lib
ARTE (A Robotics Toolbox for Education) (c) Arturo Gil 2012
http://www.arvc.umh.es/arte
>> robot=load_robot('abb', 'IRB140');

ans =
/Users/arturogilaparicio/Desktop/arte_lib2.7/robots/abb/IRB140

Reading link 0
EndOfFile found...
Reading link 1
EndOfFile found...
Reading link 2
EndOfFile found...
Reading link 3
EndOfFile found...
Reading link 4
EndOfFile found...
Reading link 5
EndOfFile found...
Reading link 6
EndOfFile found...

>> teach
Select the desired view for your robot
>>
```

The graphical application appears (Figure 10):

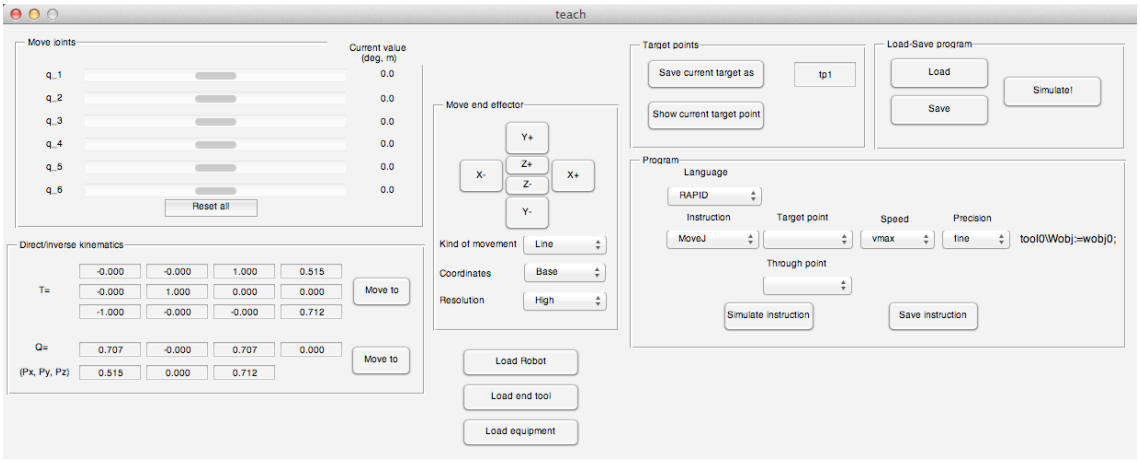


Figure 1

This graphical application covers different topics, such as direct kinematics, inverse kinematics, quaternions and RAPID programming. For the moment we will concentrate on the direct kinematic problem. Use the slider controls to modify the joint coordinates of the robot. You can observe the position and orientation of the end effector by looking at the homgeneous matrix `T` and also with the orientation quaternion Q and (Px, Py, Pz).