Rigid motion and homogenous transformation

Exercise 1 :

Create a Matlab function that calculates the forward kinematics of the 2-DOF planar robot arm with equations (1.1) and (1.2) in SPONG (p.20-24) – in other words, given inputs q1 and q2 for the two angles as well as (constant) lengths a1 and a2, calculate the cartesian coordinates x and y of the end effector. You may use my template function "PlanarRobotFK.m".

Also try to draw the robot arm. You may use the "PlanarRobotPlot.m" function to draw the robot.
Test the function with different values for a1, a2, q1 and q2 - use e.g. "PlanarRobot_test.m" for your tests

Exercise 2:

Create a Matlab function that calculates the inverse kinematics with equations (1.7) and (1.8) for the similar robot to exercise 1. This function should output the angles q1 and q2 when given the inputs x and y for the end effector position (as well as constants a1 and a2). You may use my template function "PlanarRobotIK.m".
NOTE: I suggest to use the "atan2"-function in MATLAB as it handles positive/negative x- and y-values. Write "help atan2" in MATLAB for more info.

Test the function similarly to exercise 1.

Exercise 3:

You now have to program the robot arm simulator to follow a specific path (in Matlab). Try with different paths – use "PlanarRobot_test.m" as starting point.
What is the "reach" of the robot ? (or, how far can it reach..)
Will it be able to grasp any (2D) object within this reach ?

Exercise 4 :

Create a new function to calculate the forward kinematics similar to exercise 1, but that use instead a homogeneous representation and homogeneous transformation matrices to calculate the joint positions of the 2D planar robot arm.
Test the function similarly to exercise 1.
(OBS: notice how the homogeneous transformation matrix contains more information than just the (x,y)-coordinates of the end effector. It also contains the end effector angle, ie. a complete specification of a 2D rigid motion (3 parameters)).

Exercise 5 :

Try to extend your function from exercise 4 to make a robot with, say, 4 joints – i.e. a 2D planar robot with 4 revolute joints. This is simply using two extra similar transformation matrices with two new extra parameters q3 and q4 (angles).
You may use the “PlanarRobotPlot.m “ function to verify your own function as it (implicitly) calculates the transformations when plotting.