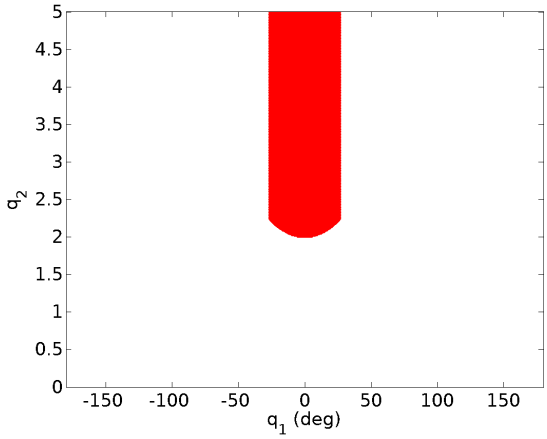
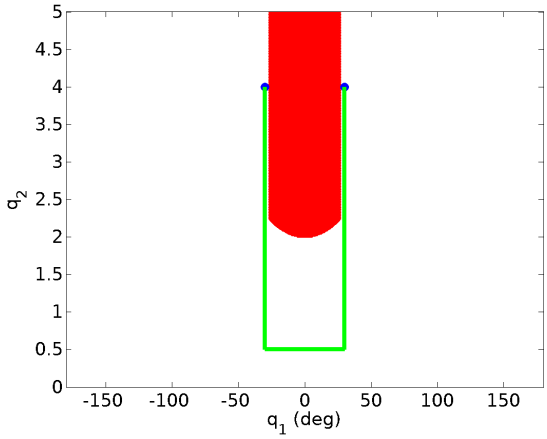
Homework 10 Solutions

1. The set of points occupied by the robot (considering the links to be very thin straight lines) is given by $(\lambda\cos(q_1), \lambda\sin(q_1))$ for all $\lambda \in [0, q_2]$. Hence, from the provided geometry of the obstacle set, we see that there is a collision between some part of the robot and the obstacle if $|q_1| < \alpha_{min}$ and $q_2 > \frac{r}{\cos(q_1)}$ where $r = 2$ and $\alpha_{min} = \tan^{-1}\left(\frac{1}{2}\right)$. Hence, we see that the set of all values of (q_1, q_2) which result in a collision are as shown in the figure below. The region shown in red is the obstacle set represented in configuration space.



We are given the initial configuration $(q_1, q_2) = (30, 4)$ with q_1 specified in degrees and the desired final configuration $(q_1, q_2) = (-30, 4)$. It is seen that just moving from the initial to final configuration along a straight line in configuration space will cause a collision with the obstacle. However, we can move around the obstacle as, for example, shown in the figure below. Here, we pick the intermediate configurations $(q_1, q_2) = (30, 0.5)$ and $(q_1, q_2) = (-30, 0.5)$. This corresponds to actuating the prismatic joint to make the overall robot length small enough and then rotating to the desired final value of q_1 and then extending the arm again by actuating the prismatic joint. Note that if the prismatic joint can actuate in the negative direction as well, then one intermediate configuration would be sufficient of the form $(q_1, q_2) = (0, -L)$ with L being sufficiently large so as to avoid collisions. Another possibility would be to just increase q_1 while keeping q_2 fixed so as to move counter-clockwise to avoid the obstacle while moving from the initial to the final desired configuration. Then, the intermediate configuration could be, for example, $(q_1, q_2) = (180, 4)$.



2. With the locations of the provided set of points denoted as $(\overline{x}_i, \overline{y}_i)$, the measurements d_i of the distances from the robot position to each of these points provide equations of the form: $\sqrt{(x_r - x_i)^2 + (y_r - y_i)^2} = d_i$. Denoting the left hand sides of these equations by $f_i(x_r, y_r)$, the equations above are of the form $f_i(x_r, y_r) = d_i$. Now, defining $e = \sqrt{\sum_{i=1}^{n}(f_i(x_r, y_r) - d_i)^2}$, we want to find (x_r, y_r) to minimize the error e, i.e., to find the best fit for the robot position given the distance measurements. Matlab code to find (x_r, y_r) to minimize the error e is given in `http://crrl.poly.edu/EL5223/triangulation_example.m`. Running this Matlab code, we get $(x_r, y_r) = (100, 100)$, i.e., we find the robot position perfectly (to within numerical noise). With real sensors, there will always be some amount of noise. For example, considering distance measurement noise of 0.1 units, we get position estimation errors of around 0.1 to 0.2 units (e.g., $(x_r, y_r) = (100.09, 99.92)$). Code to add this random noise is also included in the Matlab code linked above (you will need to uncomment the line as indicated in the comments in the code). Note that since the added noise is randomly generated, the numerical output will be different each time you run it.