KFUPM
Term 081
EE 656 – Robotics & Control

Homework 1

Solution

Submitted to
Dr. Ahmed Masoud

By
Mohammad Shahab
ID. 227598

25 October 2008

KFUPM

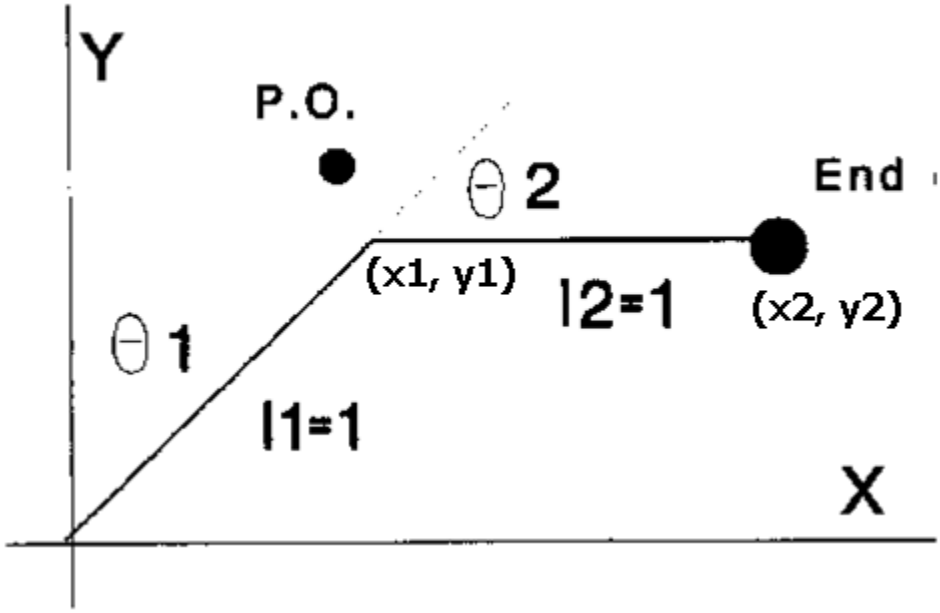
Term 081

EE 656: Robotics & Control

Homework 1

Question 1

The robotic arm is shown in below figure.



As you can see, the arm has the specifications of:

- Link 1. L1 = 1
- Link 2, L2 = 1
- $0 \leq \theta_1 \leq 2\pi$
- $0 \leq \theta_2 \leq 2\pi$
- Fixed frame with Link 1's bottom attached to (0, 0)

- Requirements:

  Is to compute the Work Space (W-Space) and the Configuration Space (C-Space) of this robotic arm with obstacles of:

  1. No obstacles
  2. Point obstacle @ (0.8, 0.8)
  3. Point obstacle @ (1.6, 1.6)
  4. Point obstacle @ (3, 3)

Computer Program

For constructing the computer program that will compute W-Space & C-Space, **MATLAB2007b** is used. Here in this section will be the description of the program. You can also look at the MATLAB full code in the appendix.

Note: in the program, a **resolution** (Δ) is used for the magnitude of iterations increment in the variables throughout the program. The default resolution used is Δ= 0.05 for increments on θ_1, θ_2.

1) With the specifications above, we can compute the locations of (x_1, y_1) & (x_2, y_2), as shown in the above figure, at any θ_1, θ_2values with the equations:

$$x_1 = L1 \sin\theta_1$$
$$y_1 = L1 \cos\theta_1$$
$$x_2 = x_1 + L2 \sin(\theta_1 + \theta_2)$$
$$y_2 = y_1 + L2 \cos(\theta_1 + \theta_2)$$

2) However, we are not only interested in the 'end' points of each link only. We also need to compute the lines connecting these end-points together. By that, we will have the points that form the whole robot arm in space. To compute the linking straight lines between $(0, 0)$ & (x_1, y_1) and (x_1, y_1) & (x_2, y_2) for any time:

   - for (x_p, y_p) which describe any point at link 1, we construct line equation
     - $0 \le x_p \le x_1$ With increments by the designated resolution, or $x_1 \le x_p \le 0$ depending on the sign of x_1
     - $y_p = \frac{y_1}{x_1} x_p$

   - for (x_{pp}, y_{pp}) which describe any point at link 2, we construct line equation
     - $x_1 \le x_{pp} \le x_2$ With increments by the designated resolution, or $x_2 \le x_{pp} \le x_1$ depending on the sign of $(x_2 - x_1)$
     - $y_{pp} = \frac{y_2 - y_1}{x_2 - x_1}(x_{pp} - x_1) + y_1$

3) So having (x_p, y_p) & (x_{pp}, y_{pp}) for any values of (x_1, y_1) & (x_2, y_2) and therefore for any value of θ_1, θ_2, we check for obstacles, say (x_o, y_o) by:

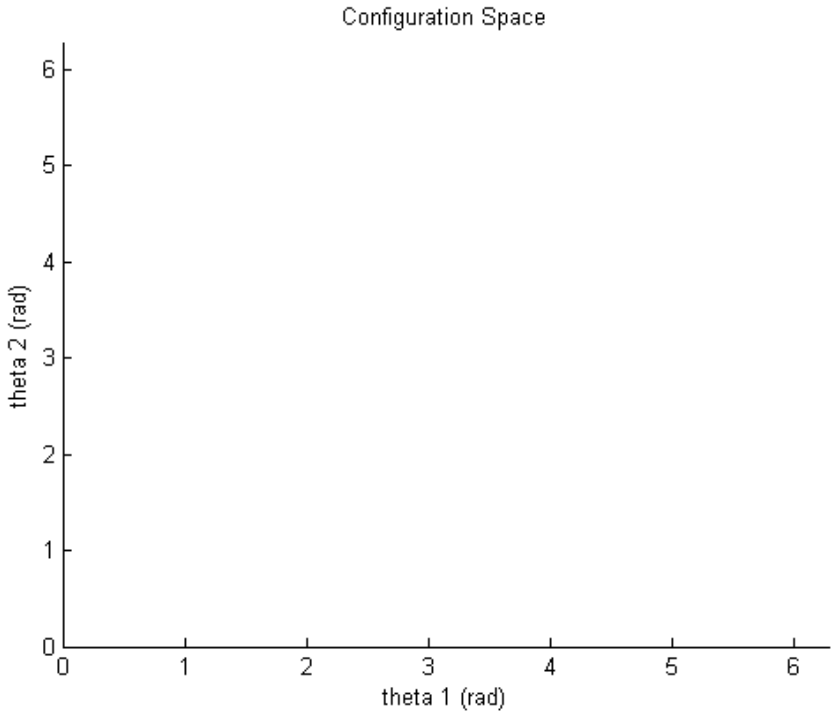
   - Checking $(x_o - \Delta, y_o - \Delta) \le (x_p, y_p) \le (x_o + \Delta, y_o + \Delta)$ or
   - Checking $(x_o - \Delta, y_o - \Delta) \le (x_{pp}, y_{pp}) \le (x_o + \Delta, y_o + \Delta)$
   - If any condition above satisfied, θ_1, θ_2 of that iteration is marked as a **C-Space Obstacle**.

4) Plots are shown to display:
   1. C-Space with obstacles
   2. W-Space with obstacles
   3. Start point & end point at both C-Space & W-Space

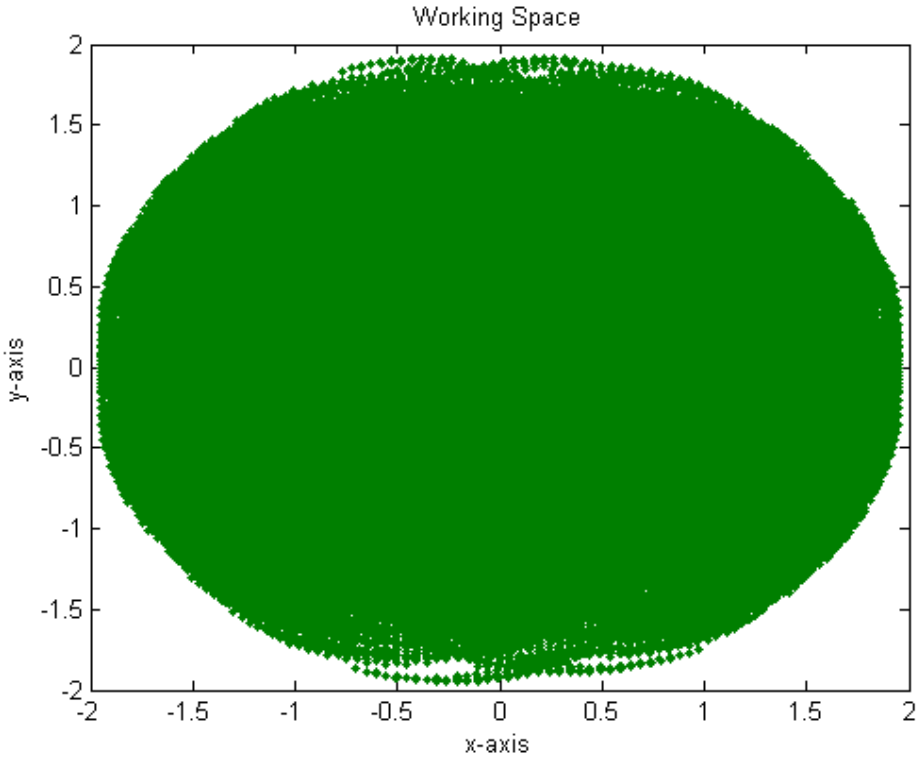
Results

As mentioned above, we will investigate the Spaces for four different situations.

1. If environment with *no obstacles*, the C-Space and W-Space is shown below

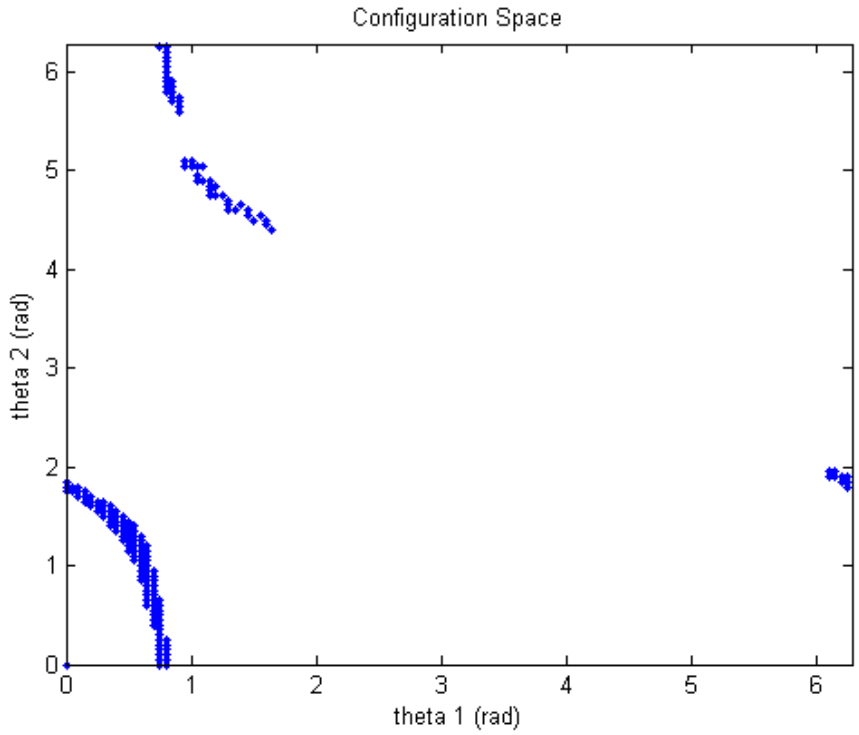


With the light area as the C-Space

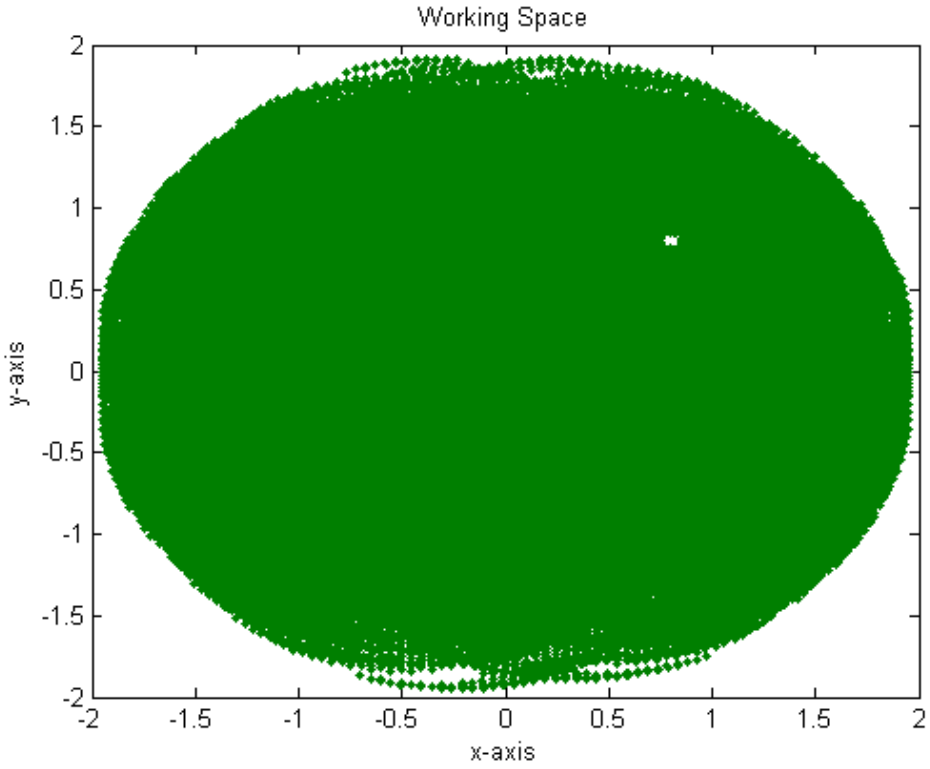


With the dark area as the W-Space

2. With obstacle @ (0.8, 0.8), the spaces are shown below



With light area as the C-Space & dark area as the '**C-Space Obstacle**'



With dark area as the W-Space & light area as the W-Space Obstacle

3. For both situations of obstacles @ **(1.6, 1.6) & (3, 3),** we can see that they are outside the reachable of the arm. We can run the computer program to see this result. The plots of the C-Space & W-Space are the same as the result of the first situation of no obstacles.

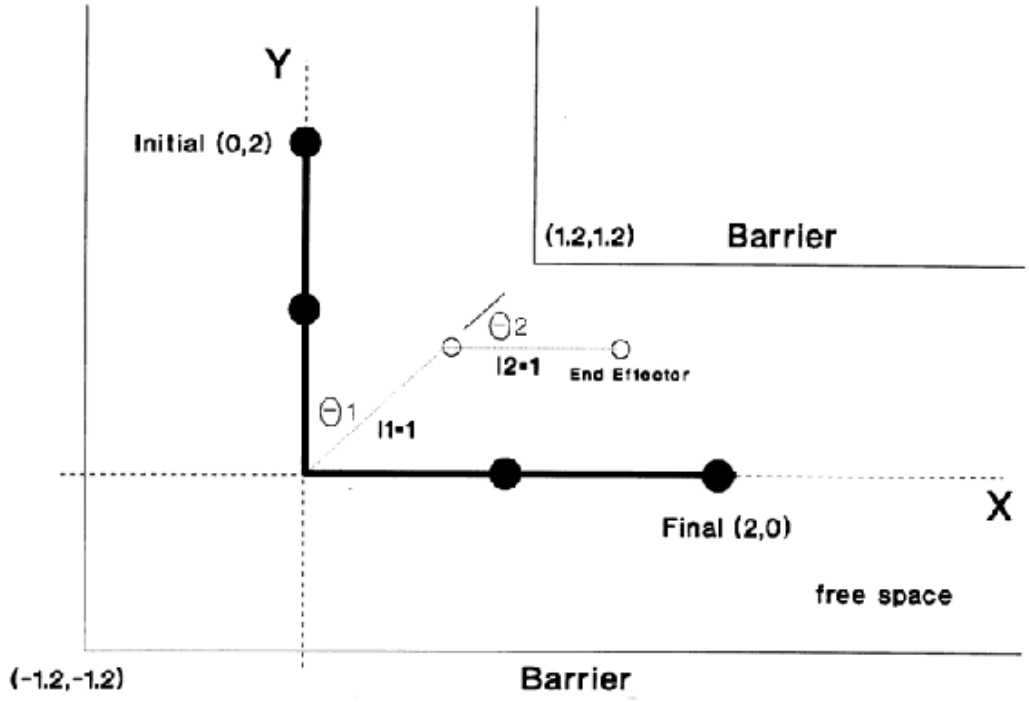
Note: it is important to mention computation speed of the program. For question 1, with $\Delta = 0.05$**,** the computation time of the C-Space & W-Space is approximately = 2.7 minutes; run on MATLAB2007b on Pentium 4 (2.6 GHz).

Question 2

Here we are going to extend the problem of Q1 to larger obstacles. A planning is required to maneuver the arm from:

- End-effector initial position of (x_i, y_i) = (0, 2)
- End-effector final position of (x_f, y_f) = (2, 0)

Taking mind of the obstacles in the environment. The environment is shown in the below figure



The obstacles now are not only point obstacles. You can see the barriers described in above figure.

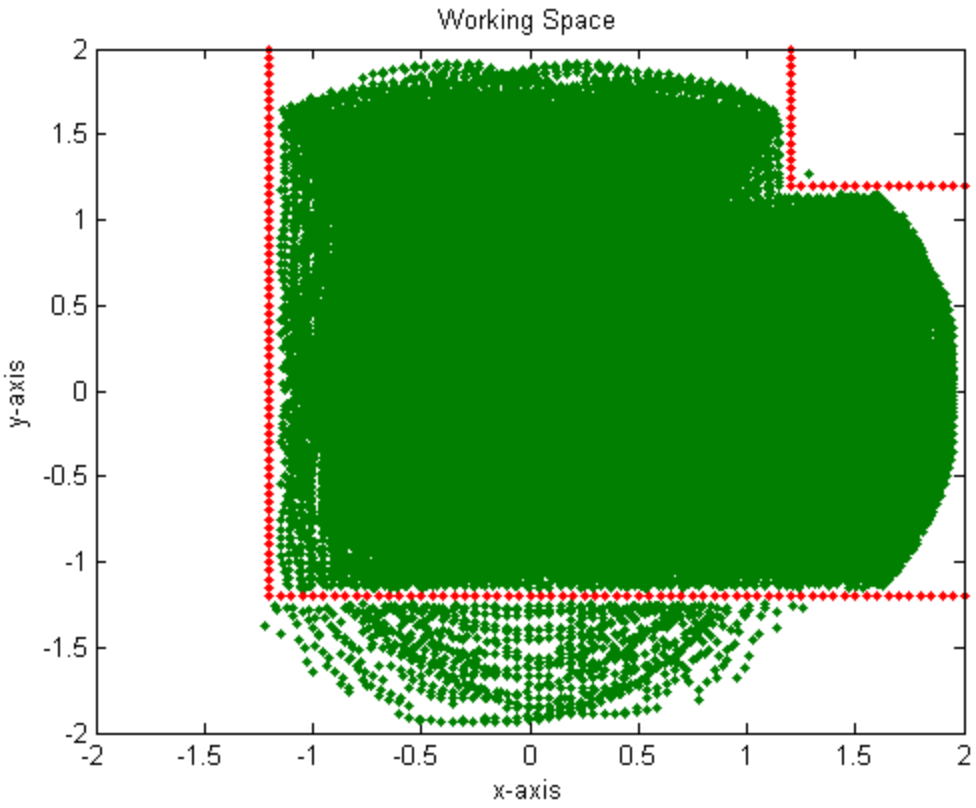
With initial & final end-effector values we can compute initial & final values of the configuration variables. This is computed to have:

$$(\theta_{1i}, \theta_{2i}) = (0, 0)$$

$$(\theta_{1f}, \theta_{2f}) = (\frac{\pi}{2}, 0)$$

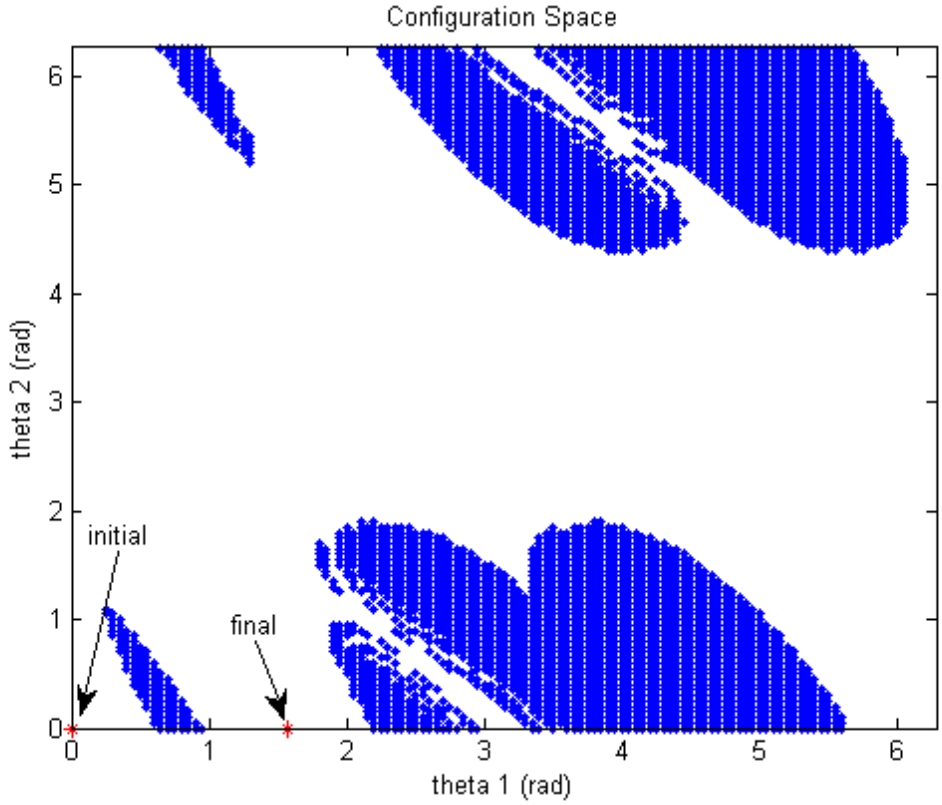
C-Space & W-Space

So, by using the same program explained in Question 1 with the same resolution $\Delta = \mathbf{0.05}$, we can construct the plots for the C-Space and the W-Space of this specific problem. By injecting the Barrier information to the program we can have the W-Space as shown below



You can see that the algorithm computed the W-Space with some errors in the bottom side of the barrier. But the plot clearly describes the working space of the robot arm.

In order to plan for the values of θ_1, θ_2, we have to compute the C-Space with '**C-Space Obstacles'**. The C-Space of this problem can be shown in the below plot



With dark areas corresponds to the 'C-Space Obstacle'. The figure above also plots the initial and final points of (θ_1, θ_2). **Note**: the computation time required for this problem on the same PC took about 1.5 minutes.

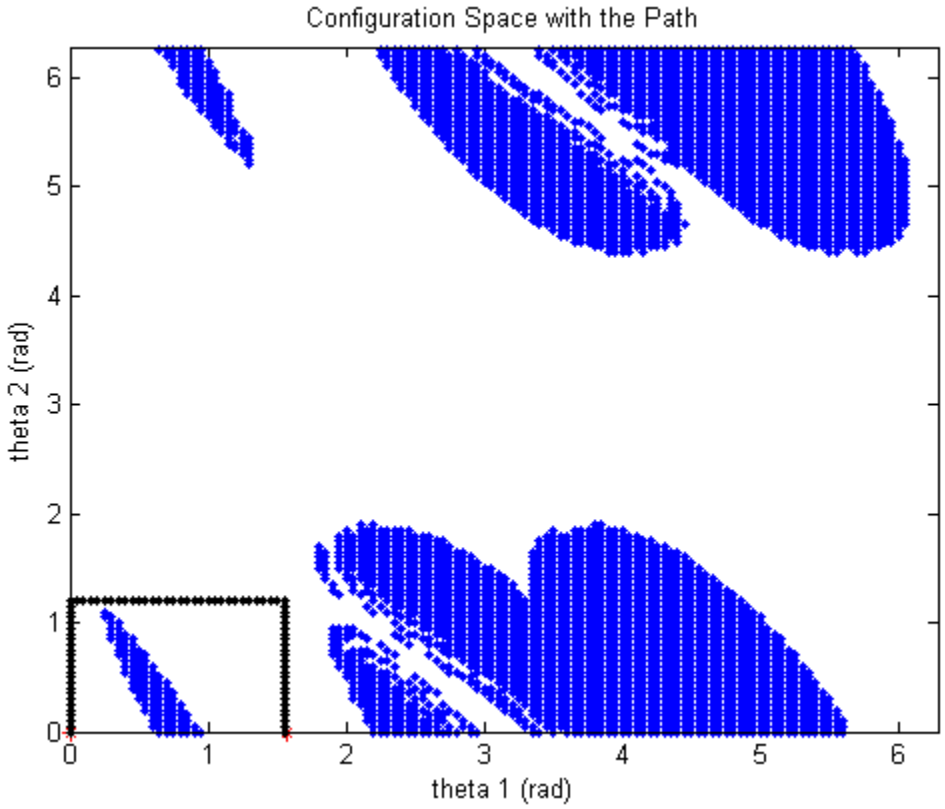
Motion Planning

After observing the above C-Space, we can plan for different scenarios of the motion. Here, only **one scenario** is tested.

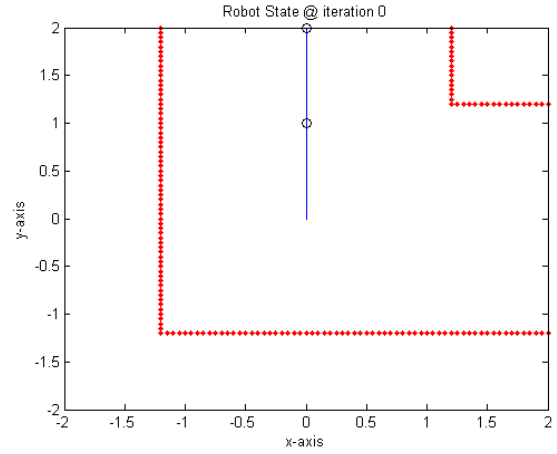
1. The logical plan is to manipulate only $\theta_2$at first. $\theta_2 = 0 \rightarrow 1.2$
2. Then, we manipulate only $\theta_1 = 0 \rightarrow \frac{\pi}{2}$
3. Then, we manipulate only $\theta_2 = 1.2 \rightarrow 0$

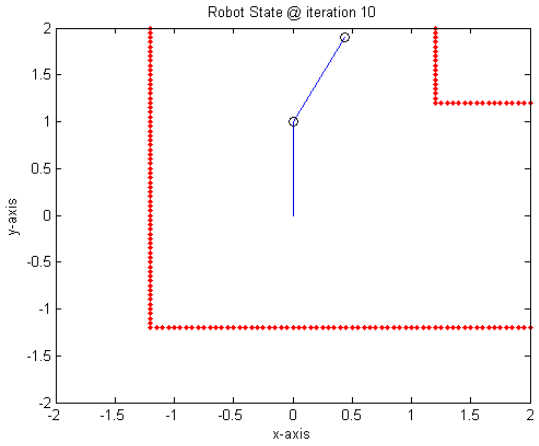
Infinite other scenarios could be used as long as the path does not touch the obstacle areas.

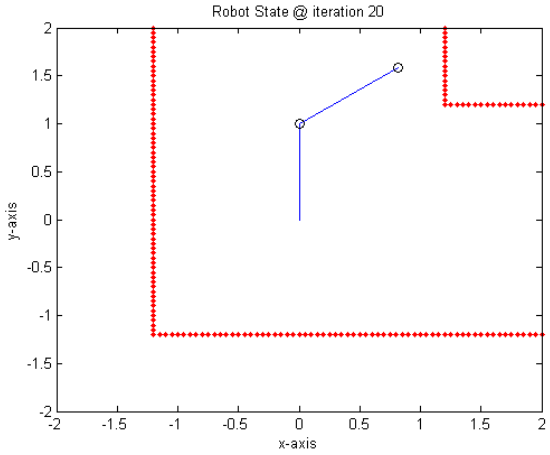
So, using MATLAB, we can construct the path for the motion and showing them in both C-Space & W-Space. Below you can see the designed path in C-Space as described above:

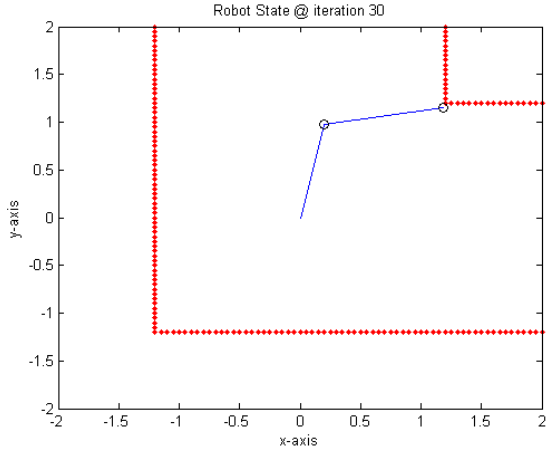


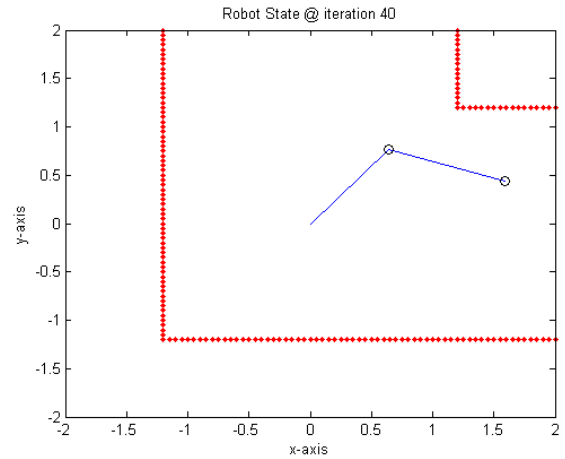
So, performing this path will move robot joint to have the motion as shown in the next figures. You can see arm state at different iterations.

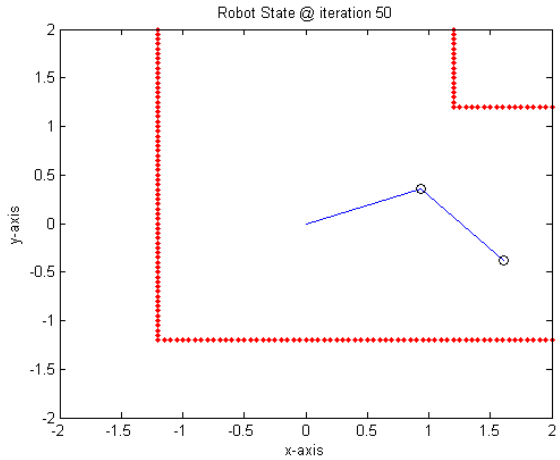


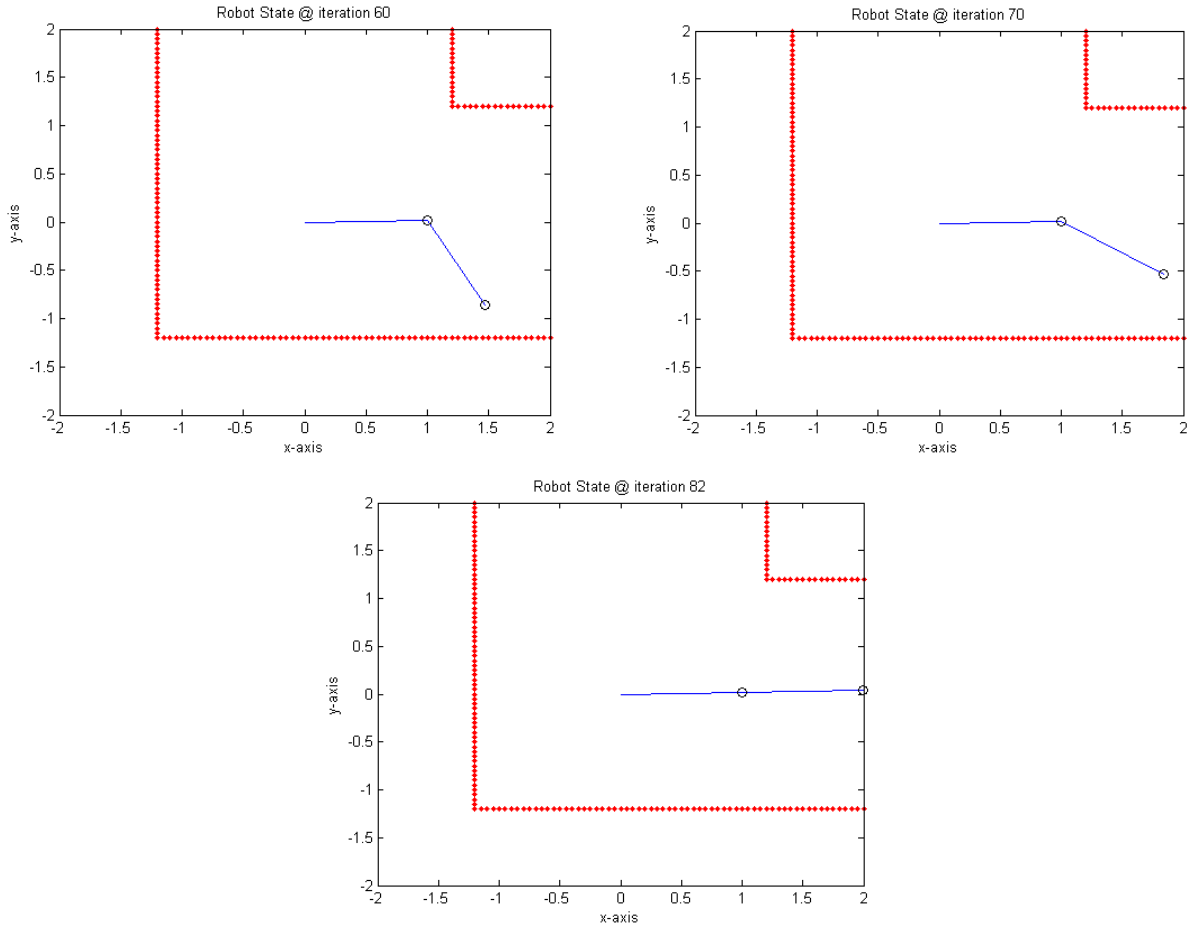












You can see above that the arm moves to the target without hitting any barrier.

Appendix

Here you can see the MATLAB code of the program

```
clear
clc
tcompu1=cputime;
%% Links Data
L1=1;
L2=1;
%% Initial Data
Xmin=-(L1+L2); %minimum at x-axis
Ymin=-(L1+L2); %minimum at y-axis
Xmax=(L1+L2); %maximum at x-axis
Ymax=(L1+L2); %maximum at y-axis
th1min=0; %manimum at theta1
th2min=0; %minimum at theta2
th1max=2*pi; %maximumat theta1
th2max=2*pi; %maximum at theta2
res=.05; %program resolution
pres=res*1;

domW=[Xmin,Xmax,Ymin,Ymax];
domC=[th1min,th1max,th2min,th2max];

th1=th1min:res:th1max;
th2=th2min:res:th2max;

dummy1=0;
dummy2=0;

XX1=[];
YY1=[];
XX2=[];
YY2=[];
th1OK=[];
th2OK=[];
th1NO=[];
th2NO=[];

%% Barriers & Obstacles Data
S1x=1.2:res:Xmax;
S1y=1.2*ones(size(S1x));

S2y=1.2:res:Ymax;
S2x=1.2*ones(size(S2y));

S3x=-1.2:res:Xmax;
S3y=-1.2*ones(size(S3x));

S4y=-1.2:res:Ymax;
S4x=-1.2*ones(size(S4y));

Sx=[S1x S2x S3x S4x];
Sy=[S1y S2y S3y S4y];
```

```
%% Spaces Construction
k=1;
for i=1:length(th1)
    for j=1:length(th2)

    x1(k)=L1*sin(th1(i));
    y1(k)=L1*cos(th1(i));

    x2(k)=x1(k)+L2*sin(th1(i)+th2(j));
    y2(k)=y1(k)+L2*cos(th1(i)+th2(j));

    xp=0:(sign(x1(k))*pres):x1(k);
    xpp=x1(k):(sign(x2(k)-x1(k))*pres):x2(k);

    yp=(y1(k)./x1(k)).*xp;
    ypp=((y2(k)-y1(k))./(x2(k)-x1(k))).*(xpp-x1(k))+y1(k);

for ks=1:length(Sx);
    for kd=1:length(xp)
        if((xp(kd)<=Sx(ks)+res && xp(kd)>=Sx(ks)-res) && (yp(kd)<=Sy(ks)+res &&
yp(kd)>=Sy(ks)-res))

        dummy1=1;
        th1NO(k)=th1(i);
        th2NO(k)=th2(j);
        break
        end
    end
end

for ks=1:length(Sx);
    for kd=1:length(xpp)
        if((xpp(kd)<=Sx(ks)+res && xpp(kd)>=Sx(ks)-res) && (ypp(kd)<=Sy(ks)+res &&
ypp(kd)>=Sy(ks)-res))

        dummy2=1;
        th1NO(k)=th1(i);
        th2NO(k)=th2(j);
        break
        end
    end
end

if(dummy1 || dummy2)
    dummy1=0;
    dummy2=0;
else
%     th1OK=[th1OK th1(i)];
%     th2OK=[th2OK th2(j)];
    XX1=[XX1 xp];
    YY1=[YY1 yp];
    XX2=[XX2 xpp];
    YY2=[YY2 ypp];
end
    k=k+1;
    end
k=k+1;
end
```

```
%% Plots
plot(XX1,YY1,'.',XX2,YY2,'.',Sx,Sy,'.r')
axis(domW)
xlabel('x-axis')
ylabel('y-axis')
title('Working Space')

% sres=1;
% th2_1=0:res*sres:1.2;
% th1_1=zeros(size(th2_1));
%
% th1_2=0:res*sres:pi/2;
% th2_2=th2_1(end)*ones(size(th1_2));
%
% th2_3=th2_1(end):-res*sres:0;
% th1_3=th1_2(end)*ones(size(th2_3));
%
% % thPlan=[th2plan th1plan];
%
% th1p=[th1_1 th1_2 th1_3];
% th2p=[th2_1 th2_2 th2_3];
%
% xL=[];
% yL=[];
% for kkk=1:length(th1p)
%
%     x1p(kkk)=L1*sin(th1p(kkk));
%     y1p(kkk)=L1*cos(th1p(kkk));
%
%     x2p(kkk)=x1p(kkk)+L2*sin(th1p(kkk)+th2p(kkk));
%     y2p(kkk)=y1p(kkk)+L2*cos(th1p(kkk)+th2p(kkk));
%
%     xp1=0:(sign(x1p(kkk))*res):x1p(kkk);
%     xp2=x1p(kkk):(sign(x2p(kkk)-x1p(kkk))*res):x2p(kkk);
%
%     yp1=(y1p(kkk)./x1p(kkk)).*xp1;
%     yp2=((y2p(kkk)-y1p(kkk))./(x2p(kkk)-x1p(kkk))).*(xp2-x1p(kkk))+y1p(kkk);
%
%   xL=[xL xp1 xp2];
%   yL=[yL yp1 yp2];
% end
%
% ,figure
% ik=82;
% plot([0 x1p(ik) x2p(ik)],[0 y1p(ik)
y2p(ik)],Sx,Sy,'.r',x1p(ik),y1p(ik),'ko',x2p(ik),y2p(ik),'ko')
% axis(domW)
% xlabel('x-axis')
% ylabel('y-axis')
% title('Robot State @ iteration 82')
 ,figure
plot(th1NO,th2NO,'.',0,0,'*r',pi/2,0,'*r',th1p,th2p,'k.')
axis(domC)
xlabel('theta 1 (rad)')
ylabel('theta 2 (rad)')
title('Configuration Space with the Path')
tcompu2=cputime;
computation_time=tcompu2-tcompu1
```