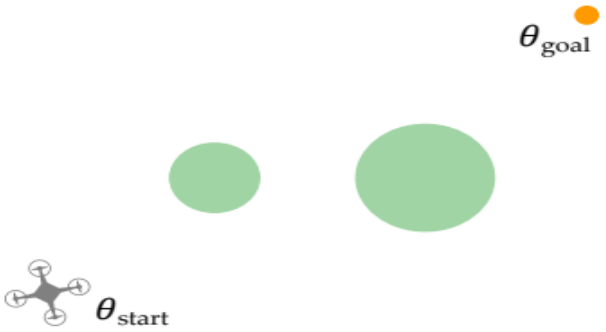
1 Potential Fields



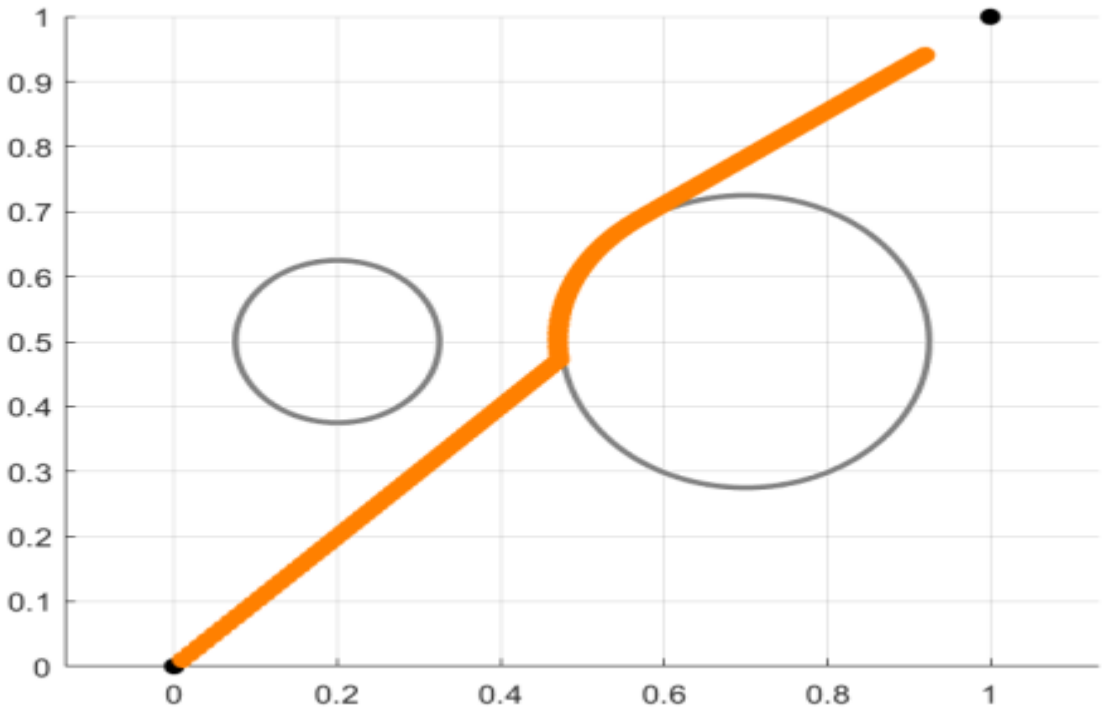
In this problem you will use potential fields to get a motion plan for the 2-DoF environment shown above. Here the drone's position is $\theta = [x, y]^T$.

1.1 (15 points)

Implement the potential fields approach:

- Set $\theta_{start} = [0, 0]^T$ and $\theta_{goal} = [1, 1]^T$
- The first obstacle has center $c_1 = [0.2, 0.5]^T$ and radius $r_1 = 0.125$
- The second obstacle has center $c_2 = [0.7, 0.5]^T$ and radius $r_2 = 0.225$
- **Hint:** Start with a low learning rate α in your gradient descent algorithm. The result shown below was obtained with $\alpha = 0.01$.

The motion plan must reach a final position within 0.1 units of the goal. Turn in your code and a plot of the result using **Publish** in Matlab. Visualize the obstacles in your plot: your solution should look like the example below.




```
clear
```

```
close all

% Start and goal environments
theta_start= [0; 0];
theta_goal = [1; 1];

% Obstacle parameters
obs_c21= [0.2; 0.5];
obs_r21= 0.125;
obs_c22= [0.7; 0.5];
obs_r22= 0.225;

% Visualizing the environment
figure
grid on
hold on
axis([0, 1, 0, 1])
axis equal
viscircles(obs_c21', obs_r21, 'Color', [0.5, 0.5, 0.5]);
viscircles(obs_c22', obs_r22, 'Color', [0.5, 0.5, 0.5]);
plot(0, 0, 'ko', 'MarkerFaceColor', 'k')
plot(1, 1, 'ko', 'MarkerFaceColor', 'k')

%variables
alpha= 0.01;
epsilon = 0.1;
delta= 0.01;

% initial trajectory
theta(:,1) = theta_start;
t=1;
del_Unet=1;

while norm(del_Unet)> epsilon
    del_Ux= U_theta(theta(:,t) + [delta;0]);
    del_Uy= U_theta(theta(:,t)+ [0;delta]);
    del_U= U_theta(theta(:,t));

    del_Unet=[del_Ux-del_U; del_Uy-del_U]/delta;

    theta(:,t+1)= theta(:,t)- alpha*del_Unet;
    t=t+1;
end
```

```
    0
    0
    0
    0
    0
    0
    0
    0
```

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0.0106
0
0
0.0133
0
0
0.0065
0
0
0.0198
0
0
0
0
0
0.0196
0
0
0
0
0
0.0138
0
0
0.0016
0
0
0.0175
0
0
0
0
0
0.0090
0
0
0.0043
0
0
0.0090

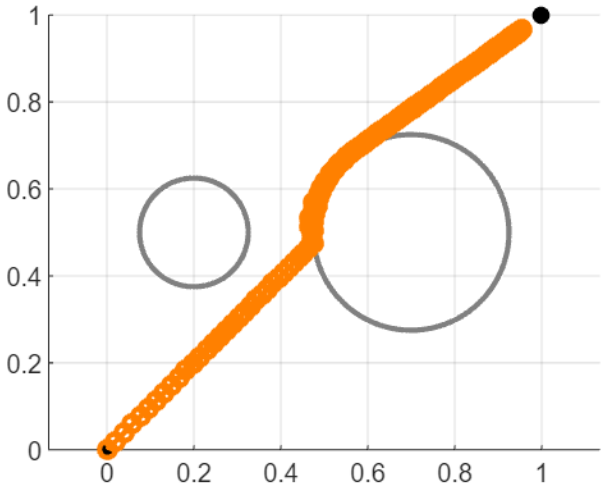
0
0
0.0031
0
0
0.0079
0
0
0.0031
0
0
0.0059
0
0
0.0034
0
0
0.0041
0
0
0.0032
0
0
0.0029
0
0
0.0024
0
0
0.0019
0
0
0.0013
0
0
8.2385e-04
0
0
3.6303e-04
0
0
4.6431e-05
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

```
     0
     0
     0
     0
     0
     0
     0
     0
```

```
grid on
hold on
axis equal
plot(theta(1,:), theta(2,:), 'o-',...
    'Color', [1, 0.5, 0], 'LineWidth', 2);
```




```
function U = U_theta(theta)
    beta=2;
    gamma=1;
    theta_goal = [1; 1];
    obs_c21= [0.2; 0.5];
    obs_r21= 0.125;
    obs_c22= [0.7; 0.5];
    obs_r22= 0.225;

    Urep1=0;
    Urep2=0;


    Uatt=0.5*beta*norm(theta_goal-theta)^2;

    if norm(obs_c21-theta)<= obs_r21
```

```
        Urep1=0.5* gamma*((1/norm(obs_c21-theta))- (1/obs_r21))^2;
    end
    if norm(obs_c22-theta)<= obs_r22
        Urep2= 0.5*gamma*((1/norm(obs_c22-theta))- (1/obs_r22))^2;
    end

    Urep= Urep1+ Urep2;
    disp(Urep)
    U= Uatt + Urep;
end
```