Assignment 2, Problem 9 Computation Teamwork

February 7, 2018

Group: Austin Wang, Lawrence Moore, Joe Higgins

In [1]:
```
# Import necessary modules
import cvxpy as cvx
import numpy as np
```

0.1 Introduction

In this notebook, we will be solving the two-dimensional sensor localization problem with two sensors and three anchors. We assume that we know the distances between sensor 1 and anchors 1 and 2, and the distances between sensor 2 and anchors 2 and 3. In addition, we know the distance between the two sensors. Our goal is to use this information to determine the location of the sensors.

We will formulate this problem as both an SOCP relaxation problem and an SDP relaxation problem to analyze how the two methods compare.

0.2 Problem Data

In [2]:
```
# Define anchors
anchors = np.matrix([
    [ 1, 0],
    [-1, 0],
    [ 0, 2]
])

# Number of dimensions
n = 2

# Case 1: Sensors are both in each other's convex hulls
s1 = np.matrix([
    [ 0, .1],
    [0, 0.5]])

# Case 2: Both sensors in convex hull of anchors,
# only one sensor in convex hull of the other
s2 = np.matrix([
    [ 0, 0.1],
```

```
        [0.05, 1.8]])

    # Case 3: Only one sensor in convex hull of anchors,
    # and same sensor outside convex hull of the other
    s3 = np.matrix([
        [ 0, .1],
        [-.5, 10]])

    # Case 4: Both in convex hull of anchors, but not in
    # each others' convex hulls
    s4 = np.matrix([
        [ -.5, .5],
        [.5, 0.5]])
```

In [3]:
```
# Calculate distances (case 1 used here as an example)

d0 = list(map(lambda a: np.linalg.norm(s1[0, :] - a), anchors))
d1 = list(map(lambda a: np.linalg.norm(s1[1, :] - a), anchors))
d_both_sensor = np.linalg.norm(s1[0, :] - s1[1, :])
```

0.3 Part 1: SOCP Relaxation Problem Experimentation

In [4]:
```
# Construct the problem.
x = cvx.Variable(2,n)
objective = cvx.Minimize(0)

x1 = x[0,:]
x2 = x[1, :]

constraints = [cvx.norm(x1 - anchors[0]) <= d0[0],
               cvx.norm(x1 - anchors[1]) <= d0[1],
               cvx.norm(x1 - x2) <= d_both_sensor,
               cvx.norm(x2 - anchors[1]) <= d1[1],
               cvx.norm(x2 - anchors[2]) <= d1[2]
              ]

prob = cvx.Problem(objective, constraints)
result = prob.solve(solver = 'MOSEK')
```

0.4 Part 2: SDP Relaxation Problem Experimentation

The following functions will be useful in the code that follows:

In [5]:
```
def sum_elem_product(A,B):
    return cvx.sum_entries(cvx.mul_elemwise(A, B))

def col_vec_4elem(a,b,c,d):
    return np.matrix([[a],[b],[c],[d]])
```

In [6]:
```
# Constraints will define the proper structure for Z. The optimal value will be
# located in the top right 2x2 matrix of Z.
Z = cvx.Semidef(4)

# Objective does not matter;
# We are simply concerned with solving feasibility conditions
objective = cvx.Minimize(0)

v0 = col_vec_4elem(1,0,0,0)
v1 = col_vec_4elem(0,1,0,0)
v2 = col_vec_4elem(1,1,0,0)

a0 = col_vec_4elem(anchors[0,0], anchors[0,1], -1, 0)
a1 = col_vec_4elem(anchors[1,0], anchors[1,1], -1, 0)

a2 = col_vec_4elem(anchors[1,0], anchors[1,1], 0, -1)
a3 = col_vec_4elem(anchors[2,0], anchors[2,1], 0, -1)

v3 = col_vec_4elem(0, 0, 1, -1)

constraints = [
    sum_elem_product(v0*np.transpose(v0), Z) == 1,
    sum_elem_product(v1*np.transpose(v1), Z) == 1,
    sum_elem_product(v2*np.transpose(v2), Z) == 2,
    sum_elem_product(a0*np.transpose(a0), Z) == cvx.square(d0[0]),
    sum_elem_product(a1*np.transpose(a1), Z) == cvx.square(d0[1]),
    sum_elem_product(a2*np.transpose(a2), Z) == cvx.square(d1[1]),
    sum_elem_product(a3*np.transpose(a3), Z) == cvx.square(d1[2]),
    sum_elem_product(v3*np.transpose(v3), Z) == cvx.square(d_both_sensor)
]

prob = cvx.Problem(objective, constraints)
result = prob.solve(solver = 'MOSEK')

# Optimal solution
x_1_star = [Z[0,2].value, Z[1,2].value]
x_2_star = [Z[0,3].value, Z[1,3].value]
```

0.5 Case 1: Sensors are Both in Each Other's Convex Hulls

True Sensor Location 1: [0., 0.1]
True Sensor Location 2: [0., 0.5]

0.5.1 SOCP

Result: **SUCCESS**
SOCP optimal sensor 1 location: [-1.57683210e-08, 1.00000278e-01]
SOCP optimal sensor 2 location: [-2.60824842e-04, 5.00000142e-01]

0.5.2 SDP

Result: **SUCCESS**
SDP optimal sensor 1 location : [0.0, 0.09999999558367077]
SDP optimal sensor 2 location : [-1.6547981873671347e-09, 0.5000000008273991]

0.6 Case 2: Only One Sensor in Convex Hull of the Other, Both Sensors in Convex Hull of Anchors

True Sensor Location 1: [0., 0.1]
True Sensor Location 2: [0.5, 0.5]

0.6.1 SOCP

Result: **FAILURE**
SOCP optimal sensor 1 location: [0.00033328, 0.02926632]
SOCP optimal sensor 2 location: [-0.09786292, 0.53287777]

0.6.2 SDP

Result: **SUCCESS**
SDP optimal sensor 1 location : [0.0, 0.09999997299798209]
SDP optimal sensor 2 location : [0.49999998983139116, 0.5000000050843044]

0.7 Case 3: Only One Sensor Outside Convex Hull of the Other, Same Sensor Outside of Convex Hull of Anchors

True Sensor Location 1: [0., 0.1]
True Sensor Location 2: [-0.5, 10.]

0.7.1 SOCP

Result: **FAILURE**
SOCP optimal sensor 1 location: [-0.00014567, 0.00163918]
SOCP optimal sensor 2 location: [-0.30427143, 0.77674806]

0.7.2 SDP

Result: **FAILURE**
SDP optimal sensor 1 location: [0.0, -0.046121775533644674]
SDP optimal sensor 2 location: [0.9782381058091758, 9.260880947095412]

0.8 Case 4: Both Sensors in Convex Hull of Anchors, Not in Each Others' Convex Hulls

True Sensor Location 1: [-0.5, 0.5]
True Sensor Location 2: [0.5, 0.5]

0.8.1 SOCP

Result: **FAILURE**
SOCP optimal sensor 1 location: [-0.44050297, 0.14807826]
SOCP optimal sensor 2 location: [-0.40809585, 0.74565233]

0.8.2 SDP

Result: **FAILURE**
SDP optimal sensor 1 location: [-0.5000000000000001, 0.2071082576424906]
SDP optimal sensor 2 location: [0.21648819130490615, 0.6417559043475469]

0.9 Conclusion

In case 1, in which both sensors are in the convex hull of the anchors and in the convex hull of each other (with their two specific anchors), both SOCP and SDP find the correct solutions. This seems to be the case because in this situation, the "forces" corresponding to the Lagrangian multipliers are able to cancel each other out and lead to a rigid system. The resulting problem is similar to the problem seen in the Homework 1, except with two systems instead of one.

In case 2, we observe a similar phenomenon to that which we observed in Homework 1. SOCP fails when one of the sensors lies outside the convex hull of the other, while SDP succeeds.

In case 3 and case 4, we find that both SOCP and SDP fail to find the correct sensor locations. This may be the case because in those specific configurations, there are other possible locations that still satisfy the distance constraints, meaning the forces cannot cancel out and the system is not rigid; there may be another configuration of sensors that lead to the same distances.

It may be possible that a rigid solution exists when both sensors are outside the convex hull of the anchors and outside of the convex hull of each other, as the forces could again cancel out. However, we had difficulty finding such a set of points during our experimentations.