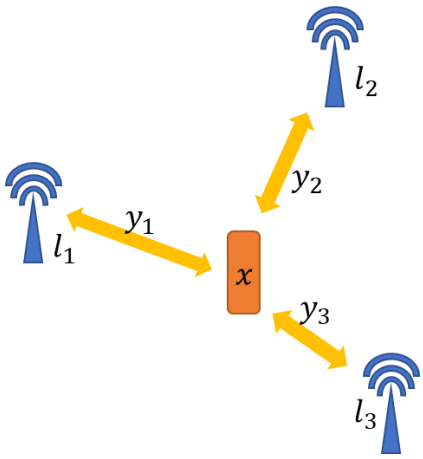
CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Computing@CIT

DECISION ANALYTICS.

Lab02: Least-squares parameter estimation



BACKGROUND.

In Lecture 3 we looked at the radio location problem. The goal is to find the position of a mobile device given the distance measurements between the device and the surrounding cell towers. This technology is for instance being used by emergency services to pinpoint the exact location of a caller.

(In a more general formulation with slightly different constraints it is also the underlying approach for satellite-based positioning systems, such as GPS)

Task 1.

Let the locations of the three visible cell towers be given by

$$l_1 = \begin{pmatrix} 5 \\ 25 \end{pmatrix}$$

$$l_2 = \begin{pmatrix} 32 \\ 22 \end{pmatrix}$$

$$l_3 = \begin{pmatrix} 29 \\ 5 \end{pmatrix}$$

As a first guess, we can assume that the mobile device is somewhere in the middle between the three. Write a Python program that calculates the initial guess for the position x_0 to be the average between the three cell tower positions.

(Hint: Use https://numpy.org/ for vector and matrix operations; the average position can be calculated using the function https://docs.scipy.org/doc/numpy/reference/generated/numpy.mean.html)

Task 2.

In lecture 3 we derived the linearized constraint $Ax + By = c$ for this problem with the coefficient matrices

$$A = 2\begin{pmatrix} x_0^T - l_1^T \\ x_0^T - l_2^T \\ x_0^T - l_3^T \end{pmatrix}$$

$$B = -I_3$$

$$c = \begin{pmatrix} x_0^T x_0 - l_1^T l_1 \\ x_0^T x_0 - l_2^T l_2 \\ x_0^T x_0 - l_3^T l_3 \end{pmatrix}$$

Write a Python program that calculates these matrices.

(Hint: use https://docs.scipy.org/doc/numpy/reference/generated/numpy.matmul.html for matrix multiplication and https://docs.scipy.org/doc/numpy/reference/generated/numpy.eye.html to create the identity matrix)

Task 3.

Now we consider the measured distances between the cell towers and the mobile device

$$\bar{d}_1 = 17$$

$$\bar{d}_2 = 13$$

$$\bar{d}_3 = 15$$

The linearized constraints derived in lecture 3 were based on squared distances ($\bar{y}_i = \bar{d}_i^2$) and we determined that the minimum of $(y - \bar{y})^2$ under these constraints could be obtained by solving the linear equation system

$$\lambda^T A = 0$$

$$2(y - \bar{y}) + \lambda^T B = 0$$

$$Ax + By - c = 0$$

Write a Python program that builds this 3 sets of linear equations:

(1) Create the 2x8-matrix $M_1 = [0_{2\times 2} \quad 0_{2\times 3} \quad A^T]$ and the vector $b_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ corresponding to the first linear constraint

(2) Create the 3x8-matrix $M_1 = [0_{3\times 2} \quad 2I_3 \quad B^T]$ and the vector $b_2 = 2\bar{y}$ corresponding to the second linear constraint

(3) Create the 3x8-matrix $M_1 = [A \quad B \quad 0_{3\times 3}]$ and the vector $b_3 = c$ corresponding to the third linear constraint

(Hint: use https://docs.scipy.org/doc/numpy/reference/generated/numpy.append.html to stack matrixes, https://docs.scipy.org/doc/numpy/reference/generated/numpy.transpose.html to transpose, and https://docs.scipy.org/doc/numpy/reference/generated/numpy.zeros.html for creating matrices containing zeros)

Task 4.

Having constructed all the elements, we now want to solve the equation system to obtain the position of the mobile device from the distance measurements. To do this we need to stack the equation systems derived in the previous task and obtain a single 8x8 linear equation system as follows:

$$\begin{bmatrix} M_1 \\ M_2 \\ M_3 \end{bmatrix} \begin{pmatrix} x \\ y \\ \lambda \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

Write a Python program that solves this linear equation system and computes the 8 vector

$$s = \begin{pmatrix} x \\ y \\ \lambda \end{pmatrix}$$

Finally, extract the position of the mobile device x from the solution vector. If you followed all the steps above correctly you should obtain the actual position to be

$$x = \begin{pmatrix} 20 \\ 17 \end{pmatrix}$$

(Hint: use https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.solve.html to solve the linear equation system)

Task 5.

In task 3 we used the exact distances between the mobile device and the cell towers as input. In reality measurements are never exact, and the approach presented above does not require exact distances to work.

To see this, add some random noise to the distances d_1, d_2, d_3 and run the algorithm again. Further to that, run the algorithm repeatedly always using the resulting position as initial position for the next iteration ($x_0 \coloneqq x$). Observe, how this search procedure converges towards the final position finding the configuration that minimises the residual error.

(Hint: to generate Normal distributed random number use the function https://docs.scipy.org/doc/numpy-1.15.1/reference/generated/numpy.random.randn.html)