Homework 4: robotic navigation

Preamble: You have built a robot for navigating in an indoor environment. The environments you consider have flat floors, so that the whole task is always two-dimensional. The robot has a sensing capability, as it can sense a location with respect to special beacons. You have given a robot to a friend, who uses it to map different interiors. To do the mapping, your friend places beacons at the perimeter of the interior, and then moves the robot through some trajectory.

Every so often the robot stops, uses its compass to align itself with the direction to the north and then tries to sense the location of the beacons (relative to the robot's current position). Typically, it manages to locate several closest beacons. Your friend has conducted three experiments in three different interiors. You need to recover the location of beacons and the locations of the robot (where the measurements have been taken).

From each experiment you get four arrays of measurements: **'beacons', 'robots', 'x', 'y'**. It is interpreted as follows: while the robot is in the position number **'robots(i)'**, it senses the beacon number **'beacons(i)'**, and the beacon is displaced by **'x(i)'** and **'y(i)'** from the robot's position. For each robot position, the robot senses several beacons. This data should be sufficient to recover the positions of the robot and of the beacons (upto a global translation, that does not matter for the purposes of your friend).

Experiment one (task1.mat) - 1 point: This is the simplest environment (a room with a simple shape), the robot sensor works well (although a certain amount of Gaussian noise is present). The reconstruction can be reduced to solving a simple least-squares problem. *Hint*: use linear least-squares. Avoid using the explicit inversion of matrix. Instead, use backslash ('mrdivide') to solve equations. Note that backslash adds a (small) regularization to the system. Try using inversion ('inv') to see how the reconstruction performs without regularization.

Experiment two (task2.mat) - 4 points: This is a more complex environment. On top of that, there are strong electrical currents under the floor, so that the compass on the robot makes significant errors, so that the robot is not perfectly oriented. Thus, in addition to robot positions you would also need to recover

robot orientations. *Hint and instructions*: you can still try to solve the problem as if the robot is perfectly oriented. You can then use what you obtain as an initialization to the non-linear initialization (angles can be initialized at zero). For the non-linear optimization, you should use non-linear least-squares. You can then either code Levenberg-Marquardt method yourself or figure out how to use MATLAB's 'lsqnonlin' (which also uses Levenberg-Marquardt). In the latter case, to get the points, your final version must provide the MATLAB's routine with a Jacobian computed (analytically) by your code.

Experiment three (task3.mat) - **6 points**: This is another complex environment. On top of the complications from the previous experiment, your robot beacon identification system is now working imperfectly, so that every so often (say, in 3% cases), the robot would confuse the number of the beacon that it observes. You have to make your reconstruction algorithm robust to such mistakes. *Instruction:* use non-linear optimization 'fminunc' in Matlab (you get 2 points for any implementation that produces an accurate reconstruction). You are recommended to supply it with an analytically-computed gradient (+2 points). Use the option 'Large Scale=off' to make Matlab to use a quasi-Newton BFGS method, play with the number of iterations that are allowed ('MaxIter') to see the effect; you can also try to use 'Large Scale=on' for an algorithm that tries to compute the Hessian more accurately than quasi-Newton, you would need to provide at least a Hessian sparsity pattern to make this approach fast. You get +2 points if your approach devised this way will work faster or comparable with quasi-Newton.

Output. For each experiment, your pdf file should include:

- A printout of the reconstruction: use 'scatter' to plot it. E.g.: *scatter(robotpos(1,:), robotpos(2,:),'r','filled'); hold on; scatter(beaconpos(1,:), beaconpos(2,:),'g','filled'); saveas(gcf,'out1.png').*
- An explanation of what you have tried and what you have observed.

Of course, your code should be attached.