CS 4610/5335: Robotic Science and Systems (Spring 2021) Robert Platt
Northeastern University

HW 5: Simultaneous Localization and Mapping

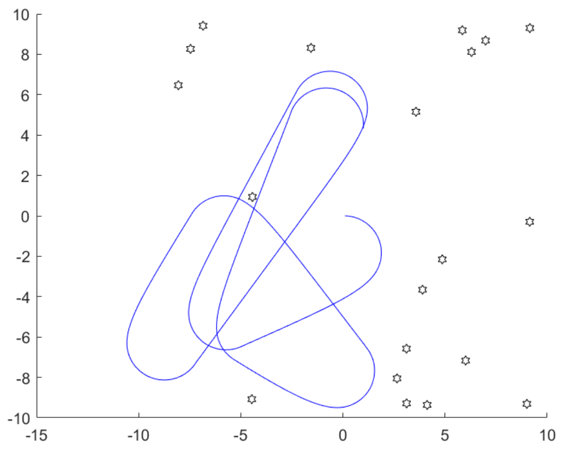
Please remember the following policies:

- Submissions should be made electronically via the Canvas. Please ensure that your solutions for both the written and/or programming parts are present and zipped into a single file.
- Solutions may be handwritten or typeset. For the former, please ensure handwriting is legible.
- You are welcome to discuss the programming questions (but *not* the written questions) with other students in the class. However, you must understand and write all code yourself. Also, you must list all students (if any) with whom you discussed your solutions to the programming questions.

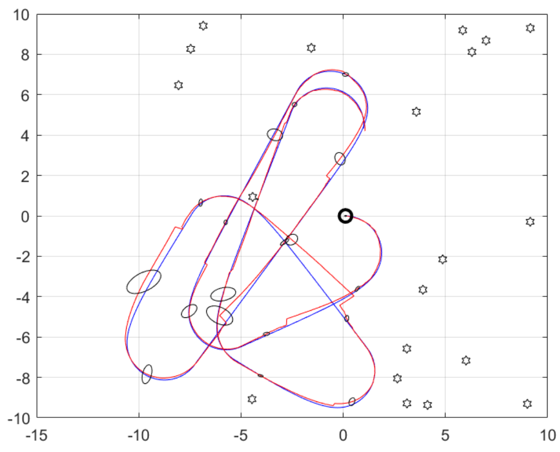
Download the starter code from Piazza (`hw5.zip`). In this file, you will find:

- `hw5.m`: Main function. Call `hw5(<questionNum>)` with an appropriate question number (0–10) to run the code for that question. Do not modify this file! (Feel free to actually make changes, but check that your code runs with an unmodified copy of `hw5.m`.)
- `E0.m` – `E3.m`: Code stubs that you will have to fill in for the respective questions.
- `e1.mat` – `e3.mat`, `e3_r8.mat`: Data files for each question.
- `parameters.m`: Parameters for EKF-SLAM.
- `visualize.m, visualize_histogram.m, visualize_path.m, visualize_map.m, average_error.m`: Helper functions for visualizing results and computing errors.

Part 1



E0: Ground truth map and trajectory.



E0: Robotics toolbox EKF localization solution (in red).

E0. **2 points.** The Robotics toolbox actually provides an implementation of EKF-based SLAM already. Read through the code examples in Chapter 6 and learn how to use the existing functions provided by the toolbox. Run the code that is listed in the textbook for localization, mapping, and SLAM. **Use a maximum range of 4 and field-of-view of $[-\frac{\pi}{2}, \frac{\pi}{2}]$ for the sensor.** For example, the code for localization should be very similar to that shown on page 163 (the example code already has the correct maximum range and field of view). Return the EKF objects for each and save them in your workspace, because they will provide a way for you to check your own implementation in the upcoming questions.

If you just run the code from the textbook, you will probably not get the same map and trajectory as illustrated in Figure 6.7 on page 163. This is because the map, trajectory, and noise (both odometry and sensing) are generated *randomly*. To ensure that your EKF objects use the same data that we have provided in our `*.mat` files, reset the random number generator's seed by calling `rng(0)`. In your code, should do this *every* time before you call `LandmarkMap` (which randomly generates a map) and `ekf.run` (which randomly generates the trajectory and noisy odometry/sensor readings).

If you reset the seed correctly, you should get the same map as shown in Figure 6.7, but the path is probably still different. (We actually do not know what random seed generates the trajectory shown in the textbook.) You can visualize both by calling: `visualize({}, {}, [], map, veh, 'n')`
which produces the left figure above. Visualizing the result of EKF localization gives the figure on the right.

Remember to call `startup_rvc` at the beginning of each MATLAB session.

When I ran this w/ Matlab 2020b, I got an error "unrecognized property axis for class patch". I corrected this by inserting the following line: *args = cell(1,0);* at line 100 of plot_poly.m. This fix is only needed for E0. The rest of the problems will run fine without it.

E1. **6 points.** Implement EKF-based localization, given a known map.

Observe how the `E1` function is called in `hw5.m`; data is loaded from `e1.mat`, which contains odometry and observation data (`odo`, `zind`, `z`) as well as the ground truth map and vehicle objects (`map`, `veh`). The known map is passed to `E1`, but the true trajectory is not – it is only given to `visualize` for comparison. If you saved the EKF object `ekf_l` from `E0` and pass it in to `hw5.m`, then it will also be used for comparison in `visualize`, so you can see the ground truth, your estimate, and the Robotics toolbox EKF estimate. It is okay if your estimate is different from the toolbox EKF estimate, but they should be similar.

Useful functions: `atan2`, `angdiff` – make sure you are using the Robotics toolbox version of `angdiff`, which is *different* from the one provided in the official Robotics system toolbox. Correct version:
`https://github.com/petercorke/spatial-math/blob/master/angdiff.m`

E2. **5 points.** Implement EKF-based mapping, given a known vehicle trajectory.

The given odometry readings `odo` are assumed to be accurate, unlike in the previous question. Note that the order of the landmarks in the state vector is dependent on when they are first observed in the path, therefore the state vector will be scrambled – use the `indices` vector to keep track of the alignment. Also, for the path taken, some of the landmarks will never be observed, therefore they will never appear in the state estimate.

E3. **7 points.** Mathematically derive the Jacobians in EKF-SLAM, and then implement EKF-SLAM.

The exposition in the textbook is rather light on derivations, and in particular for SLAM, is not very explicit about the Jacobians to be used in the algorithm. On a separate written document, calculate the Jacobians for the transition function (F_x and F_v), the observation function (H_x and H_w), and the insertion Jacobian (denoted Y_z in the textbook). Of course, you should arrive at the same matrices as are given in the book. You should calculate F_x and F_v from f and H_x and H_w from h. Y_z is calculated by writing down state as a function of the observation and taking a first order Taylor expansion. Also, think about the dimensions of each of these Jacobian matrices, and make sure your matrices have the correct shape.

Once you have explicitly derived the Jacobians, transfer this algorithm to code, using your code from `E1` and `E2` as the foundation.