Begin by downloading the “filters code” from the course website. Perform both activities below, but type up your answers to just one of the two parts. *If you understand this code well, you will do well on the final exam!*

1. **Particle filter**

Run the filter by executing the `demo` command and observe the demo for a few iterations. Press Ctrl+C to quit when you are finished.

   a. Change the value of the `sigma_z` variable to 1.0. What do you notice about the spread of the particles before and after the change? Explain why this shows the particle filter’s adaptivity to uncertainty.

   b. Restore `sigma_z` to its original value, and change the value of `kidnap_every` to 20. This will result in the robot getting kidnapped every 20 iterations of the algorithm. Comment on the ability of the filter to track the robot after kidnapping.

   c. Now, set `numrnd` to 20, and restart the filter, while keeping `kidnap_every` set to 20. Is performance better? Why or why not?

2. **Kalman filter**

Run the filter by executing the `demo` command, and observe the graph that is produced. The blue circles are 95% confidence intervals on the position of the ball.

   a. Explain how the Kalman filter is able to estimate both the position and the velocity of the ball, despite receiving only a noisy measurement of position. How does the `K` matrix figure into this? Comment on the elements of the `K` matrix at the first timestep of the simulation, when the velocity is not known well.

   b. How do the initial values of `pp` and `pv` affect the filter? What do they represent? Change both values to be quite small (i.e. $10^{-3}$) and re-run the filter. Is performance improved or decreased? Why?

   c. Restore the values of `pp` and `pv` to their initial values, and simulate a noisier sensor by setting the value of `rp` to 0.5. How does the filter perform? Does the quality of the estimate improve as the simulation progresses? Why or why not?