CS685 - Homework 2, due March 1st, *Jana Košecká*

1. **Moving to a point** Consider a differential drive model of the mobile robot discussed in the class with a configuration space $[x, y, \theta]^T$, where we can control linear and angular velocity of the robot v and ω. The kinematic model describing the motion of the mobile robot is as follows:

$$\begin{aligned} \dot{x} &= v\cos(\theta) \\ \dot{y} &= v\sin(\theta) \\ \dot{\theta} &= \omega \end{aligned}$$

   Implement the closed loop feed-back control law for the robot to move a desired goal poing $[x_d, y_d]^T$. Make the robot's velocity proportional to

$$v = k_v\sqrt{(x - x_d)^2 + (y - y_d)^2}$$

   and the angular velocity

$$\omega = k_\omega(\theta_d - \theta)$$

   where θ_d is the relative angle between the vehicle and the goal:

$$\theta_d = \tan^{-1}\frac{y_d - y}{x_d - x}.$$

   Write a function `[x,y,theta] = goToPoint(`x_0`,` x_d`)` which will take the initial pose, the goal coordinates and returns the trajectory of the robot. Hand in the code and plots of trajectories going from at least 5 different positions and headings to the desired goal.

2. **Line Following** Consider a feedback controller for a robot to follow a line. The line in a plane is defined as $ax + by + c = 0$. To follow a line the robot requires to street toward the line and also adjust its heading with the line. The steering towards the line can be obtained by making the angular velocity proportional to the distance of the robot from the line

$$d = \frac{[a, b, c]^T[x, y, 1]}{\sqrt{a^2 + b^2}}.$$

   The second term needs to adjust the angle such that the orientation is aligned with the line

$$\theta_d = \tan^{-1}\frac{a}{b}.$$

   Implement combined proportional control law:

$$\omega = -k_d d + k_\theta(\theta_d - \theta).$$

   Hand in the code and plots of trajectories going from at least 5 different positions and headings to the desired goal. Experiment with different choices of k_d, k_θ to generate qualitatively different trajectories and comment on the behavior.

3. **Potential Field Based Control**. Consider a point like robot in the workspace with the area $[0, 100] \times [0, 100]$. Represent the obstacles in the environment as circles with centers at $[40, 30]$ and $[70, 40]$ each with radius 5. Assume that the initial position of the robot is $x_0 = [50, 50]$. Write down a function `GoToObst`(x_g, y_g), which takes as input arbitrary goal position in the workspace and returns the trajectory which the robot followed to get to the goal using potential field based method. The parameters of the potential functions can be set as variables inside of `GoToObst`. Post the Matlab code and two plots demonstrating the capability of the robot to avoid obstacles and reach the goal.

4. Consider the *pose-to-pose* streering control for differential drive robot discussed in class. The control law is designed by transforming the problem to polar coordinates and designing the control law there. The resulting control strategy is

$$v = k_\rho \rho \quad (1)$$
$$\omega = k_\alpha \alpha + k_\beta \beta \quad (2)$$

where ρ, α, β is the configuration of the robot expressed in polar coordinates (with respect to the goal). Function `goTo.m` implements this control strategy. There is a snippet of code in the file `sample_scripts.m` showing how to use the function. All Matlab files are available in `cs.gmu.edu/~kosecka/cs485/code`. You will also need functions `plot_robot.m` and `diffDrive.m` for plotting and simulating the robot.

   (a) Experiment with different values of $k_\rho, k_\alpha, k_\beta$ which are parameters inside `goTo.m` function and describe any observations of the resulting trajectories.

   (b) Test the control law for variety of initial/goal positions. Post the plots of $x(t), y(t), \theta(t)$, the printout of the code and any observations you may have.

   (c) Test the control law for a version of a parallel parking maneuver and comment on the usability of the resulting trajectory.

5. Try and experiment with the simulations in the `sample_scripts.m` file, some of which use the `rvctoolbox` functions.