Homework 2 - Due April 17, 2020

Please turn in a PDF that includes any code used to complete the following problems. (Note that the PDF will be the only version of your code that you turn in.) Questions and deliverables that should be included with your submission are shown in **bold**.

1. (10 pts) Use the Gateaux derivative to verify that if $f(x) = \frac{1}{2}x^TQx$, and $Q = Q^T$, then $Df(x) = x^TQ$. **Include a scanned copy of your hand written solution in your PDF submission.**

2. (20 pts) This problem helps you examine the role of the dimension of a problem on how well "gradient descent" works. Suppose that $x \in \mathbb{R}^n$ and that we are optimizing a cost function of the form $f(x) = \frac{1}{2}x^TQx$ with $Q = Q^T$ and Q is a diagonal matrix close to the identity. Moreover, assume that the optimization update rule is

$$x_{i+1} = x_i - \nabla f(x_i).$$

   That is, we are changing our estimate of the minimizer by moving in the direction of the negative gradient. Lastly, assume we are going to continue an optimization until $\|\nabla f(x)\|_2 < \epsilon$, where $\epsilon = 0.1$. What happens if we change the dimension of the problem?

   To investigate this, write a program that—for a choice of dimension n—takes 100 random initial iterates with elements of x_0 between -10 and 10. Moreover, for each x_0, choose a random Q with the elements of its diagonal between 0.5 and 1.9. For each of these 100 trials, evaluate the number of iterations required before the termination criterion is reached. Lastly, run this program starting from $n = 1$ until $n = 100$, evaluating the average number of iterations required before termination. **Turn in a plot showing the average number of iterations vs. problem dimensionality (size of n).** You should expect that the number of iterations goes up as a function of dimension, but you may not have strong intuition about how it goes up.

3. (20 pts) Given the function $f(x, y) = (x - a)^2 + (x - a)(y - b) + (y - b)^2$ and $x_0 = c$ and $y_0 = d$, what is

$$(c, d)^T - [D^2f(c, d)]^{-1} \cdot [Df(c, d)]$$

   equal to? **Symbolically evaluate the expression above.** This can be done by hand or in python by creating a function to symbolically compute the second derivative of $f(x, y)$, also known as the Hessian. Can you see why I asked you to solve this problem? Re-run your code from Problem 2, replacing the update rule with this one, and note the difference in terms of how many iterations are required. **Turn in a plot for the new update rule showing the average number of iterations vs. problem dimensionality (size of n).**

4. (20 pts) Forward simulate a differential drive vehicle for a length of time $T = 2\pi$ seconds subject to the dynamics,

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta)u_1 \\ \sin(\theta)u_1 \\ u_2 \end{bmatrix}, \quad (x(0), y(0), \theta(0)) = (0, 0, \pi/2).$$

   Use a semi-circle as an initial trajectory. You can get this by simulating the system forward using $[u_1(t), u_2(t)] = [1, -1/2]$. **Turn in: A plot of the semi-circle trajectory of the simulated kinematic car.** Hint: Acceptable methods to solve this problem include Euler integration or an ODE solver like scipy.integrate.solve_ivp.

5. (30 pts) Apply finite-dimensional numerical optimization to the differential drive vehicle in Problem 4. The desired trajectory of the vehicle is $(x_d, y_d, \theta_d) = (\frac{4}{2\pi}t, 0, \pi/2)$. How is the movement of this differential drive vehicle constrained? For one, the starting position of the vehicle is at $x = (0, 0, \pi/2)$. Use an objective function of standard form (from Section 3.2 in the notes) with tuned matrices for Q, R, and P. **Turn in: A plot of the initial trajectory and the optimized trajectory, a plot of the optimized control signal, and the final values for Q, R, and P.** The initial trajectory will be your solution to Problem 4 in the form of a matrix of states and control signals for a discrete number of steps. Hint: If you use scipy.optimize.minimize, the matrix for the initial condition will need to be reshaped into an 1D array (possibly using numpy.reshape) and reshaped again within functions for the objective function and constraints.