COL 726 Homework 5

Due date: Monday, 22 April, 2019

1. (a) Use the composite trapezoidal rule to approximate $I = \int_0^\pi \sin x \, dx$, first with 4 equally spaced nodes at $0, \frac{1}{3}\pi, \frac{2}{3}\pi, \pi$, and then with 7 equally spaced nodes at $0, \frac{1}{6}\pi, \ldots, \frac{5}{6}\pi, \pi$. Finally, use Richardson extrapolation to obtain an even more accurate estimate of I. What is the order of accuracy of the extrapolated result? [2.5 points]

   (b) Suppose instead of the composite trapezoidal rule, you used Newton-Cotes quadrature rules to obtain values Q_4 and Q_7 of the highest possible degree. Ignoring stability issues, is it possible to apply Richardson extrapolation here to obtain even more accuracy than Q_7? Explain. [1 point]

2. Suppose you are given a differentiable function $f : [0, \infty) \to \mathbb{R}$ as a black box, i.e. you are not allowed to evaluate $f(x)$ for any $x < 0$. Derive a finite difference scheme to estimate $f'(0)$ to second-order accuracy. Prove via Taylor series that your scheme is indeed second-order accurate. [3 points]

3. Let $C \subseteq \mathbb{R}^n$ be a convex set.

   (a) Show that the intersection of all halfspaces that contain C is equal to C itself. You may use the separating hyperplane theorem (Boyd and Vandenberghe 2.5.1): If C and D are nonempty disjoint convex sets, i.e. $C \cap D = \emptyset$, then there exist $\mathbf{a} \neq \mathbf{0}$ and b such that $\mathbf{a}^T\mathbf{x} \leq b$ for all $\mathbf{x} \in C$ and $\mathbf{a}^T\mathbf{x} \geq b$ for all $\mathbf{x} \in D$. [1.5 points]

   (b) Let $f : C \to \mathbb{R}$ be the distance to the boundary of C, i.e. $f(\mathbf{x})$ is the largest r such that the ball $B(\mathbf{x}, r) = \{\mathbf{z} : \|\mathbf{z} - \mathbf{x}\| \leq r\}$ lies entirely inside C. Prove that f is concave on C. [1.5 points]

4. In class, we have stated the second-order condition for convexity: *A twice-differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if its Hessian is positive semidefinite everywhere, i.e. $\nabla^2 f(\mathbf{x}) \succeq 0$ for all $\mathbf{x} \in \mathbb{R}^n$*. Prove both directions of this equivalence. You may use the first-order condition for convexity without proving it. [3 points]

   Hint: Try using Taylor's theorem as given in class. Partial marks will be given if you prove the desired result only for the one-dimensional case $f : \mathbb{R} \to \mathbb{R}$.

5. Let $f : \mathbb{R} \to \mathbb{R}$ be a twice-differentiable convex function with a unique minimizer $x^* \in \mathbb{R}$, and bounded second derivative, $f''(x) \leq M$ for all $x \in \mathbb{R}$.

   (a) Prove that $x^{(0)}, x^{(1)}, x^{(2)}, \ldots$ converges to x^* if and only if $f'(x^{(k)})$ converges to zero. [1 point]

   (b) Suppose $f(x^{(0)}) > f(x^{(1)}) > f(x^{(2)}) > \cdots$. Is this sufficient to guarantee that $x^{(k)} \to x^*$ as $k \to \infty$? Justify your answer with a proof or a counterexample. [1 point]

   (c) Modify the analysis of backtracking line search (Boyd and Vandenberghe 9.3.1) to prove that it converges even if f'' is not bounded away from zero, i.e. there is no $m > 0$ such that $f''(x) \geq m$ for all $x \in \mathbb{R}$. [1.5 points]

   Hint: Consider the total decrease in the objective, $f(x^{(0)}) - f(x^{(k)})$. Is this bounded? What does that tell you about $\lim_{k\to\infty} f'(x^{(k)})$?

6. Numerical optimization can be used to find optimal trajectories for robots, drones, virtual characters, etc. Here we will compute the most efficient path that keeps away from a set of obstacles. The provided starter code defines a function f_{obs} which represents the penalty for being close to an obstacle, and its gradient $\mathbf{g}_{\text{obs}} = \nabla f_{\text{obs}}$. You can visualize the obstacle field by calling `visualizeObs()`.

   (a) Implement a function `gradientDescentStep(f, g, x)` that performs one step of gradient descent with backtracking line search. It should take as input an arbitrary objective function f, its gradient $\mathbf{g} = \nabla f$, and the starting point $\mathbf{x}$. Use parameters $\alpha = 0.3$ and $\beta = 0.5$.

   Test your function on f_{obs} with $\mathbf{x}^{(0)} = (0, 0)$, and check that repeated iterations of gradient descent converge to the local minimum at $\approx (-1.26, 0.43)$. Run 20 iterations and plot the points $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots$ on top of the obstacle field; include this plot in your report. [1 point]

   (b) A trajectory from a point $\mathbf{p}_0 = (-1.5, 0)$ to another point $\mathbf{p}_{12} = (1.5, 0)$ can be described via a sequence of intermediate points $\mathbf{p}_1 = (-1.25, y_1), \mathbf{p}_2 = (-1, y_2), \ldots, \mathbf{p}_{11} = (1.25, y_{11})$.[1] Let us define the cost of this trajectory as the sum of a velocity term and an obstacle term,

$$f(\mathbf{y}) = \frac{1}{2} k_{\text{vel}} \sum_{i=1}^{12} (y_i - y_{i-1})^2 + k_{\text{obs}} \sum_{i=1}^{11} f_{\text{obs}}(x_i, y_i),$$

   where $\mathbf{y} = (y_1, y_2, \ldots, y_{11})$ contains all the unknowns, and the parameter values are $k_{\text{vel}} = 10$, $k_{\text{obs}} = 0.1$. Prove that the ith component of ∇f is

$$g_i(\mathbf{y}) = k_{\text{vel}}(-y_{i-1} + 2y_i - y_{i+1}) + k_{\text{obs}} \big(g_{\text{obs}}(x_i, y_i)\big)_2.$$

   Implement functions f and $\mathbf{g}$ in your code, so that you can apply your gradient descent routine. Plot the trajectory found after 20 iterations on top of the obstacle field. Separately, plot the norm of the gradient $\|\mathbf{g}(\mathbf{y}^{(k)})\|$ as a function of k, with a log scale on the y-axis. [1.5 points]

   (c) If f contained only the k_{vel} term, its Hessian would simply be

$$\tilde{\mathbf{H}} = k_{\text{vel}} \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{bmatrix}.$$

   Implement a function `Htilde` that returns this.

   Then, implement a function `steepestDescentStep(f, g, P, x)` that performs one step of steepest descent under the norm $\|\mathbf{x}\|_{\mathbf{P}} = \sqrt{\mathbf{x}^T \mathbf{P} \mathbf{x}}$. Run 20 iterations of steepest descent for f with $\mathbf{P} = \tilde{\mathbf{H}}$, and make the same plots as in part (b). Which algorithm converges faster? [1.5 points]

Please place all your functions `gradientDescentStep`, `f`, `g`, `Htilde`, and `steepestDescentStep` in a single file `hw5.py`.

[1] To simplify the problem we have made the (unrealistic) assumption that the x-coordinates are equally spaced.