EN.553.761: Nonlinear Optimization I

Homework Assignment #2

Starred exercises require the use of MATLAB.

Exercise 2.1: Let $f_i : \mathbb{R}^n \to \mathbb{R}$ be a *convex* function and $0 \leq \alpha_i \in \mathbb{R}$ for $i = 1, \ldots, k$.

(a) Prove that

$$f(x) = \sum_{i=1}^{k} \alpha_i f_i(x)$$

is a convex function.

(b) Prove that

$$f(x) = \max\big(f_1(x), f_2(x), \ldots, f_k(x)\big)$$

is a convex function.

(c) Let $T(x) = Ax + b$ be any affine function from $\mathbb{R}^n \to \mathbb{R}^m$ and let $g : \mathbb{R}^m \to \mathbb{R}$ be a convex function. Prove that $f(x) = g(T(x))$ is a convex function.

(d) Prove that the function $f : \mathbb{R}^n_> \to \mathbb{R}$ given by $f(x) = \sum_{i=1}^n x_i \log(x_i)$ is convex, where $\mathbb{R}^n_>$ denotes the set of vectors with strictly positive ccoordinates, so that $\log(x_i)$ is well-defined.

Exercise 2.2: [Constant step size strategies] In this exercise, we will analyze a constant step size strategy, as opposed to the Armijo backtracking strategy from class, for line-search methods. Consider a function $f : \mathbb{R}^n \to \mathbb{R}$ that is continuously differentiable, and the gradient map $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$ is Lipschitz continuous with Lipschitz constant $\gamma > 0$. Further, we assume that the function is bounded from below, i.e., there exists $\ell \in \mathbb{R}$ such that $f(x) \geq \ell$ for all $x \in \mathbb{R}^n$.

Suppose that we run a Newton-type (modified/quasi) method for generating the search direction p_k at iteration $k = 0, 1, 2, \ldots$. More precisely, at every iteration k, we generate a positive definite matrix B_k and the step direction is chosen as the minimizer of the quadratic approximation at this iteration (we use the same notation from class: $f_k = f(x_k), g_k = \nabla f(x_k)$)

$$m_k^Q(p) = f_k + g_k^T p + p^T B_k p.$$

Recall that this means the step chosen is $p_k = -B_k^{-1} g_k$. We assume that there are global bounds $\lambda_{\min}$ and $\lambda_{\max}$ on the minimum and maximum eigenvalues of all the B_k, $k = 0, 1, 2 \ldots$.

Let the step size α_k be some constant $\alpha > 0$ to be chosen appropriately (see part (ii) below). Let x_0 be an arbitrary starting point, and the iterates $\{x_k\}_{k \geq 0}$ be generated as in a standard line-search scheme:

$$x_{k+1} = x_k + \alpha p_k.$$

(i) Show that for all $k \geq 0$, the following relation holds:

$$f(x_{k+1}) \leq f(x_k) + \alpha g_k^T p_k + \frac{\gamma}{2} \alpha^2 \|p_k\|^2.$$

Deduce that

$$f(x_{k+1}) \leq f(x_k) - \alpha \frac{\|g_k\|^2}{\lambda_{\max}} + \frac{\gamma}{2\lambda_{\min}^2} \alpha^2 \|g_k\|^2.$$

(ii) Consider the quadratic expression $-\alpha\frac{\|g_k\|^2}{\lambda_{\max}} + \frac{\gamma}{2\lambda_{\min}^2}\alpha^2\|g_k\|^2$ in α from the above inequality. For what value of α is it minimized? Is the minimum value positive or negative? Is the minimizing α positive or negative?

(iii) Use the step size α to be the minimizing value from part (ii) above. Now show that there exists a constant M such that for all $T \geq 1$

$$\min_{k=0,\ldots,T} \|\nabla f(x_k)\| \leq \frac{M}{\sqrt{T+1}}.$$

Consequently, for any $\epsilon > 0$, within $\lceil \left(\frac{M}{\epsilon}\right)^2 \rceil$ steps, we will see an iterate where the gradient has norm at most ϵ. In other words, we reach an "ϵ-stationary" point in $O((\frac{1}{\epsilon})^2)$ steps.

(iv) What is the specific value of step size α and the constant M in part (iii) (in terms of the parameters of the problem) when B_k is taken to be the identity matrix at every iteration? Recall that this gives the steepest descent direction at every iteration for the search direction.

Exercise 2.3***:** Write a Matlab m-function that computes a modified Newton matrix based on Algorithm 2 in the course lectures. The function call should have the form

```
[ B, flag ] = modNewton( H, beta )
```

where the input `H` is required to be a symmetric matrix and `beta` > 1 is an upper bound on the required condition number of the modified matrix. On exit, the (possibly) modified positive-definite matrix is stored in `B`, and `flag` should contain the value 0 if no modification was required and the value 1 otherwise.

Exercise 2.4***:** Consider the problem

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \;\; f(x)$$

where f is a twice continuously differentiable function.

(a) Write a Matlab m-function that minimizes a twice continuously differentiable function f using a backtracking-Armijo linesearch. The function call should have the form

```
[x,F,G,H,iter,status] = uncMIN(fun,x0,step,maxit,printlevel,tol)
```

where `fun` is of type *string* and represents the name of a Matlab m-function that computes $f(x)$, $\nabla f(x)$, and $\nabla^2 f(x)$ for some desired function f; it should be of the form

```
[F,G,H] = fun(x)
```

where for a given value x it returns the values of the function, gradient, and Hessian, respectively. The parameter `x0` is an initial guess at a minimizer of f, `step` indicates how the search direction should be computed, `maxit` is the maximum number of iterations allowed, `printlevel` determines the amount of printout required, and `tol` is the final stopping tolerance. If `step` has the value 0, then a steepest-descent search direction should be used; otherwise, a modified-Newton search direction should be computed using your m-file from Exercise 2.3. In the code, if the parameter `printlevel` has the value zero, then no printing should occur; otherwise, a single line of output is printed (in column format) per iteration. On output, the parameters `x`, `F`, `G`, and `H` should contain the final iterate, function value, gradient vector, and Hessian matrix computed by the algorithm. The parameter `iter` should contain the total number of iterations performed. Finally, `status` should have the value 0 if the final stopping tolerance was obtained and the value 1 otherwise.

(b) Write a separate MATLAB m-file with function declaration `[F,G,H] = fun(x)` that returns the value F, gradient G, and Hessian H at the point $x \in \mathbb{R}^2$ of the function

$$f(x) = 10(x_2 - x_1^2)^2 + (x_1 - 1)^2.$$

Use your m-function `uncMIN.m` from part (a) to minimize f with input `step` $= 0$ and then a second time with `step` $= 1$. In both cases, start with $x_0 = (0, 0)$. Comment on your results.