Stanford University, Management Science and Engineering (and ICME)

CME 338 Large-Scale Numerical Optimization

Instructor: Michael Saunders Spring 2019

Homework 4, Due Monday May 20

`http://stanford.edu/class/cme338/homework.html`

1. Consider the LO problem

$$\begin{array}{rll} \min & c^T x & \\ \text{s.t.} & Ax = b, & (1) \\ & x \geq \ell, & (2) \end{array}$$

where A is $m \times n$ $(m < n)$. The primal simplex method is an active-set method that moves from vertex to vertex within the feasible region. A vertex is defined by a set of n independent equations drawn from the constraints (1) and (2). In terms of the usual basis partition

$$AP = \begin{pmatrix} B & N \end{pmatrix}, \qquad x = P \begin{pmatrix} x_B \\ x_N \end{pmatrix}$$

(where P is a column permutation), write a single matrix equation that defines the current basic and nonbasic variables x_B, x_N as a vertex. You may assume that ℓ is finite and the nonbasic variables are currently on their lower bounds. The matrix equation should involve B and N and a few other items.

2. Suppose $P = I$ and the above basic solution is optimal, with dual variables (y, z) satisfying $B^T y = c_B$ and $z = c - A^T y$. Also suppose the last nonbasic variable has bound $\ell_n = 1$ and reduced gradient $z_n = c_n - a_n^T y = 1.0$. Now suppose ℓ_n is changed from 1 to -1. Is the previous solution still optimal? (Yes/No/Maybe. Why? What does z_n tell us?)

3. Primal simplex proceeds by *moving a nonbasic variable away from its current value* while remaining feasible. For the previous example, show how primal simplex can be restarted at the previous optimal solution (with $x_n = 1$). What will (or might) happen during the first iteration?

4. If the simplex implementation thought that nonbasic variables had to be on a bound, it would set $x_n = -1$ before restarting. What does the first basic solution x_B then look like? (Which linear system defines x_B? Is the solution sure to be feasible? Why was it a good idea to start with $x_n = 1$?)

5. Suppose the vector x satisfies $\ell \leq x \leq u$ and p is a search direction. Write an efficient (vectorized) Matlab function to solve the 1D optimization problem

$$\max\ \alpha \quad \text{s.t.} \quad \ell \leq x + \alpha p \leq u, \tag{3}$$

returning α and the index r that keeps $\alpha < \infty$ (else $r = 0$). Allow for some elements of ℓ and u being infinite, and some elements of p being zero. Avoid creating `inf`s and `nan`s.