CO 602: Fundamentals of Optimization
Fall 2010
Problem Set 3
S. Vavasis

Handed out: 2010-Oct-4.

Due: 2010-Oct-13 in lecture.

1. Establish the following claim. Suppose the simplex method encounters a BFS $\mathbf{x}$ such that all the reduced costs are positive (not merely nonnegative). Then $\mathbf{x}$ is unique optimizer for the LP.

2. Suppose one is given an invertible matrix $C \in \mathbf{R}^{m\times m}$ and a second matrix $D \in \mathbf{R}^{m\times n}$, where $n \geq m$. Suppose also one is given C^{-1}, $C^{-1}D$, and two vectors $\mathbf{u}, \mathbf{v} \in \mathbf{R}^m$. Come up with an $O(mn)$ algorithm to evaluate $(C + \mathbf{u}\mathbf{v}^T)^{-1}D$.

3. Suppose we are given an $m \times n$ matrix A of rank m. Consider the following algorithm, written in pseudo-Matlab notation, for finding a subset W of columns A that form a basis of $\mathbf{R}^m$.

   (1) $W = \emptyset$;
   (2) for $i = 1 : m$
   (3) find $p \in \bar{W}$ that maximizes $|A(i, p)|$;
   (4) $W = W \cup \{p\}$;
   (5) $\mathbf{v}^T = A(i, :)/A(i, p)$;
   (6) $A(:, \bar{W}) = A(:, \bar{W}) - A(:, p)\mathbf{v}^T(\bar{W})$;
   (7) end

   This program uses the notation $\bar{W}$ to stand for $\{1, \ldots, n\} - W$. Statement (6) is called a "rank-one update"; the notation $A(:, p)\mathbf{v}^T(\bar{W})$ is an "outer product" of two vectors that yields a matrix. In particular, it is a matrix whose every column is a multiple of $A(:, p)$. Assume that $A(i, p)$ chosen in step (3) is always nonzero. (This can proved; it is a consequence of the assumption that A has rank m.)

   Use the notation A_i to denote the value of A after the ith loop iteration, and similarly use the notation W_i for the ith value of W. Show that $A_i(1 : i, \bar{W}_i)$ is identically zero.

   [Hint: use statements (5)–(6) to argue that that $A_i(i, \bar{W}_i) = 0$, and use induction to argue that $A_i(1 : i - 1, \bar{W}_i) = 0$.]

4. Write a Matlab subroutine `findbasis` that carries out the operation described in Question 3. It should test if $|A(i, p)|$ is zero in step (3), and if so, report that A is not full rank. The program should check that the entries claimed by Q2 to be zero are really zero (or at least are zero within roundoff error).

Test your program on three random 5×8 matrices `A1`, `A2` and `A3` available from the course webpage. Matrices `A1` and `A2` were constructed using `randn(5,8)`. On the other hand, `A3` was constructed to be random and rank deficient; in particular, it was constructed by `A3=randn(5,4)*randn(4,8)`. Does your program detect rank deficiency?

Hand in listings of your programs, the sample runs, and a brief discussion concerning whether the routine detects rank deficiency.

The header for the routine:

```
function wmask = findbasis(A)
% w = findbasis(A)
% This routine takes as input an m-by-n matrix A and looks for a subset of
% m columns that form a basis for R^m.  The routine returns wmask, a 1-by-n
% logical vector, with 1's in the m positions of the basis and 0's
% elsewhere.  If it detects that rank(A) < m, then it generates an error
% message.
```

Here is an example of how the program works on `A1`:

```
>> findbasis(A1)
ans =
     1     1     1     1     0     0     0     1
```

Some additional guidelines are as follows. The program needs to use logical variables and logical subscripting; type `help logical` for more information on this topic. To find the maximum as in step (3), you can use the form of the `max` function that yields two return variables. However, if you apply this function to a subvector of the current row, then the position variable that is returned is with respect to the subvector rather than the original numbering $1 : n$ and so the position with respect to $1 : n$ needs to be recomputed. To generate an error message, use the `error` function. When your program checks that the entries that are supposed to be zero are really zero, it may turn out that because of roundoff error, they are not exactly zero. Thus, to carry out this test, check whether the entries that are supposed to be small have shrunk by a factor of at least 10^{-15}. In other words, `abs((a/b)*b-a)` may not be exactly 0 due to roundoff but it should be a factor of 10^{-15} or more smaller than `abs(a)`.