AMSC 607 / CMSC 764 Fall 2008
HMWK 3: Due October 28, 2pm

Show all work. If you use a reference book, cite it, or you will lose credit!
You may work in groups of up to 4 people. If you do, include a statement, signed by all members of the group, stating the work done by each person. I believe that you can almost finish the homework during classtime on Thursday, October 16.
There may be another assignment before this one is due.

Write a Matlab program using a feasible direction method to solve linear programming problems

$$\max_{\boldsymbol{x}} \boldsymbol{b}^T\boldsymbol{x}$$

$$\boldsymbol{A}^T\boldsymbol{x} \geq \boldsymbol{c}$$

where $\boldsymbol{x} \in \mathcal{R}^n$ and $\boldsymbol{c} \in \mathcal{R}^m$ with $n \leq m$. Assume a constraint qualification.

Write a Matlab function `xopt = lpfeasdir(A,b,c,x)`. The parameters to your feasible direction algorithm are $\boldsymbol{A}$, $\boldsymbol{b}$, $\boldsymbol{c}$, an initial feasible point $\boldsymbol{x}$.

- Use `qrupdate, qrinsert, qrdelete` (instead of the $\boldsymbol{B}$ and $\boldsymbol{N}$ method in the notes) to update a factorization of the matrix $\hat{\boldsymbol{A}}$ corresponding to the currently active constraints.
- At each iteration, $\hat{\boldsymbol{A}}$ gains one row, and it may also lose one: if there is no feasible downhill direction, remove the constraint corresponding to the most negative (estimated) Lagrange multiplier.
- The next point is $\boldsymbol{x} + \alpha\boldsymbol{p}$, where $\boldsymbol{p}$ is determined from solving the system involving a column of the identity matrix, and α defines the longest step that is possible without violating any of the constraints. The constraint that we hit becomes the added one.
- Stop when there is no feasible downhill direction.
- You must apply the feasible direction approach to the problem as written above, not to the dual problem.

Find one linear programming problem on which to test your algorithm.

Grading: 30 points total.

- 20 points for the efficient implementation of the algorithm as a bug-free Matlab function, with good documentation for the calling sequence and the algorithm. "Efficient" means not using an order of magnitude more computation than necessary.
- 10 points for the script that tests the algorithm.

Note. Let `A` and `B` be matrices, and let `c` be a vector. Make sure you understand why the statements `A*(B*c)` and `A \ (B*c)` take much less time than `A*B*c` and `A \ B * c`, and then use this knowledge in your programming.