8. (20) Write a Matlab program that uses a feasible direction method to solve the linear programming problem

$$\min \limits_{x} \ c^T x$$

$$Ax = b,$$

$$x \geq 0,$$

where \( x, c \in \mathbb{R}^n \) and \( b \in \mathbb{R}^m \) with \( m < n \). Assume a constraint qualification. Also assume that the initial point is a vertex (i.e., exactly \( n \) active constraints), and step from vertex to vertex, as in the simplex method.

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

- Use \( \text{qrupdate}, \text{qrinter}, \) and/or \( \text{qrdelete} \) (instead of the \( B \) and \( N \) method in the notes) to update a factorization of the matrix \( A_W \) corresponding to the currently active constraints.
- At each iteration, one row of \( A_W \) is replaced by another.
- The next point is \( x + \alpha p \), where \( p \) is determined from solving the system involving a column of the identity matrix, and \( \alpha \) 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.

Test your program on this linear programming problem (Griva, Nash, and Sofer, p.221):

\[
A = \begin{bmatrix}
2 & 4 & 2 & 1 & 0 & 0 & 0 & 0 \\
3 & 5 & 4 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1
\end{bmatrix};
\]

\[
b = [50; 80; 20; 20; 20];
\]

\[
c = [-5; 10; 15; 3; 0; 2; 0; 0];
\]

\% x0 is the initial point.

\[
x0 = [\text{zeros}(3,1); b];
\]

You can check your answer using Matlab’s simplex algorithm for linear programming:
options = optimset('largescale','off');
[x,fval,exitflag,output,lambda] = ... 
    linprog(c,[],[],A,b,zeros(8,1),inf*ones(8,1),x0,options)

Grading: 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.

Notes

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

• Make sure that each iteration of your algorithm uses only $O(mn + n^2)$ multiplications. This is possible if you compute a QR factorization once, at the beginning of the algorithm, and then, for each iteration, use the updating functions rather than refactoring, or computing an inverse, or solving a linear system involving a general matrix.

• The grader will test your program on a larger problem. See the sample problem generator on the homepage.

• Your program does not need to handle error conditions such as violation of the constraint qualification, infeasibility, etc. This is just an exercise to understand the algorithm. Use a trusted routine such as linprog.m if you ever really need to solve such a problem.