Revised simplex algorithm

Motivation:

The main motivation of the revised simplex algorithm is to make the simplex algorithm run faster and more efficiently. The main problem in the simplex algorithm is the fact that every iteration we have to calculate the whole simplex tableau which is an extremely inefficient process because the tableau contains lots of zeros and many of the columns are not changed in every iteration. The revised simplex algorithm exploits the fact that only little columns are changed in every iteration.

Theory:

The simplex algorithm is inefficient because it calculates the whole simplex tableau in every iteration. However, by observing the iterations we realize that only few columns of the constraint matrix change. Linear programing problems typically have much more variables than the constraint which makes the constraint matrix sparse with many zeros that doesn't change or affect other columns during the pivoting step. The revised algorithm tries to calculate only the columns that change and at the same time store the calculations done in previous steps in a basis matrix and in the constant of constraints vector. Then, the algorithm uses the basis matrix to calculate the cost function at every iteration which is then used to determine the entering variable. The original column of the entering variable is then updated by the basis matrix and used with the constant vector to determine the leaving variable. Finally, pivoting is done to update the basis matrix and the constant vector. By doing so, the calculations are reduced greatly and the algorithm runs much faster especially in problems with a larger number of variables compared to constraints.

Algorithm:

1) Write the given system of equations in canonical form and adding the artificial variables to all the constraints to guarantee the detection of a feasible point in order to start the second phase.
2) The first iteration starts with the artificial variables as the basic variables and thus the basis matrix is initialized as an identity matrix.
3) Calculate the new cost factors d (phase 1 coefficients {w}) and c (objective function {f}) by
$$d_j = d_{jorginal} - \sigma^T * A_j$$
$$c_j = c_{joriginal} - \pi^T * A_j$$
Where σ, π $corresponds\ to\ w\ and\ f\ coefficients\ respecitively$
$A\ is\ the\ orginal\ constraint\ matrix$
4) If the current cycle is phase 1 and if all $d_j \geq 0$, there are two possibilities. The first if $w_0 > 0$ then the problem has no feasible solution. The second is $w_0 = 0$ then the current point is a feasible point and thus phase 2 can start by dropping the w row and all the columns of the artificial variables.
On the other hand, if some $d_j < 0$ then, we choose x_j corresponding to the most negative coefficient as the entering variable.
Finally, if the current cycle is a phase 2 cycle, then if all $c_j \geq 0$ then we have reached our optimal point. Otherwise, if some $c_j < 0$ we choose x_j corresponding to the most negative coefficient as the entering variable.
5) Compute the elements of the new pivot column x_s as
$$A_s = B * A_{soriginal}$$
6) Determine the leaving variable by finding the least entry in the new column calculated as $\frac{b}{A_s}$ where $b\ is\ the\ constraint\ constant\ vector.$
7) Finally, perform a pivot operation on the A_s column using the entry corresponding to the leaving variable.
8) Repeat from step 3

Implementation:

The implementation is done in MATLAB and the codes are attached with the report.

Application:

The revised simplex algorithm is used to solve linear programing problems which models many applications specially in operations research. It is also used in the early formation of microeconomics and is also used in company management where many problems such as planning, production and transportation can be modeled as linear programing problems. It is also used in maximizing profits and minimizing costs in many real-life problems.

Comparison with linprog:

The linprog in MATLAB use dual simplex algorithm which uses the dual problem of the linear problem to efficiently solve the primal problem.

To compare we use the following problem

$$Max\ F = x_1 + 2x_2 + x_3$$

Subject to:

$$2x_1 + x_2 - x_3 \leq 2$$

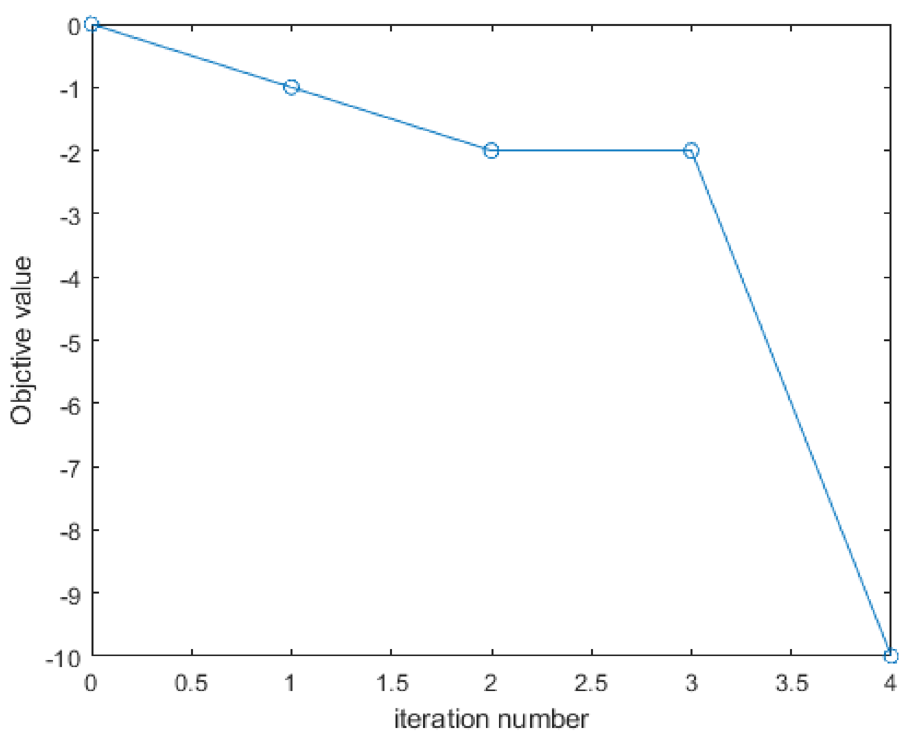
$$2x_1 - x_2 + 5x_3 \leq 6$$

$$4x_1 + x_2 + x_3 \leq 6$$

$$x_1, x_2, x_3 \geq 0$$

Both algorithms give the optimum point at (0,4,2) with $F_{max} = 10$

The linprog finds it in only two iterations while my implementation finds it in 4 iterations with the following $-f$ values in each iteration



However, the first two iterations are done in phase 1 to find a feasible point. So, I tried the same implementation but without phase and the algorithm found it in only 2 iterations