A hybrid gradient and feasible direction pivotal solution algorithm for general linear programs

Hossein Arsham

University of Baltimore, 1420 N Charles Street, Baltimore, MD 21201, USA

Abstract

The simplex solution algorithm for linear programs (LP) can be considered as a sub-gradient direction method therefore a full gradient solution algorithm might be more efficient. We have developed a full gradient method which consists of three phases. The initialization phase provides the initial tableau which may not have a full set of basis. The push phase uses a full gradient vector of the objective function to obtain a feasible vertex. This is then followed by a series of pivotal steps using the sub-gradient, which leads to an optimal solution (if exists) in the final iteration phase. At each of these iterations, the sub-gradient provides the desired direction of motion within the feasible region. The algorithm hits and/or moves on the constraint hyper-planes and their intersections to reach an optimal vertex (if exists). The algorithm works in the original decision variables space, therefore, there is no need to introduce any new extra variables such as artificial variables. The proposed algorithm is practical, easy for the user to understand, and provides useful information for the decision makers.

© 2006 Elsevier Inc. All rights reserved.

Keywords: Linear programming; Full gradient Simplex algorithm; Artificial-free; Pivoting algorithm; Feasible direction method; Advance basis

1. Introduction

The simplex method is a well-known and widely used method for solving LP problems. The number of iterations needed to solve an LP by the simplex method depends mainly upon the pivot columns used, and can be exponential for certain LP problems. Quandt and Kuhn [1] showed experimentally that the steepest unit ascent method will generally require a perceptibly higher number of iterations than, for example, the use of gradient criteria. There has been growing interest in developing alternative polynomial bounded algorithms for the LP problems. Khachian [2] makes one such attempt. However, the Khachian algorithm is unsatisfactory for practical problems (of even small size) and its average behavior is inferior to the modern simplex based LP-codes. Recent work, the polynomial projection algorithm of Karmarkar [3], sparked enormous interest in the operations research community.

E-mail address: harsham@ubalt.edu
Terlaky and Zhang [4] discuss the various pivot rules of the simplex method and its variants that have been
developed in the last two decades. All these methods can be considered as methods of sub-gradient direction.
The authors conclude that the hardest and long standing open problems in the theory of LP are still concerned
with pivot methods.

In this paper we develop a gradient pivotal method for solving LP problems. The algorithm is applicable to
general LP problems in the sense that, unlike the standard simplex method, some or all decision variables can
be unrestricted in sign and RHS $P_0$ is not required (therefore there is no need to introduce artificial vari-
ables). The algorithm works in the space of the original variables. The computational procedures developed
here have been embedded within otherwise simplex-like tableaux. The algorithm consists of three phases. The
initialization phase provides the initial tableau, which may not have a complete basic variable set. The push
phase provides hits with some constraint hyper-planes using the full gradient direction of the objective func-
tion (if needed) to reach a feasible vertex. At the beginning of a typical final iteration phase of the procedure
there is a current basic feasible solution in a tableau. The search continues using an updated sub-gradient until
a hit with a face of a constraint hyper-plane is obtained. This is then followed by a series of pivotal steps which
leads to an optimal solution, if one exists. At each step, the updated sub-gradient provides the desired direc-
tion of motion, all computational basis being the Gauss–Jordan pivoting (GJP) used in the simplex method.
The simplex method moves along the edges of the feasible region and does not provide for solutions other than
the vertices of the constraint set. This procedure may be viewed as an extension of the simplex method which
makes such a provision.

We present the gradient-based procedure in Section 2 and give a proof of finiteness. Section 3 deals with
the computational aspects of the algorithm with examples to illustrate all aspects of the algorithm.
Section 4 handles treatment of special cases and Section 5 presents concluding remarks and future research
directions.

2. The proposed solution algorithm

Consider the following general linear programming problem:

Problem GLP:

$$\begin{align*}
\text{Maximize} & \quad C^T X \\
\text{subject to} & \quad A_1 X \leq b_1, \\
& \quad A_2 X \geq b_2,
\end{align*}$$

where $C \in R^n$, $[A_1, A_2]^T \in R^{m \times n}$, $(b_1, b_2)^T \in R^m$, and $b_1 > 0, b_2 \geq 0$. We assume that $[A_1, A_2]^T$ has a full row
rank. Any restricted variable constraints are included in this set of constraints. Note that if there is an unre-
stricted variable in equality Constraint, one may use the constraint equation to eliminate the variable. This
reduces the size of problem in both number of variables and number of constraints. Otherwise, convert each
equality constraint into two inequality constraints. We assume that there are $m$ constraints after all such
transformations.

2.1. Intuitive description of the algorithm

The basic idea is to start at the origin and utilize the full gradient of the objective function to get the first hit
into the feasible region. We explain the process using the following problem:

$$\begin{align*}
\text{Maximize} & \quad z = 3X_1 + 5X_2 \\
\text{subject to} & \quad 2X_1 + X_2 \leq 40, \\
& \quad X_1 + 2X_2 \leq 50, \\
& \quad X_1 \geq 0, X_2 \geq 0.
\end{align*}$$

In practical terms, we may suppose that the given problem represents profit maximization, subject to two
capacity restrictions (e.g. machine-hours on different machines).
If the inequality sign of a constraint is replaced by an equality sign, a straight line (hyper-plane) is obtained which forms the boundary of the half space. If the problem is to be solved geometrically, the straight line \( z = 3X_1 + 5X_2 \) for an arbitrary value \( z \) is perpendicular to the gradient vector. The optimal \( z \) belongs to an iso-objective function line value which passes through a feasible vertex with the largest distance from the origin that is 130.

If we move in the direction of the full gradient in search of optimal vertex, this search strikes a boundary hyper-plane to get the first hit, which may or may not be feasible. If a constraint hyper-plane is hit, the algorithm would not hit it again. Its path is then restricted to the boundary hyper-planes and their intersection and it continues on in such a manner that at each point it is moving in the direction of the updated full gradient of the objective function; namely that direction which takes it to feasibility and then to the optimal increase of the objective value. Once feasible, we reduce the dimensionality of the gradient space at each step and get closer to the optimal solution. The unique path defined by the starting point consists of a finite number of directional line segments (i.e., vectors) that provides the coordinates of the optimal vertex, see Gal [5] for a misleading graph.

### 2.2. Development of gradient-based algorithm

Following are some additional notations used in the new gradient pivotal-based algorithm, hereafter referred to as algorithm GP:

- **BVS** basic variable set
- **PR** pivot row
- **PC** pivot column
- **PE** pivot element
- **S_i** the \( i \)th slack/surplus variable, \( 1 \leq i \leq m \)
- **R_i** the \( i \)th row of a tableau, \( 1 \leq i \leq m \)
- **OR** open row, not yet assigned in the BVS
- ? label for an open row
- **C_k** vector of the last row of the \( k \)th tableau
- **Z_k** vector \( \{ Z_{kj}; Z_{kj} = C_{kj} \text{ for } 1 \leq j \leq n, Z_{kj} = 0 \text{ for } j > n \} \)
- **C_k^+** vector \( \{ C_{kj}^+; C_{kj}^+ = C_{kj} \text{ if } C_{kj} \geq 0, C_{kj}^+ = 0 \text{ otherwise} \} \)
- \( N(C_k^+) \) the number of positive entries in vector \( C_k^+ \)
- **RHS** right hand side
- **Y_k** the \( k \)th inner product of two vectors
- **L** column ratios, RHS/\( Y_k \)

The algorithm consists of an initialization phase, a push phase, and a final iteration phase of sequential pivoting, to reach an optimal solution.

#### 2.2.1. Algorithm GP

##### 2.2.1.1. Initialization phase

Step I1: Convert the LP to the standard form GLP.
Step I2: Introduce slack/surplus variables for each constraint.
Step I3: Initialize \( k = 1 \).
Step I4: Set up the initial simplex tableau by keeping rows for all \( \geq \) constraints, except for the non-negativity constraints, open.

To start the algorithm the LP should be converted to the standard form of problem GLP with all \( \leq \) inequalities with positive RHS. Any constraint with 0 in the RHS, except for non-negativity constraints, should appear as a \( \geq \) inequality only. The non-negativity constraints should be converted to \( \leq \) inequalities to add the slack variables. The rows for these constraints are not open. This phase sets the origin as the starting point.
Note that by this construction all slack variables would appear in the initial BVS. If all constraints are \( \leq \) inequalities, there will be no open row. This implies that the push phase is not needed. See Arsham [6] for a discussion of open rows.

2.2.1.2. Push phase

Step P1: Check if a ? label exists in the BVS. If no, go to step F1.
Step P2: Create column \( Y_k \) as inner product of each row with \( Z_k \), \( Y_{ki} = Z_k \cdot R_i \).
Step P3: Save non-zero \( Z_{kj} \)'s as \( C_k(X_j) = Z_{kj} \).
Step P4: Create vector \( L \) of column ratios.
Step P5: If \( L_i < 0 \) for all ORs, go to Step P7.
Step P6: Determine step length as \( L_i = \min \{ L_i: R_i \text{ is OR}, L_i \geq 0 \} \). Set PE = \( Y_{ki} \). Go to Step P8.
Step P7: Determine step length as \( L_i = \min \{ |L_i|: R_i \text{ is OR} \} \). Set PE = \( Y_{ki} \).
Step P8: Perform GJP on the \( k \)th tableau. Adjust the BVS.
Step P9: Set \( k \) to \( k + 1 \). Go to step P1.

The purpose of this phase is to complete the BVS of the initialization phase while moving in the direction of the full gradient to get some hits on a boundary hyper-plane. The variables are brought into open rows marked by (?) only, and there is no replacement of variables in the BVS. Thus, in this phase, we push toward a basic feasible solution.

Let the starting point, the origin, be denoted by \( X^0 \), which may be an incomplete basic solution already available in tabular form. Starting at \( X^0 \), the algorithm moves in the direction of the full-gradient whose elements are the elements of \( C_1 \). Find the first hyper-plane which is a hit with the smallest non-negative step size. If all eligible entries are negative, we take the smallest absolute step size. Let the hit point be \( X^1 \). Pivot on \( X^1 \). Continue till the BVS is complete, indicating a hit \( X^k \) with a feasible hyper-plane. If by the end of this phase, \( C_k \geq 0 \), this is an optimum solution.

2.2.1.3. Final iteration phase

Step F1: If \( C_{kj} \leq 0 \) for all \( j \), go to Step F10.
Step F2: Create column \( Y_k \) as inner product of each row with \( C_k^+, Y_{ki} = C_k^+ \times R_i \).
Step F3: Save \( C_{kj}^+ \)'s as \( C_k(X_j) = C_{kj}^+ \).
Step F4: Create vector \( L \) of column ratios.
Step F5: Determine the permissible step length as \( L_i = \min \{ L_i, 1 \leq i \leq m, L_i \geq 0 \} \). Set PR = 1.
Step F6: Set PE = \( Y_{ki} \).
Step F7: Perform GJP on the \( k \)th tableau. Set \( k \) to \( k + 1 \). Adjust the BVS.
Step F8: If \( N(C_k^+) < N(C_{k-1}^+) \), go to Step F1.
Step F9: Set \( k \) to \( k - 1 \), delete the \( i \)th row from the \( k \)th tableau. Go to Step F1.
Step F10: STOP. The tableau is optimal. Construct optimal solution.

The purpose of this phase is to reach to an optimal vertex while maintaining feasibility. Now, starting from \( X^k \) we move toward the sub-gradient whose elements are \( C_k^+ \). This pivoting operation gives a hit point \( X^{k+1} \). One of two distinct cases arises:

(i) The order of gradient reduces (i.e. the number of positive entries in \( C_k^+ \) decreases): implies that the hit hyper-plane contains an optimal solution. Use \( X^{k+1} \) as the new starting point. Continue with the new sub-gradient to search for a new hit. The working space of the problem is now in this reduced space. The intersection of this hyper-plane with its adjacent hyper-planes provides the boundaries of the new feasible region.

(ii) The order of gradient remains the same (i.e. the number of positive \( C_k^+ \) remains the same): implies that the hit hyper-plane does not contain an optimal solution. Starting from \( X^{k+1} \) we move along the new sub-gradient to find a new hit. Delete the hyper-plane of the previous hit. This reduces the size of the problem and prevents any zigzagging.
While remaining on a hyper-plane, the algorithm moves from the current sub-space to hit another hyper-plane at the intersection of these two hyper-planes (a space of lower dimensions). This process is repeated to reach an optimal vertex (i.e., zero dimension) provided there is no degeneracy. This phase is similar to the feasible direction method, Zoutendijk [7].

The optimal solution, \( x^*_j \) for \( 1 \leq j \leq n \), are obtained using the final tableau as follows:

\[
x^*_j = x_j \text{ (value in the final tableau)} + \sum I(Y_k) C_k(x_j)
\]

where \( \Sigma \) is over all values of \( k \), and \( I(Y_k) = 1 \) if \( Y_k \) is in BVS, \( I(Y_k) = 0 \) otherwise. The needed information to compute \( x^*_j \) is extracted at the end of each tableau to avoid saving the whole tableau. This saves storage as, like the simplex method, we only need the current tableau to generate the next one. Compute slack/surplus values in the same manner, or use the original constraints.

2.3. Notes on the algorithm GP

1. If all constraints of an LP are inequalities with RHS > 0 and \( C > 0 \), a feasible hit is certain and the push phase is bypassed. Non-negative constraints may be present.
2. The first tableau provides a BVS which may not be complete, but always provides the origin as an initial starting point.
3. In the push phase, to ensure the first hit we use the full gradient direction which has \( n \) components. The strategy is to get on a feasible boundary hyper-plane via gradient direction. In the process, we may get a hit outside of the feasible boundary hyper-plane, i.e., RHS may become negative. At the end of this phase, a basic feasible solution is available. The push phase can be viewed as a gradient pivotal phase.
4. The \( Y_k \) column is always the PC. The smallest step size determines the PR, hence PE is always in the \( Y_k \) column, and only \( Y_k \)’s can come into the BVS, see, Mitra et al. [8], for the concept of inner product.
5. In the final iteration phase we restrict ourselves to only positive entries of \( C_k \), the last row of \( k \)th tableau, and use the vector \( C^+_k \) to determine the sub-gradient direction (which is a feasible direction) at each step of iteration to ensure optimal increase, see Able [9]. This phase can be viewed as a feasible direction method.
6. The final iteration phase maintains feasibility hence all slack/surplus variables should maintain non-negative RHS. Negative slack/surplus is a sign of infeasibility. \( Y_k \)’s can have negative RHS.
7. The number of positive entries in \( C_k \) either remains the same or reduces. This is unlike the simplex method where this number may increase from iteration to the next one.
8. At every iteration, before a new tableau replaces the old one, we save the \( C_k \) entries used in the inner product to create \( Y_k \). These entries are used later to construct optimal values of the variables. This saves the storage requirement as in the simplex method.
9. The algorithm terminates when all elements of a \( C_k \) row are non-positive and all RHS for slacks are \( \geq 0 \). As discussed in Section 4, the algorithm also discovers if the problem is unbounded or infeasible.

2.4. General comments

(i) The simplex method brings one column at a time into the BVS. The algorithm GP brings \( Y_k \), which is constructed using many eligible columns, into the basis at each step. Thus it combines several variables at each step.
(ii) If any unrestricted variable, \( x_j \), appears in an equality constraint, one may use the constraint equation to eliminate \( x_j \) from the problem and reduce the size of the problem in both number of variables and number of constraints. Otherwise, convert each equality constraint to two inequality constraints to force the gradient direction to hit either of corresponding hyper-plane so that these constraints are satisfied at subsequent steps.
(iii) Combining both the deletion and the inclusion of the original hyper-planes of the boundary of the feasible region, together with the fact that the gradient vector, if it hits a hyper-plane, hits it once only, we never return to the same hyper-plane, see Zoutendijk [7]. For this reason there is no zigzagging precaution and,
thus, finiteness follows. The amount of computation per iteration required by the proposed algorithm is 
insignificantly greater than that required by the standard simplex method. The algorithm is also robust 
with respect to gradient data perturbation.

(iv) The algorithm is most suitable for problems with \( m \gg n \). Otherwise one may solve the dual by the pro-
posed algorithm, and then use the complementary slackness theorem to compute the optimal solution of 
the original (primal) problem.

Theorem 1. Problem LP:

Maximize \( C^T X \)

subject to \( AX \leq b, \)

\( X \geq 0, \)

where \( X, C \in \mathbb{R}^n, A \in \mathbb{R}^{mxn}, \) and \( b \in \mathbb{R}^m \). We assume that \( A \) has a full row rank.

Provided the non-degeneracy assumption is not violated, the gradient algorithm finds an optimal solution in at 
most \( 2m + n \) iterations (or shows that it is unbounded or infeasible), where \( m \) is the number of constraints 
excluding the non-negativity constraints.

Proof. Note that if an inequality sign of a constraint is replaced by an equality sign, a hyper-plane is obtained 
which forms the boundary of the half space. There are \( m + n \) hyper-planes. If \( b > 0 \) in the above problem the push phase is completely bypassed. Otherwise, in the worst case, when \( b \leq 0 \) (i.e., all constraints are \( \geq \) inequalities), algorithm GP takes \( m \) steps to complete the BVS in the push phase. The final iteration phase moves in the direction of the updated sub-gradient, moving from one hyper-plane to another in search of 
an optimal solution, never returning to a hyper-plane once visited. An optimal vertex is the intersection of at least \( n \) hyper-planes (for degenerate optimal vertex more than \( n \)). Thus, there are at most \( m \) hyper-planes which do not contain an optimal solution (these are non-binding constraints). The worst case happens when the first \( m \) hits belong to hyper-planes that do not contain an optimal solution (not binding). On the other hand, an optimal solution is a vertex which is the intersection of \( n \) hyper-planes. This requires at most \( n \) iterations to reach that vertex hitting all these hyper-planes. Thus, the number of iterations in the final iteration phase is at most \( m + n \). Algorithm GP, therefore, requires at most \( m + (m + n) = 2m + n \) iterations, \( m \) being the number of constraints excluding the non-negativity constraints. \( \Box \)

Theorem 2. Problem GLP:

Maximize \( C^T X \)

subject to \( A_1 X \leq b_1, \)

\( A_2 X \geq b_2, \)

where \( C \in \mathbb{R}^n, [A_1, A_2]^T \in \mathbb{R}^{mxn}, (b_1, b_2)^T \in \mathbb{R}^m, \) and \( b_1 > 0, b_2 \geq 0 \) \( m \) being the total number of constraints, including the non-negativity conditions, if any. Algorithm GP requires at most \( 2m \) iterations to find an optimal solution of problem, if one exists, provided the problem is not a degenerate one.

Proof. Follows easily from a similar discussion as for Theorem 1. \( \Box \)

Corollary 1. Let \( m_1 \) be the number of \( \geq \) inequalities in problem GLP, excluding the non-negativity constraints. Algorithm GP requires at most \( m + m_1 \) iterations to solve this problem.

2.5. A comment on the maximum number of iterations for the case of non-degenerate problems

Klee and Minty [10] produce LP examples with \( n \) variables and \( 2n \) constraints to demonstrate that the standard simplex requires \( 2^n - 1 \) iterations, i.e. an exponential number. For \( n = 3 \), the problem is
Maximize \(100X_1 + 10X_2 + X_3\)
subject to \(X_1 \leq 1,\)
\(20X_1 + X_2 \leq 100,\)
\(200X_1 + 20X_2 + X_3 \leq 100,000,\)
\(X_i \geq 0 \quad \text{for} \quad i = 1, 2, 3.\)

This problem requires \(2^3 - 1 = 7\) simplex iterations while our algorithm takes at most \(m = 6\) iterations in final iteration phase, to reach the optimal vertex. Clearly, no single algorithm will work efficiently across the broad domain of LP. For large problems, i.e., for large \(n, 2^n - 1 > 2n\) (for \(n > 3\)), and the savings by algorithm GP could be significant.

3. Illustrative numerical examples

Examples to demonstrate various steps and computational aspects of algorithm GP

3.1. LP with all constraints with inequalities

Problem P1:

Max \(2X_1 + 3X_2\)

subject to \(X_1 + 2X_2 \leq 8,\)
\(2X_1 + X_2 \leq 10,\)
\(X_2 \leq 7/2,\)
\(X_1 \leq 4.\)

Since all constraints are inequalities, algorithm GP would take at most \(m = 4\) iterations to solve this LP.

3.1.1. Initialization phase

After performing steps 1 thru 4, we obtain the following initial tableau:

<table>
<thead>
<tr>
<th>BVS</th>
<th>(X_1)</th>
<th>(X_2)</th>
<th>(S_1)</th>
<th>(S_2)</th>
<th>(S_3)</th>
<th>(S_4)</th>
<th>(Y_1)</th>
<th>RHS</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S_1)</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>8</td>
<td>1 ←</td>
</tr>
<tr>
<td>(S_2)</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>10</td>
<td>10/7</td>
</tr>
<tr>
<td>(S_3)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>7/2</td>
<td>7/6</td>
</tr>
<tr>
<td>(S_4)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>(C_1)</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td>13</td>
<td>↑</td>
</tr>
</tbody>
</table>

Note that there is no open (?) row, since there is no constraint with \(\geq\) inequality. Hence the push phase is not needed.

3.1.2. Final iteration phase

Calculate \(Y_1\) and \(L\) (column/row ratios). \(Y_1\) uses \(C_1(X_1) = 2,\) and \(C_1(X_2) = 3.\) Save this information. To avoid redundancy of calculations, we add \(Y_1\) and \(L\) columns to the previous tableau. From now on this will be the standard way of placing these columns. Since \(L_1 = 1\) is the smallest ratio, and the \(C_1\) row has the largest positive number in the \(Y_1\) column, \(PE = Y_{11}\), perform GJP.
We continue to calculate $Y_2$ and $L$ columns in the same tableau after performing GJP. Save $C_3(X_1) = \frac{3}{8}$. Looking at the positive $L$ entries for a possible step length, we find a tie. We randomly pick $S_4$ to exit. After GJP, we obtain the following:

This tableau is optimal, as there is no positive entry in the $C_3$ row. Note that this tableau is not like the usual final simplex tableau, i.e., there is no basis inverse nor a dual solution. By knowing the optimal solution, clearly, these can be computed by inverting the optimal basis matrix and using the complementary slackness theorem respectively. The optimal solution can be found by using the information saved at the end of iterations on $Y_j$ and the final tableau, as follows:

$$X_1 = 0 + 2\left(\frac{2}{3}\right) + \frac{3}{8}(\frac{64}{9}) = 4,$$

$$X_2 = 0 + 3\left(\frac{2}{3}\right) = 2,$$

$$S_1 = 0, \quad S_2 = 0, \quad S_3 = \frac{3}{2}, \quad S_4 = 0.$$

### 3.2. Deletion of a hyper-plane

**Problem P2:**

Max $2X_1 + 2X_2$

subject to $X_1 + X_2 \geq 1$,

$$X_1 + X_2 \leq 2,$$

$$-2X_1 + X_2 \leq 0,$$

$$X_1 \geq 0, \quad X_2$$ unrestricted.

There are 2 inequalities and 4 constraints. Thus, algorithm GP would need 2 iterations in the push phase and at most 4 iterations in the final iteration phase.

### 3.2.1. Initialization phase

Add slack and surplus variables. Note that we do not introduce any artificial variables. Construct the initial tableau by keeping the rows for the first and third constraints open.
3.2.2. Push phase

After obtaining the tableau, we introduce $Y_1$. Save $C_1(X_1) = 2$ and $C_1(X_2) = 2$. In this phase we can allow RHS to become negative because it is possible to hit outside the feasible region in the process of reaching it. We consider column $L$ entries for open rows only. After GJP on $Y_{13}$, we get the next tableau:

<table>
<thead>
<tr>
<th>BVS</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$Y_1$</th>
<th>RHS</th>
<th>$L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-4</td>
<td>-1</td>
<td>1/4</td>
</tr>
<tr>
<td>$S_2$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>?</td>
<td>-2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>-2</td>
<td>0</td>
</tr>
<tr>
<td>$S_4$</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-2</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$C_1$</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

Create $Y_2$. Save $C_2(X_1) = -6$ and $C_2(X_2) = 6$. The next iteration completes the BVS.

<table>
<thead>
<tr>
<th>BVS</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$Y_1$</th>
<th>$Y_2$</th>
<th>RHS</th>
<th>$L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>-3</td>
<td>3</td>
<td>-1</td>
<td>0</td>
<td>-2</td>
<td>0</td>
<td>0</td>
<td>36</td>
<td>1</td>
<td>1/36</td>
</tr>
<tr>
<td>$S_2$</td>
<td>-3</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>-2</td>
<td>0</td>
<td>0</td>
<td>36</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>$Y_1$</td>
<td>1</td>
<td>-1/2</td>
<td>0</td>
<td>0</td>
<td>1/2</td>
<td>0</td>
<td>1</td>
<td>-9</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$S_4$</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-12</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$C_2$</td>
<td>-6</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>-4</td>
<td>0</td>
<td>0</td>
<td>72</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

3.2.3. Final iteration phase

Again, we add columns $Y_3$ and $L$ in the previous tableau. Perform GJP.

<table>
<thead>
<tr>
<th>BVS</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$Y_1$</th>
<th>$Y_2$</th>
<th>$Y_3$</th>
<th>RHS</th>
<th>$L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y_2$</td>
<td>-1/12</td>
<td>1/12</td>
<td>-1/36</td>
<td>0</td>
<td>-1/18</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1/18</td>
<td>1/36</td>
<td>-1/2</td>
</tr>
<tr>
<td>$S_2$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1/2 ←</td>
</tr>
<tr>
<td>$Y_1$</td>
<td>1/4</td>
<td>1/4</td>
<td>-1/4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-1/2</td>
<td>1/4</td>
<td>-1/2</td>
</tr>
<tr>
<td>$S_4$</td>
<td>0</td>
<td>0</td>
<td>-1/3</td>
<td>0</td>
<td>1/3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-2/3</td>
<td>1/3</td>
<td>-1/2</td>
</tr>
<tr>
<td>$C_3$</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2/3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BVS</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$Y_1$</th>
<th>$Y_2$</th>
<th>$Y_3$</th>
<th>RHS</th>
<th>$L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y_3$</td>
<td>0</td>
<td>0</td>
<td>1/2</td>
<td>1/2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>$Y_1$</td>
<td>1/4</td>
<td>1/4</td>
<td>0</td>
<td>1/4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>$S_4$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1/3</td>
<td>1/3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2/3</td>
<td>2/3</td>
</tr>
<tr>
<td>$C_4$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
This is the optimal solution. The solution is
\[ X_1 = 0 + 2(1/2) - 6(1/18) = 2/3, \]
\[ X_2 = 0 + 2(1/2) + 6(1/18) = 4/3. \]

*Note:* There is no reduction in the dimension of the gradient when we move from tableau 2 (with \( C_2 \)) to tableau 3 (with \( C_3 \)). This indicates that the first constraint is never a binding constraint. So, as per step F9, we could go back to the previous tableau deleting the first constraint. Recall that this step reduces the complexity of the LP and eliminates zigzagging among hits. The adjusted tableau is as follows:

<table>
<thead>
<tr>
<th>BVS</th>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( S_2 )</th>
<th>( S_3 )</th>
<th>( S_4 )</th>
<th>( Y_1 )</th>
<th>( Y_2 )</th>
<th>RHS</th>
<th>( L )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_2 )</td>
<td>-3</td>
<td>3</td>
<td>1</td>
<td>-2</td>
<td>0</td>
<td>18</td>
<td>2</td>
<td>1/9</td>
<td>←</td>
</tr>
<tr>
<td>( Y_1 )</td>
<td>1</td>
<td>-1/2</td>
<td>0</td>
<td>1/2</td>
<td>0</td>
<td>1</td>
<td>-3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>( S_4 )</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-6</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>( C_2 )</td>
<td>-6</td>
<td>6</td>
<td>0</td>
<td>-4</td>
<td>0</td>
<td>0</td>
<td>36</td>
<td></td>
<td>↑</td>
</tr>
</tbody>
</table>

3.2.4. Final iteration phase

Note that in this phase, like in simplex, only those 0 values are permissible in \( L \) where RHS = 0 and the corresponding PC value is positive. Calculate \( Y_2 \) and \( L \). Save \( C_2(X_2) = 6. \) Performing GJP on \( Y_2 \) yields:

<table>
<thead>
<tr>
<th>BVS</th>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( S_2 )</th>
<th>( S_3 )</th>
<th>( S_4 )</th>
<th>( Y_1 )</th>
<th>( Y_2 )</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Y_2 )</td>
<td>-1/6</td>
<td>1/6</td>
<td>1/18</td>
<td>-1/9</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1/9</td>
</tr>
<tr>
<td>( Y_1 )</td>
<td>1/2</td>
<td>0</td>
<td>1/6</td>
<td>1/6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1/3</td>
</tr>
<tr>
<td>( S_4 )</td>
<td>0</td>
<td>0</td>
<td>1/3</td>
<td>-1/3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1/3</td>
</tr>
<tr>
<td>( C_3 )</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

This is the optimal tableau with solution:
\[ X_1 = 0 + 2(1/3) = 2/3, \]
\[ X_2 = 0 + 2(1/3) + 6(1/9) = 4/3. \]

3.3. Full gradient with negative direction

Problem P3:

\[
\text{Max} \quad -X_1 - X_2
\]
subject to
\[ 2X_1 + X_2 \geq 4, \]
\[ X_1 + 2X_2 \leq 5. \]

Algorithm GP takes at most \( 2m = 4 \) iterations to solve problem P3.

3.3.1. Initialization phase

<table>
<thead>
<tr>
<th>BVS</th>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( S_1 )</th>
<th>( S_2 )</th>
<th>( Y_1 )</th>
<th>RHS</th>
<th>( L )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( ? )</td>
<td>2</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>-3</td>
<td>4</td>
<td>-4/3 ←</td>
</tr>
<tr>
<td>( ? )</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>-1</td>
<td>-3</td>
<td>5</td>
<td>-5/3</td>
</tr>
<tr>
<td>( C_1 )</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td></td>
<td>↑</td>
</tr>
</tbody>
</table>
3.3.2. Push phase

Now, PC = Y. Save $C_1(X_1) = -1$ and $C_1(X_2) = -1$. Since $L_i < 0$ for all $i$, this indicates that the full gradient direction is away from the feasible region. By determining PE as in Step P6, we change the direction to get a first hit. Also notice that $C_1 < 0$ is not an indication of optimality. In this sense, the initial tableau and the subsequent iterative tableaux differ in characteristics. The following tableaux are obtained:

<table>
<thead>
<tr>
<th>BVS</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$Y_1$</th>
<th>$Y_2$</th>
<th>RHS</th>
<th>$L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y_1$</td>
<td>$-2/3$</td>
<td>$-1/3$</td>
<td>$1/3$</td>
<td>$0$</td>
<td>$1$</td>
<td>$-1/9$</td>
<td>$-4/3$</td>
<td>$-$</td>
</tr>
<tr>
<td>$? $</td>
<td>$-1$</td>
<td>$1$</td>
<td>$1$</td>
<td>$-1$</td>
<td>$0$</td>
<td>$-2/3$</td>
<td>$1$</td>
<td>$-3/2$ $\leftarrow$</td>
</tr>
<tr>
<td>$C_2$</td>
<td>$1/3$</td>
<td>$-1/3$</td>
<td>$-2/3$</td>
<td>$0$</td>
<td>$0$</td>
<td>$2/9$</td>
<td>$-$</td>
<td>$\uparrow$</td>
</tr>
</tbody>
</table>

Create $Y_2$. Save $C_2(X_1) = 1/3$ and $C_2(X_2) = -1/3$. Perform GJP on $Y_2$.

<table>
<thead>
<tr>
<th>BVS</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$Y_1$</th>
<th>$Y_2$</th>
<th>RHS</th>
<th>$L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y_1$</td>
<td>$-1/2$</td>
<td>$-1/2$</td>
<td>$1/6$</td>
<td>$1/6$</td>
<td>$1$</td>
<td>$0$</td>
<td>$-3/2$</td>
<td>$-$</td>
</tr>
<tr>
<td>$Y_2$</td>
<td>$3/2$</td>
<td>$-3/2$</td>
<td>$-3/2$</td>
<td>$3/2$</td>
<td>$0$</td>
<td>$1$</td>
<td>$-3/2$</td>
<td>$-$</td>
</tr>
<tr>
<td>$C_3$</td>
<td>$0$</td>
<td>$0$</td>
<td>$-1/3$</td>
<td>$-1/3$</td>
<td>$0$</td>
<td>$0$</td>
<td>$-$</td>
<td>$-$</td>
</tr>
</tbody>
</table>

This is the optimal tableau, so no iteration is needed in the final phase. The optimal solution is

$$X_1 = 0 + (-1)(-3/2) + (1/3)(-3/2) = 1,$$

$$X_2 = 0 + (-1)(-3/2) + (-1/3)(-3/2) = 2.$$

3.4. Gradient direction does not intersect with a face

We consider the following problem as the worst possible case of a gradient direction.

Problem P4:

Max $X_1 - X_2$

subject to $X_1 \leq 10$,

$X_1 \geq 5$,

$X_2 \leq 10$,

$X_2 \geq 5$.

Maximum possible number of iterations = 6. The following tableaux are obtained in succession:

<table>
<thead>
<tr>
<th>BVS</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$Y_1$</th>
<th>RHS</th>
<th>$L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>$1$</td>
<td>$0$</td>
<td>$1$</td>
<td>$0$</td>
<td>$0$</td>
<td>$0$</td>
<td>$1$</td>
<td>$10$</td>
<td>$-$</td>
</tr>
<tr>
<td>$? $</td>
<td>$1$</td>
<td>$0$</td>
<td>$0$</td>
<td>$-1$</td>
<td>$0$</td>
<td>$0$</td>
<td>$1$</td>
<td>$5$</td>
<td>$5$ $\leftarrow$</td>
</tr>
<tr>
<td>$S_3$</td>
<td>$0$</td>
<td>$1$</td>
<td>$0$</td>
<td>$0$</td>
<td>$1$</td>
<td>$0$</td>
<td>$-1$</td>
<td>$10$</td>
<td>$-$</td>
</tr>
<tr>
<td>$? $</td>
<td>$0$</td>
<td>$1$</td>
<td>$0$</td>
<td>$0$</td>
<td>$0$</td>
<td>$-1$</td>
<td>$-1$</td>
<td>$5$</td>
<td>$-5$</td>
</tr>
<tr>
<td>$C_1$</td>
<td>$1$</td>
<td>$-1$</td>
<td>$0$</td>
<td>$0$</td>
<td>$0$</td>
<td>$0$</td>
<td>$2$</td>
<td>$-$</td>
<td>$\uparrow$</td>
</tr>
</tbody>
</table>
Create $Y_1$. Save $C_1(X_1) = 1$ and $C_1(X_2) = -1$. Perform GJP on $Y_{12}$.

<table>
<thead>
<tr>
<th>BVS</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$Y_1$</th>
<th>$Y_2$</th>
<th>$Y_3$</th>
<th>RHS</th>
<th>$L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>$Y_1$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>$S_3$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>15</td>
<td>-</td>
<td>-5</td>
</tr>
<tr>
<td>?</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>-2</td>
<td>10</td>
<td>-5</td>
<td>-</td>
</tr>
<tr>
<td>$C_2$</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Create $Y_2$. Save $C_2(X_1) = -1$ and $C_2(X_2) = -1$. Perform GJP on $Y_{24}$.

<table>
<thead>
<tr>
<th>BVS</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$Y_1$</th>
<th>$Y_2$</th>
<th>$Y_3$</th>
<th>RHS</th>
<th>$L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>$Y_1$</td>
<td>1/2</td>
<td>-1/2</td>
<td>0</td>
<td>-1/2</td>
<td>0</td>
<td>1/2</td>
<td>1</td>
<td>0</td>
<td>-1/2</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$S_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>$Y_2$</td>
<td>-1/2</td>
<td>-1/2</td>
<td>0</td>
<td>1/2</td>
<td>0</td>
<td>1/2</td>
<td>0</td>
<td>1</td>
<td>1/2</td>
<td>-5</td>
<td>-10</td>
</tr>
<tr>
<td>$C_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Create $Y_3$. Perform GJP on $Y_{31}$.

<table>
<thead>
<tr>
<th>BVS</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$Y_1$</th>
<th>$Y_2$</th>
<th>$Y_3$</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y_3$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>$Y_1$</td>
<td>1/2</td>
<td>-1/2</td>
<td>1/2</td>
<td>0</td>
<td>0</td>
<td>1/2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>5/2</td>
</tr>
<tr>
<td>$S_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>$Y_2$</td>
<td>-1/2</td>
<td>-1/2</td>
<td>-1/2</td>
<td>0</td>
<td>0</td>
<td>1/2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-15/2</td>
</tr>
<tr>
<td>$C_4$</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

This is the optimal tableau. The final solution is

\[
X_1 = 0 + 1(5/2) + (-1)(-15/2) = 10,
\]

\[
X_2 = 0 + (-1)(5/2) + (-1)(-15/2) = 5,
\]

\[
S_1 = 0, \quad S_2 = 0, \quad S_3 = 5, \quad S_4 = 0.
\]

4. Special cases

4.1. Unbounded solutions

Sometimes the feasible region of an LP is unbounded and the objective function can be made infinitely large without violating any of the constraints. In a manner similar to the simplex this unboundedness of a problem is captured by algorithm GP through the detection of the existence of a PC, i.e., $Y_k$, with all negative entries.

Problem P5:

\[
\text{Max } X_1 + X_2
\]

subject to

\[
-X_1 + X_2 \leq 1,
\]

\[
-2X_1 + X_2 \leq 0.
\]
Starting from initialization, the following tableaux are obtained successively:

<table>
<thead>
<tr>
<th>BVS</th>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( S_1 )</th>
<th>( S_2 )</th>
<th>( Y_1 )</th>
<th>RHS</th>
<th>( L )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_1 )</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>?</td>
<td>2</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( C_1 )</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note that there is no reduction in dimensionality (Step F9). Drop the first constraint in the second tableau with \( C_2 \). Again, \( Y_2 \) is the entering variable, but has only a negative entry in its column, indicating an unbounded solution.

4.2. Infeasibility

Sometimes there is no solution that satisfies all the constraints of a problem. Algorithm GP identifies this as follows. Note that all entries under RHS corresponding to the slack variables must always be non-negative in all phases. The presence of a negative value at any time would indicate infeasibility.

Problem P6:
Max \( 2X_1 + 2X_2 \)
subject to \( X_1 + X_2 \geq 2, \)
\( X_1 + X_2 \leq 1. \)

The following tableaux are obtained:

<table>
<thead>
<tr>
<th>BVS</th>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( S_1 )</th>
<th>( S_2 )</th>
<th>( Y_1 )</th>
<th>RHS</th>
<th>( L )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Y_2 )</td>
<td>-1/3</td>
<td>1/3</td>
<td>1/3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1/3</td>
</tr>
<tr>
<td>( Y_1 )</td>
<td>1/3</td>
<td>2/3</td>
<td>5/3</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>5/3</td>
</tr>
<tr>
<td>( C_3 )</td>
<td>10/3</td>
<td>-4/3</td>
<td>-13/3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Create \( Y_1 \). Save \( C_1(X_1) = 2 \) and \( C_1(X_2) = 2 \). Perform GJP on \( Y_{11} \).

<table>
<thead>
<tr>
<th>BVS</th>
<th>( X_1 )</th>
<th>( X_2 )</th>
<th>( S_1 )</th>
<th>( S_2 )</th>
<th>( Y_1 )</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Y_1 )</td>
<td>1/4</td>
<td>1/4</td>
<td>-1/4</td>
<td>0</td>
<td>1</td>
<td>1/2</td>
</tr>
<tr>
<td>( S_2 )</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>( C_2 )</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

This tableau has a slack variable, \( S_2 \), with a negative RHS, indicating infeasibility.
4.3. Alternate/multiple solutions

An LP problem on occasion; may have two or more alternate optimal solutions. The proposed algorithm provides a clear indication of the presence of alternative optimal solutions upon termination. The decision-maker now has the option of deciding which optimal solution to implement on the basis of other factors involved. Like the simplex method, an LP has multiple solutions if at least one of the $c_k$ entries for the non-basic variables is zero.

Problem P7:

$$\text{Max } 2X_1 + 2X_2$$
$$\text{subject to } -X_1 + X_2 \leq 1,$$
$$X_1 + X_2 \geq 2,$$
$$X_1 - X_2 \leq 1.$$ 

Here $m = 3$, therefore we need at most three iterations. However, we get the optimal tableau by one iteration.

Here is the optimal tableau:

<table>
<thead>
<tr>
<th></th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$Y_1$</th>
<th>RHS</th>
<th>$L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>$S_2$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>1/2</td>
</tr>
<tr>
<td>$S_3$</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>$C_1$</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Create $Y_1$. Save $C_1(X_1) = 2$ and $C_1(X_2) = 2$. Perform GJP on $Y_{12}$.

Here is the resulting optimal tableau:

<table>
<thead>
<tr>
<th></th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$Y_1$</th>
<th>RHS</th>
<th>$L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$Y_1$</td>
<td>1/4</td>
<td>1/4</td>
<td>0</td>
<td>-1/4</td>
<td>0</td>
<td>1</td>
<td>1/2</td>
<td></td>
</tr>
<tr>
<td>$S_3$</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$C_2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This is an optimal tableau. The zeros in the $C_2$ row for non-basic variables $X_1$ and $X_2$ indicate a presence of multiple solutions. The optimal solution from this tableau is $X_1 = 1$, $X_2 = 1$. Since $S_1 = 1$, and $S_3 = 1$, we have more positive variables than the number of constraints, indicating that the optimal solution is not at a vertex. To generate the two vertices, bring in $X_1$ in the above tableau to get $X_1 = 3/2$, $X_2 = 1/2$; or, bring in $X_2$ to get $X_1 = 1/2$, $X_2 = 3/2$.

4.4. Treatment of degeneracy

Degeneracy is a phenomenon that can arise in the simplex method, see Saigal [11]. If it does arise; it certainly influences any vertex-searching method. However, algorithm GP is not a vertex-searching method. Therefore, degeneracy is a very rare occurrence in the searching of an optimal solution. If it occurs, one can perturb the component of the gradient by $\varepsilon$-procedure or take any other anti-degeneracy precaution which works for the simplex method, see Dantzig [12].

5. Conclusions

A lot of research has been done to find a faster Simplex-like algorithm that can solve linear programming (LP) problems [13–31]. All these research activities are aimed at improving the pivoting rules. In this paper, we present a new approach to the problem of improving the pivoting algorithms. This paper develops a gradient
method, algorithm (GP), for solving general LP problems. GP is easy to use and does not introduce any artificial variables if the origin is not feasible, as in the simplex method’s use of big-M. Therefore, the proposed solution algorithm is computationally practical and stable to implement. The algorithm works in the space of the original variables. Algorithm GP is versatile in that it can identify clerical errors in data entry or data manipulation by indicating that the problem is either infeasible or unbounded.

The computational procedures developed here have been embedded within otherwise simplex-like tableaux. Starting at the origin in the initialization phase, the initial tableau itself provides the first hit with a constraint hyper-plane using the full gradient of the objective function direction. The search continues using updated full gradient until a basic feasible solution is obtained. This is then followed by a series of pivotal steps in the feasible space leading to an optimal solution, if one exists. At each step, the updated sub-gradient provides the desired direction of motion, its computational basis being the Gauss–Jordan pivoting used in the simplex method.

Acknowledgement

I am most appreciative to the reviewers for their useful comments and suggestions. Parts of this work have been supported by The NSF Grant CCR-9505732.

References


