1994

# Another look at the simplex method in linear programming

Roger N. Pederson
*Carnegie Mellon University*

# ANOTHER LOOK AT THE SIMPLEX METHOD IN LINEAR PROGRAMMING

by

R. N. Pederson

Department of Mathematics
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Research Report No. 94–168

June, 1994

# Another Look at the Simplex Method in Linear Programming

R. N. Pederson

"Notation is important. It can even solve problems. But, at some point, you must do some work yourself." K. O. Friedrichs.

## 1. Introduction and Statement of the Problem.

Without using any symbols at all, we can give a precise statement of the problem by saying that it is to find the maximum, if it exists, of a linear function of a finite number of real variables on a convex plane polyhedron of the same variables. The simplex method of solving the problem is then to find a vertex of the polyhedron and then to proceed along edges from one vertex to the next, in a manner that the linear function increases, until the maximum is reached. All the data needed to state and solve the problem can be stored in an $(m+1) \times (n+1)$ matrix $\mathcal{A}$.

The analytical statement of the problem then is to find the maximum of the objective function

$$\sum_{j=1}^{n} \mathcal{A}_{m+1,j} X_j + \mathcal{A}_{m+1,n+1} \tag{1.1}$$

subject to the constraints

$$\sum_{j=1}^{n} \mathcal{A}_{ij} X_j + \mathcal{A}_{i,n+1} \geq 0, \quad i = 1, ..., m. \tag{1.2}$$

By defining $A$ to be the matrix comprising the first $m$ rows and first $n$ columns of $\mathcal{A}$ and $b$ to be the transpose of $\mathcal{A}_{1,n+1},..,\mathcal{A}_{m,n+1}$, the constraint (1.2) takes the simpler form

$$Ax + b \geq 0, \tag{1.3}$$

meaning, of course, that each component of the column vector is non-negative. The vector $x$ is superfluous for the purpose of applying the simplex algorithm. But, working only with the matrix $\mathcal{A}$, can lead to misconception as we shall see in the next section. But first, let us find another notation for the constraint set by using $\mathcal{A}_i$ to denote the rows of $A$. Then (1.3) can be replaced by

$$L_i(x) = (\mathcal{A}_i, x) + b_i \geq 0, \quad i = 1, ..., m \tag{1.4}$$

where $b_i$ is the $i^{\text{th}}$ coordinate of $b$ and $(,)$ represents the canonical inner product.

## 2. But, Those Slack Variables are Unnecessary.

Let us re-write (1.3) as

$$(A\ C)\left(C^{-1}X\right) + b \geq 0 \tag{2.1}$$

where $C$ is any non-singular $n \times n$ matrix, noting that this does not require an equality. Now, assuming $A$ has rank $n$, we may apply elementary column operations to reduced echelon form. If $C$ is the product of the corresponding elementary column matrices and $y = C^{-1}X$, the first $n$ coordinate of (2.1) are

$$y_i + b_i \geq 0. \tag{2.2}$$

Then, by making the translation $z_i = y_i + b_i$, we may assume the constraint set to be in, what is commonly called, canonical form. Furthermore, if for one $j, 1 \leq j \leq n$, we put $x_j = z_j - \beta_j$ in (1.1),(1.2) we see that this corresponds to multiplying the $j^{\text{th}}$ column of the full matrix $\mathcal{A}$ by $\beta_i$ and subtracting it from the $(n+1)^{\text{th}}$ column; that is, it is an elementary column operation. I prefer doing elementary row operation on the transpose. Thus the simplex method reduces to transposing the matrix $\mathcal{A}$ and applying elementary row operation until the first $n$ column are in reduced echelon form, with the restriction that the pivots are to be picked from the first $n$ rows of $\mathcal{A}^T$. The only question that remains is when to start using the simplex pivoting strategy. After the system is in canonical form, we must use the simplex strategy; before that we may use instead the standard Gaussian Elimination Strategy. Note that the simplex strategy requires picking the maximum positive element of the current column and hence is a partial pivoting strategy. We shall have more to say about this in Section 5.

## 3. Empty Sets, Redundant Constraints and Lower Dimensional Sets.

Let us now suppose that the normals of the first $n$ constraints form a linearly independent set. Then, for any $k > n$,

$$\mathcal{A}_k = \sum_{i=1}^{n} \alpha_{ki} \mathcal{A}_i \tag{3.1}$$

and hence

$$L_k(x) = \sum_{i=1}^{n} \alpha_{ki} L_i(x) + \triangle_k \tag{3.2}$$

with

$$\triangle_k = \mathcal{A}_{k,n+1} - \sum_{i=1}^{n} \alpha_{ki} \mathcal{A}_{i,n+1}. \tag{3.3}$$

It follows from (1.4) and (3.2) that if $(\alpha_{k1}, ., ..., \alpha_{k,n}, \triangle_k)$ are all non-negative, the $k^{\text{th}}$ constraint is redundant and that if they are all negative the set is empty. If for some $i \leq n$, $\alpha_{ki} > 0$, $\alpha_{kji} \leq 0$ for $j \neq i$ and $\triangle_k < 0$, then the $i^{\text{th}}$ constraint is redundant.

In all other cases where none of the numbers $(\alpha_{k1}, ., ..., \alpha_{k,n}, \triangle_k)$ is zero it is easily shown that the set formed for the first $n$ and the $k^{\text{th}}$, is non-empty. The other important special case occurs when $\triangle_k = 0$ and $\alpha_{ki} \leq 0$ for $i = 1, ..., n$. Then the entire constraint set is contained in the set where $L_n(x) = 0$. Hence, we may use this constraint to eliminate a variable and obtain a lower dimensional set. This means that, by reducing the number of dimensions, we may assume that this case does not occur.

We note from (3.2) and (3.3) that, when the constraint set is in canonical form, $\mathcal{A}_{i,n+1} = 0$, $i = 1, ..., n$, so the $\alpha_{ki}$'s and $\triangle_k$ are just the coefficients of the constraint equation. From this point on we shall assume that the set is in canonical form. The origin will be called the basic vertex, the first $n$ constraints the basic constraints and the rest of the constraints the non-basic constraints.

## 4. The Simplex Algorithm with a Non-Degenerate Basic Vertex.

A vertex which is the intersection of more than $n$-planes is called a degenerate vertex. This means that, when the basic vertex is non-degenerate, all of the non-basic constraints have non-zero constants. The simplex strategy then is to increase

by one the number of positive constants among these until they are all positive and then to increase the constant in the objective function.

Let us assume that the constraints are ordered so that

$$\begin{aligned} &\mathcal{A}_{i,n+1} > 0, \quad i < p \qquad \text{and if } p < m \\ &\mathcal{A}_{i,n+1} < 0, \quad p \le i \le m. \end{aligned} \tag{4.1}$$

Our first objective is to increase $p$ by one when it is less than $m$. The first step is to choose $k$ to maximize

$$\{\mathcal{A}_{p,j} : \mathcal{A}_{p,j} > 0, \ \ 1 \le j \le n\}. \tag{4.2}$$

When $p \le m$, the results of Section 3 insure that we may assume the above set to be non-empty; when $p = m + 1$, it is only empty when we have found the maximum.

Next, we choose $\ell$ to maximize the negative numbers

$$\left\{ \frac{\mathcal{A}_{\nu,n+1}}{\mathcal{A}_{\nu,k}} : \nu \le p-1, \ \mathcal{A}_{\nu,k} < 0 \right\}. \tag{4.3}$$

Suppose that the above set is empty. If $p = m + 1$ and $\mathcal{A}_{m+1,k} > 0$ there is no maximum while if $\mathcal{A}_{m+1,k} < 0$ we may set $x_k = 0$ and continue in one less dimension. If $p \le m$ we simply set $\ell = p$.

Next, we interchange the $\ell^{\text{th}}$ non-basic constraint with the $k^{\text{th}}$ basic constraint and put the constraint set back into canonical form. This requires applying Gaussian elimination to the $k^{\text{th}}$ column of $\mathcal{A}^T$. The new elements of the matrix then are

$$\mathcal{A}'_{\ell,k} = \frac{1}{\mathcal{A}_{\ell,k}} \tag{4.4}$$

$$\mathcal{A}'_{\ell,j} = -\frac{\mathcal{A}_{\ell j}}{\mathcal{A}_{\ell,k}}, \ \ j \ne k \tag{4.5}$$

and when $i \ne \ell$,

$$\mathcal{A}'_{i,k} = \frac{\mathcal{A}_{i,k}}{\mathcal{A}_{\ell,k}}, \tag{4.6}$$

$$\mathcal{A}'_{ij} = \mathcal{A}_{ij} - \frac{\mathcal{A}_{ik}\mathcal{A}_{\ell j}}{\mathcal{A}_{\ell,k}}, \ \ j \ne k. \tag{4.7}$$

In particular,

$$\mathcal{A}'_{\ell,n+1} = -\frac{\mathcal{A}_{\ell,n+1}}{\mathcal{A}_{\ell,k}} > 0 \tag{4.8}$$

since, whether $\ell = p$ or $\ell < p, \mathcal{A}_{\ell,n+1}$ and $\mathcal{A}_{\ell k}$ have opposite signs. If $i \neq \ell$ and $\ell < p-1$, we see from (4.7) that $\mathcal{A}'_{ijn+1}$ is the sum of two positive numbers when $\mathcal{A}_{ijk} > 0$ and that when $\mathcal{A}_{ik} < 0$ it is positive as a consequence of the choice (4.3) of $\ell$. Hence, in any case, the first $p-1$ constants remain positive and if $\ell = p, \mathcal{A}_{p,n+1}$ is also positive and we have increased $p$ by one. But we also see from (4.7) that if $\ell < p$,

$$\mathcal{A}'_{p,n+1} > \mathcal{A}_{p,n+1} \tag{4.9}$$

Since the constraint set has only a finite number of vertices, we shall, in a finite number of steps either find the set to be empty, prove that $\mathcal{A}'_{p,n+1} > 0$ or arrive at a degenerate vertex.

## 5. The Case of a Degenerate Vertex.

The case of a degenerate vertex occurs when there are zero constants $\mathcal{A}_{i,n+1} = 0$. Suppose that we apply the previous strategy to the basic constraints and the non-basic constraints with non-zero constants. Then we see from (4.7) that when $\mathcal{A}_{i,n+1} = 0$,

$$\mathcal{A}'_{i,n+1} = -\left(\frac{\mathcal{A}_{\ell j}}{\mathcal{A}_{\ell k}}\right)\mathcal{A}_{ik} \tag{5.1}$$

and since $\mathcal{A}_{\ell j} < 0$, $\mathcal{A}_{\ell k} > 0$, we have $\mathcal{A}'_{i,n+1} > 0$ whenever $\mathcal{A}_{i,k} > 0$. There is no reason that this should be the case, but, by applying the simplex strategy to the first $n$ column of $\mathcal{A}$, with the $k^{\text{th}}$ playing the roll of the constants, we can use the simplex strategy to achieve this. Because the algorithm is slightly more complicated when the degeneracy is of higher order, it is convenient to introduce constants $\alpha_k, \beta_k$ satisfying, after reordering the constraints and variables

$$\begin{aligned} \mathcal{A}_{i,k} &= 0, \quad n+1 \leq i < \alpha_k \\ &> 0, \quad \alpha_k \leq i < \beta_k \\ &< 0, \quad \beta_k \leq i < \alpha_{k+1} \end{aligned} \tag{5.2}$$

with $\alpha_{n+2} = m$. The cases $\alpha_k = n+1$, $\beta_k = \alpha_k$ and $\beta_k = \alpha_{k+1}$ are used to indicate that the corresponding set is empty.

Now we apply the following algorithm to the constraint set in canonical form.

[1] $k = n + 1$

[2] Reorder the constraints so that (5.2) is satisfied.

Now, we are ready to pick the current constraint indexed by $p$. The choice agrees with 4.1 when $k = n + 1$.

[3] If $k = n + 1$ or $\beta_k < \alpha_{k+1}$, set $p = \beta_k$ and proceed to [5].

Now, when we arrive at line [4], we have $k < n + 1$ and $\beta_k = \alpha_{k+1}$. This means that the elements of the pivot column below the zeros in the $k^{\text{th}}$ row of $\mathcal{A}^T$ are all zero so we can take advantage of the remark preceding (5.1) noting that, because $k < n$ the current pivot row has already been chosen in the line [5].

[4] Replace $k$ by $k + 1$ and proceed to [7].

Now we are ready to pick the current pivot row of $\mathcal{A}^T$.

[5] Reorder the variables so that $\mathcal{A}_{p,k-1}$ maximizes the positive coefficients $\mathcal{A}_{p,i}$, $i \leq i \leq k - 1$ when it is non-empty. If it is empty proceed to [10].

If $\alpha_k = n+1$, we are ready to begin the updating subroutine. Otherwise, we decrease $k$ by 1 and return to [2].

[6] If $\alpha_k > n + 1$, decrease $k$ by 1 and return to [2].

When we arrive at line [7] we know that the $k - 1^{\text{th}}$ row of $\mathcal{A}^T$ is the current pivot row and, before updating, we must find the current pivot column.

[7] If the set $\{i < p : \mathcal{A}_{i,k-1} < 0\}$ is non-empty, choose $\ell$ to maximize the ratios $\mathcal{A}_{i,k}|\mathcal{A}_{i,k-1}$. Otherwise set $\ell = p$.

Now, we are ready to interchange the constraints indexed by $k - 1$ and $\ell$ and then put the matrix back into canonical form.

[8] Return the matrix to canonical form by applying Gaussian elimination to reduced echelon form to $\mathcal{A}^T$ using the element indexed by $\ell, k - 1$ as pivot.

We note that, since the elements $\mathcal{A}_{\ell,j}$, $j > k - 1$, are all zero the elementary row operation correspond to adding zero to the rows of $A^T$ indexed by $j > k$. Hence the $\alpha_j$'s and $\beta_j$'s, $j > k$ so they are unchanged. We now redefine the $\alpha_j$'s and $\beta_j$'s for $j \leq k$ returning to [2].

[9] Return to [2].

The program will terminate at [10].

[10] The maximum is $\mathcal{A}_{m+1,n+1}$.

We have tacitly assumed the maximum to exist, leaving to the reader the task of adding the lines, explained in Section 3, regarding empty sets, redundant constraints, lower dimensional problems and problems with no maximum.

## 6. Small Pivots and Degenerate Vertices.

In running the above algorithm, it is crucial that one distinguish between non-zero numbers and zeros represented by round-off errors. The author has studied this problem extensively on the Radio Shack Color Computer and on the Tandy 1000. Computing, respectively, to 9 and 16 places, base 10. The Random Number generator was used to supply the data and, computing to places base 10, the test for determining whether or not a number in zero was by comparison with $\gamma$, $2 \leq \gamma \leq p/2$. In order to increase the probability that the set is not empty, the probability that the origin satisfying a constraint is set at $p$, $0 \leq p \leq 1$. With no other restriction, a degenerate vertex has never been found. By building in the condition of degeneracy, e.g. by applying a similarity transformation to a known degenerate situation and adding more constraints, the program seems to work as well as in the non-degenerate case. The problem, in each case, is checked by re-running the program on the constraints forming the final basic vertex and by evaluating the objective function at the intersection of their planes.

We have also never found an ill-conditioned matrix with the random number generator. By putting in the Hilbert matrix [4], prob 169, p. 337, we find the obvious difficulty. However, by computing to a sufficient number of places, we have always been able to overcome the difficulty.

## 7. Further Methods of Speeding Up the Program.

The Simplest Method of Speeding Up the Program is to remove the redundant constraint using the test of Section 3, noting that the test requires only sign-tests of quantities that are computed anyway. Its disadvantage is that a constraint that shows up as redundant in one coordinate system does not necessarily in another. The number of degenerate constraints can be increased by adding the condition that the objective function be greater than its value at the current basic vertex.

Another method of possibly speeding up the program is to use the fact that

once a vertex has been found we know that the constraint set is non-empty. Then we can eliminate a variable using any of the constraints. If the constraint used was redundant, the new set will be empty. Otherwise, we obtain the maximum on an $(n-1)$-dimensional face. The weakness of this method is that we lose time when we use a redundant constraint to eliminate a variable. Further tests for redundance and empty sets are given in [2] and [3].

## References


[1] Dantzig, *Linear Programming and Extension,* Princeton Univ. Press.

[2] Pederson, R. N., *A Necessary and Sufficient Condition that a Non-Degenerate Linear Constraint Set be Empty or Contain a Redundant Constraint.* (Submitted for Publication).

[3] Pederson, R. N., *Sign Configurations of the Coefficients of a Linear Constraint Set and Redundant Constraints.* In preparation.

[4] Polyn, G., Szego, G., *Problems and Theorems in Analysis,* Springer-Verlag, New York, Heidelberg, Berlin, Berlin, 1972.

[5] Strang, G., *Linear Algebra and its Applications (3rd Ed.),* Harcourt, Brace, Jovanovich, San Diego.

[6] Wu, S. and Coppin, R., *Linear Programming and Extension,* McGraw Hill.



### Acknowledgement

I would like to thank Jenny Bourne Wahl for criticizing an earlier version of this manuscript.