①

REVISIONS AND EXTENSIONS TO THE SIMPLEX METHOD (With Side-lights on Programming Techniques)

Wm. Orchard-Hays

P-562 ✓

2 September 1954

[illegible]

The RAND Corporation
1700 MAIN ST. • SANTA MONICA • CALIFORNIA

SUMMARY

The simplex algorithm for solving linear programming problems may be modified in various ways to provide algorithms for various auxiliary purposes. In particular, the use of the dual system allows arbitrary changes to be made in the right hand members of the restraint equations. An allied algorithm — called parametric linear programming — allows desired changes to be introduced gradually in such a way as to maintain feasibility and optimality at all times. Similarly, changes in the optimizing form can be made either arbitrarily or in such a way as to maintain optimality. If the simplex method is viewed as being composed of certain rather comprehensive operations, then all these variations can be stated in terms of the following set of these abstract or pseudo-operations, where the order of execution may vary for different purposes.

(1) Development of a pricing vector or vectors.

(2) Choice of an index s of some column vector P_s not in the present basis.

(3) Representation of the vector P_s in terms of the present basis.

(4) Choice of an index r of some column P_{j_r} in the present basis.

(5) Change of basis, with P_s replacing P_{j_r}, or change of the right hand members of the restraint equations.

It is suggested that a breakdown of this type has important implications for flexibility and adaptability in a computer code.

REVISIONS AND EXTENSIONS TO THE SIMPLEX METHOD
(With Side-lights on Programming Techniques)

Since this session emphasizes computing techniques, I will take a little time to explain a problem which I believe should be more widely understood by people who require large-scale computations. As this concerns the economics of machine utilization, it is perhaps not entirely inappropriate.

When the original formulator of a large problem — usually, and hereafter, called the 'customer' — brings his monstrous brain child to be prepared for machine calculation, he is full of assurances that this example is the forerunner of several entirely similar ones for which the same program can be used. The customer is entirely honest in this but unfortunately what usually happens is the following. Just as the routine is nearly finished, the customer rushes down with some 'trivial' last-minute changes. These having been more or less graciously accommodated and the code finally checked out on the machine, it turns out that certain numerical difficulties crop up. Change is piled on change and eventually an acceptable set of answers is obtained but the code has lost all elegance or clarity which it may ever have possessed. In the meantime, of course, the customer has been pondering deeply over the imperfections in his method which these difficulties have revealed and, when he presents the next example, he has changed a few formulas, refined a few rules, but these he suggests are mere details which the coder can incorporate in the program before running the next case. At this point, the programmer must consider whether it is not cheaper to start all over.

Now having played both the role of programmer and, in part, of customer, I have repeatedly been faced with the problem of making extensive changes in a large code. One attempt at making these less painful and less expensive has come about as a result of considerable experience in coding up routines for linear programming problems to be solved with the simplex method on the IBM 701 [1e,f]. Since such a code may involve upwards of 5000 individual commands in its entirety, the problem is by no means a simple one.

One important phase of the programming art is the use of what are called assembly routines. Since, for many reasons, it is virtually impossible, in practice, to write down the required list of commands in machine code, some sort of convenient language intermediate between ordinary written instructions and machine commands is devised. Then a program is prepared — at considerable initial cost — which can 'read' the intermediate language and translate it into machine code. Subsequent problems are then coded in the intermediate language and, as a preliminary step in the computations, the code is processed by the assembly routine. Everytime it is necessary to re-assemble the code, the cost of the code is increased by machine time as well as programmers' time.

In all programs which are to be assembled, the notion of regions is prominent. A region is a group of consecutive memory cells set aside for some more or less well-defined set of quantities — commands or numbers or sometimes both. Regions are usually simply created at will as need arises, the reason that they are needed at all being that the logical layout of a complex procedure can be conceptualized only in terms of functional parts that make up the whole. Once a region has been coded up and checked out (with of course appropriate devices for implementing its use), it takes on a new character — it becomes in fact an abstract, or pseudo-operation which is henceforth available for use in future problems. By a <u>pseudo-operation</u> is meant some arithmetic, logical, or utility activity which requires more than one actual machine order.

For semantic purposes, let us denote adjustments in a program necessitated by coding errors as <u>corrections,</u> and those necessitated by procedural changes as <u>modifications</u>. Then once a particular region — i.e. a pseudo-operation — has been checked out, there are no more corrections to be made provided, of course, that the assumptions on which it is based are always observed. Now procedural changes do not usually involve any modification in lower order pseudo-operations such, for example, as adding two vectors together. The modifications occur — or should do so if proper planning is done — in what are called control regions, the higher order abstractions which control the order and frequency of use of the subservient regions. The big difficulty is that there is often not a sufficiently comprehensive picture of the whole problem shared by the customer and coder at the outset to allow proper planning, that is, the control regions do not bear the proper relationship to the problems at hand.

At this point I must mention that it is possible to use a pseudo-code which is never assembled into machine language at all but which requires the use of an interpretive routine for the actual running of a problem — a method widely practiced. This interpretive routine usually recognizes some fairly comprehensive list of commands but requires that a great deal be explicitly specified and is very time-consuming in operation for large, special-purpose codes. Moreover, extensive changes are nearly as difficult in a universal pseudo-code as in machine code. What is needed is a pseudo-code which takes advantage of the extensive background of conventions which inevitably arises in protracted computations and calculations along fixed lines. However, it should be possible to couple together various parts of the code on demand, just as an interpretive routine does. In short, the code should be analogous to the mathematician's thought processes. In this way variations in the structure of the main procedure are easily arranged with a minimum of coding and check-out time, and time is not wasted in repeatedly interpreting the same thing while running the job. We now have a 701 code based on this philosophy. The pseudo-operations mentioned below reflect its major structure.

Time limitations force me to make the seemingly reasonable assumption that those interested in this paper are already familiar with the simplex method. However, since we use a slightly different algorithm than most others, let me just briefly sketch the way in which we set up the computations [1a,c,e,f]. There is, first of all, a linear form to be optimized, and I will convene that, as written, it is to be minimized.

$$(6) \qquad \sum_{j=1}^{n} a_{oj}x_j = \text{min.}$$

This is to be done subject to two classes of restraints on the n variables x_j.

$$(7) \qquad x_j \geq 0 \;, \quad j = 1, \ldots, n,$$

$$(8) \qquad \sum_{j=1}^{n} a_{ij}x_j = b_i \;, \quad i = 1, \ldots, m,$$

where the a_{ij} and b_i ($i = 0, 1, \ldots, m$; $j = 1, \ldots, n$) are specified and we further assume, without loss of generality, that all the $b_i \geq 0$.

Let us assemble all the a_{ij} from (6) and (8) into one $(m+1)\text{X}(n+1)$ matrix P whose first column we make a unit vector.

$$(9) \quad P = \begin{bmatrix} 1 & a_{o1} & \cdots & a_{on} \\ 0 & a_{11} & \cdots & a_{1n} \\ \vdots & \vdots & & \vdots \\ 0 & a_{m1} & \cdots & a_{mn} \end{bmatrix} = \left[P_o, P_1, \ldots, P_n\right].$$

The columns we have called P_j ($j = 0, 1, \ldots, n$) where P_o is the unit vector. With P_o we will associate a new variable x_o and form the column of variables

$$(10) \quad X = \left\{x_o, x_1, \ldots, x_n\right\}.*$$

*Braces will be used to denote column vectors and parentheses to denote rows.

The right hand side will be denoted by the column vector

(11) $$Q = \left\{ 0,\ b_1,\ \ldots,\ b_m \right\}.$$

Now the problem may be stated as:

Maximize x_0 subject to (7) and the matrix equation

(12) $$PX = Q.$$

I will ignore the questions of determining the rank of P and obtaining some feasible solution to (12), i.e. one which satisfies (7). All this is taken care of by phase one of the simplex method, when necessary [1a,c;2]. Let us merely assume that P has rank m+1 and that a basic feasible solution is at hand which consists of a non-singular basis matrix B and an associated solution vector $V = B^{-1}Q$ where

(13) $$B = \left[P_{j_0},\ P_{j_1},\ \ldots,\ P_{j_m} \right] \qquad (P_{j_0} = P_0)$$

is formed from a subset of the columns P_j. Let the values of the corresponding variables x_{j_i} $(i = 0, 1, \ldots, m)$ be v_i, with all other $x_j = 0$. That is,

(14) $$V = \left\{ v_0,\ v_1,\ \ldots,\ v_m \right\} = B^{-1}Q. \qquad (v_0 = x_0)$$

Further denote the elements of B^{-1} by β_{ik},

(15) $$B^{-1} = (\beta_{ik})$$

and any particular row of B^{-1} by β_i,

(16) $$\beta_i = (\beta_{i0},\ \beta_{i1},\ \ldots,\ \beta_{im}).$$

Note that the first column of B^{-1} is also a unit vector, that is,

$$\beta_{00} = 1; \quad \beta_{i0} = 0,\ i \neq 0.$$

The first row β_0 of the inverse is called a pricing vector.

If we define

(17) $$\delta_j = \beta_0 P_j = \sum_{k=0}^{m} \beta_{0k} a_{kj} \qquad j = 1, \ldots, n$$

then the <u>first simplex criterion</u> is stated as follows:

(18) If $\delta_j < 0$ for any j, then the value of x_0 will be increased (or at least not decreased) if P_j is introduced into B.

If all $\delta_j \geq 0$, then $x_0 = v_0$ is max.

If some $\delta_j < 0$, the usual rule is to choose an index s by

(19) $\delta_s = \min_j \delta_j < 0$ taking smallest index in case of ties.

Then P_s is introduced into B. In order to choose an index r of some vector P_{j_r} in B which P_s is to replace, we express P_s in terms of B.

(20) $B^{-1}P_s = Y = \left\{ y_0, y_1, \dots, y_m \right\}$.

The <u>second simplex criterion</u> (in its simplified form) is now expressed as:

If any $y_i > 0$ for $i \neq 0$,

let $\theta_i = v_i/y_i$ for those $y_i > 0$ and $i > 0$

$= +\infty$ otherwise.

(21) Then $\theta_r = \min_i \theta_i$ taking smallest index in case of ties.

If all $y_i \leq 0$, then the value of x_0 has no upper bound.

Assuming that x_0 is bounded, then P_{j_r} is eliminated from B and replaced by P_s giving, after the change of basis, the new solution

(22) $B(V - \theta_r Y) + \theta_r P_s = \overline{BV} = Q$,

with a new value of x_0,

(23) $\overline{v}_0 = v_0 - \theta_r \delta_s \geq v_0$.

Those familiar with the simplex theory will note that, in using the simplified second criterion, I have ignored the possibility of endless cycling through a recurring series of bases — a matter which has been discussed at considerable length in various papers [1a, 2, 3, 4]. We feel, however, that this is a very improbable eventuality there being, to date, only two extremely synthetic examples (by Alan Hoffman and Philip Wolfe) in which cycling has occurred. It is my view that non-convergence of the simplex iterative process is only the limiting case of slow convergence. Until we understand more clearly why some problems converge quickly and others — which appear to be of similar structure — take more iterations, there seems little point in complicating a code to provide for the unlikely case while ignoring the more frequent difficulty of slow convergence. While either Dantzig's or Charnes' method [1a, 2] for rigorously resolving ties in choosing the index r will prevent absolute degeneracy — and hence avoid cycling — they still permit an impossible number of iterations, from a practical standpoint, in bad cases.

We have found it advantageous not only to transform just the basis instead of the whole matrix P on every iteration, but also to express B^{-1} as a product of elementary column matrices [1e, 4]. These elementary matrices are formed, conceptually, from a unit matrix by replacing the rth column by a vector $\eta = \left\{ \eta_0, \eta_1, \ldots, \eta_m \right\} = \bar{B}^{-1} P_{j_r}$ where $\eta_r = 1/y_r \neq 0$. Actually it is only necessary to record the index r and the vector η from each iteration. This reduces considerably the amount of writing necessary to record B^{-1} and requires less reading for the first several iterations. This form of the inverse is also convenient for several other purposes, including calculating the effects of various transformations on a system and facilitating checking and restarting procedures — the latter being extremely important in practice. This product form has been written up and circulated rather widely [5] so that I shall not develop the details of it here. It seemed necessary to remark on it however, since it requires the redevelopment of the pricing vector β_0 on each iteration and certain statements to follow might not be clear unless this fact were understood. Similarly, when a P_s is chosen, it is still in its original form and must actually be transformed into the vector Y by applying B^{-1}.

In the form which I have outlined, then, one iteration of the simplex method can be considered as consisting of the following five pseudo-operations.

(i) Develop β_i for $i = 0$.

(ii) Compute δ_j and either choose a δ_s or terminate. (Optimum attained)

(iii) Compute $Y = B^{-1}W$ for $W = P_s$.

(iv) Choose the index r (or terminate if x_0 unbounded.)

(v) Make the change of base, constructing a new η vector and transforming V.

As will appear, this is a sufficient breakdown of the whole complex of operations to permit discussion of important variations in the method.

The solution of a linear programming problem is often merely the first step in the investigation of a larger economic or administrative problem. As one example, a linear program may contain a parameter which can be changed to give different linear approximations to some non-linear relationship among the variables. If the non-linear function is continuous, then it is expected that optimal programs will differ only slightly for small changes in the parameter. It is clearly desirable to be able to introduce slight variations using a previous optimal solution as a starting point, rather than to consider each parametric change as a whole new programming problem.

There are three essentially distinct questions of this kind each of which can be approached in two ways. Of the resulting six problems, the following four are the most amenable to automatic computing schemes. We refer to these as <u>post-optimality problems</u> and the techniques for handling them as <u>parametric linear programming</u> or PLP.

Given an optimal solution BV = Q to the problem of maximizing x_0 subject to (7) and (12), then:

Pr.1) How much can Q be changed in some specified way before the basis B is non-feasible, i.e. before the necessary change in V makes some $v_i < 0$?

Pr.2) If an arbitrary change is made in Q and the basis becomes non-feasible, how can this be corrected or it be determined that the whole new problem is non-feasible?

Pr.3) How much can the form (6) be changed in some specified way before the solution is no longer optimal, i.e. before the necessary change in β_0 makes some $\delta_j < 0$?

Pr.4) If some arbitrary change is made in the a_{oj}'s, how can this be accounted for in B^{-1} (more particularly in β_0) so that vectors not in the basis — or even in the original system — can be priced out?

We could also ask a pair of similar questions about changes in the original system (8), i.e. in the matrix P with the zero-th row and the zero-th column excepted. However, for changes outside the basis B, the problems are trivial whereas, for changes within B, the general case is quite messy. It is better to improvise tricks for the particular model at hand or else start all over, in the latter case. Consequently, we will make no further mention of parameters in the left members of the restraint equations although they are used fairly frequently.

Pr. 2 can be answered very quickly. Simply make the desired change in Q and re-solve for V. If the resulting V contains no negative elements, all is well since, in any event, the pricing vector β_0 remains unchanged. Hence we still have all $\delta_j \geq 0$ and the solution remains optimal as well as feasible. If some $v_i < 0$, then apply the dual simplex algorithm to regain feasibility, making the necessary changes of basis. If this cannot be done, then no feasible solution exists for the new right hand side.

The dual algorithm, as the name implies, operates on the dual problem, but does so in terms of the primal system so that all computations can still be carried out in (m+1)-space instead of (n+1)-space which is typically much larger [1b,f]. Without attempting to reconstruct the theory of the dual, let us note the following pertinent facts concerning the complete dualized system with notation consistent with the above.

	PRIMAL	DUAL
Objective:	maximize $x_0 = -\sum_{j=1}^{n} a_{oj}x_j$	minimize $z = \sum_{i=1}^{m} b_i \beta_{oi}$
Feasibility:	$v_i \geq 0$ for $i \neq 0$	$\delta_j \geq 0$ for $j \neq 0$
Optimality:	$\delta_j \geq 0$ for $j \neq 0$	$v_i \geq 0$ for $i \neq 0$
Criterion for improvement:	$\delta_s = \min \delta_j < 0$	$v_r = \min v_i < 0$
Criterion for elimination:	$\theta_r = \min_{y_{is}>0} v_i/y_{is}$ $(i \neq 0)$	$\varphi_s = \min_{y_{rj}<0} \frac{y_{oj}}{-y_{rj}} = \min \frac{\beta_o P_j}{-\beta_r P_j}$
Unbounded case:	All $y_{is} \leq 0$	All $y_{rj} \geq 0$

The dual algorithm can now also be stated in terms of five pseudo-operations which I number in a manner analagous to those for the regular primal algorithm.

(iva) Choose the index r, or terminate. (Optimum attained)

(i) Compute β_i for $i = 0$ and r.

(iia) Choose ϕ_s, or terminate if no $\beta_r P_j < 0$. (Optimum unbounded)

(iii) Compute $Y = B^{-1}W$ for $W = P_s$.

(v) Make the change of base, constructing a new η vector and transforming V.

Pr.1 is handled by a very similar algorithm but with somewhat more finesse. Let q be the vector of changes which it is desired to make in Q. Then assume that an optimum solution BV = Q has been obtained for the restraints

(24) $PX = Q + \Theta q \qquad (x_i \geq 0, \quad i \neq 0)$

with the scalar $\Theta = 0$. We now compute the representation of $(-q)$ in terms of B.

(25) $B^{-1}(-q) = \dot{Y} = \left\{\dot{y}_0, \dot{y}_1, \ldots, \dot{y}_m\right\}$.

Using the same pseudo-operation (iv) as for the normal simplex algorithm, we now compute, providing any $\dot{y}_i > 0$,

(26) $\Theta_r = \min\limits_{i \neq 0} {}^{v_i}/\dot{y}_i$ for $\dot{y}_i > 0$.

Proceeding as though we were going to introduce $(-q)$ into B in place of P_{j_r}, we have

(27) $$\sum_{i \neq r} P_{j_i}(v_i - \Theta_r \dot{y}_i) + \Theta_r(-q) = Q + 0 \cdot q.$$

whence adding $\Theta_r q$ to both sides and considering that P_{j_r} is really still in B,

(28) $$\sum_{i \neq r} P_{j_i}(v_i - \Theta_r \dot{y}_i) + 0 \cdot P_{j_r} = B(V - \Theta_r \dot{Y}) = Q + \Theta_r q.$$

Now (28) is still a basic feasible optimal solution for the new right hand side and Θ_r is the greatest value of Θ which allows this with the present basis.

If it is desired to change Q by more than $\Theta_r q$ it will be necessary to change basis and the basis vector to eliminate ~~what~~ has already been determined, namely P_{j_r} since this is where the 'bind' occurs as Θ increases. Hence we must now determine some P_j not in B which can be used as the real P_s to replace P_{j_r} and perhaps allow Θ to increase further. Now if we had such a P_s with the representation $B^{-1}P_s = \left\{y_{os}, y_{1s}, \ldots, y_{ms}\right\}$, it would be newessary that $y_{rs} = \beta_r P_s \neq 0$. We also wish to maintain optimality if possible, that is feasibility of the dual. This means that the new $\bar{\beta}_0$ (i.e. the first row of $\bar{B}^{-1}$ where $\bar{B}$

is formed from B by replacing P_{j_r} with P_s) must have the property that $\bar{\beta}_0 P_j \geq 0$ for $j = 1, \ldots, n$. The rules for elimination give

$$\bar{\beta}_0 P_{j_r} = \bar{\delta}_{j_r} = \frac{y_{os}}{-y_{rs}} = \frac{-\beta P_s}{\beta_r P_s} .$$

Hence we should choose $y_{rs} < 0$ since $y_{os} = \delta_s \geq 0$. From this it is easy to show that the index s is chosen by the same rule (iia) as was used in the straight dual algorithm above. Hence the PLP algorithm for changes in the right hand side goes as follows, where a new pseudo-operation (vi) is needed.

(iii) Compute $\dot{Y} = B^{-1}W$ for $W = (-q)$.

(iv) Choose the index r, or terminate. (max θ unbounded with present B.)

(vi) Subtract $\theta_r \dot{Y}$ from V and add $\theta_r q$ to Q.

(i) Compute β_i for $i = 0$ and r.

(iia) Choose ϕ_s, or terminate. (max x_0 unbounded with present θ.)

(iii) Compute $Y_s = B^{-1}W$ for $W = P_s$.

(v) Make the change of base, constructing a new η vector and transforming V.

/ The following theorem is of considerable interest for this type of PLP.

<u>Theorem:</u> If the choice of r in (26) is unique (assuming some $\dot{y}_i > 0$) and

(a) if $\beta_r P_j < 0$ for some $j = 1, \ldots, n$,

<u>then there exists a finite range of θ,</u> $\theta_r \leq \theta \leq \theta_r + \varepsilon$,

<u>for which the solution obtained by replacing</u> P_{j_r} <u>in B by</u> P_s (chosen by (iia)) <u>is both feasible and optimal;</u> or

(b) <u>if all</u> $\beta_r P_j \geq 0$, <u>then there is no feasible solution</u> for $\theta > \theta_r$.

A constructive proof of this theorem is quite straightforward and even provides a formula for the value of ϵ. This proof can be found in another paper which I prepared about a year ago and will be omitted here [4].

In order to consider Prs. 3 and 4, let us denote the top row of P — i.e. the coefficients of x_0 and the minimizing form — by

$$\rho_0 = (1, a_{01}, a_{02}, \ldots, a_{0n}).$$

If certain (or all) of the a_{0j} are to be altered by proportionate amounts σ_j of a parameter ϕ, let

$$\sigma = (0, \sigma_1, \sigma_2, \ldots, \sigma_n)$$

and consider the matrix P as having its top row given by $\rho_0 + \phi\sigma$, $\phi \geq 0$.

Let a given basic feasible optimal solution to the original problem be $BV = Q$ which is then a solution for $\phi = 0$. Now if ϕ is increased, B, and hence B^{-1}, is changed. But due to the special structure of B — i.e. with its first column a unit vector — the only change in B^{-1} will also be in its top row β_0, or as an economist would say, in the shadow prices. This can be seen from the following schematic diagram of B^{-1} compared with B. The matrix A is simply B with its first row and first column deleted.

$$\begin{bmatrix} 1 & (a_{oj_1} \; \dots \; a_{oj_m}) \\ 0 & \\ \vdots & A \\ 0 & \end{bmatrix}^{-1} = \begin{bmatrix} 1 & (\beta_{o1} \; \dots \; \beta_{om}) \\ 0 & \\ \vdots & A^{-1} \\ 0 & \end{bmatrix}$$

Letting σ_{j_i} correspond to vector P_{j_i} in B, a little algebra shows the new value of β_o, call it $\bar{\beta}_o$, is

$$(29) \qquad \bar{\beta}_o = \beta_o - \phi \sum_{i=1}^{m} \sigma_{j_i} \beta_i .$$

Thus Pr. 4 is solved. The new pricing vector $\bar{\beta}_o$ given by (29) is used to price out as usual, but with the altered cost row, ρ_o + that is, the new values of the δ_j, call them $\bar{\delta}_j$, are

$$(30) \qquad \bar{\beta}_o (P_j + \phi \sigma_j P_o) = \bar{\delta}_j .$$

Pr. 3 can be stated as: find the critical value of $\phi \geq 0$ at which some $\bar{\delta}_k$ turns negative. Expanding (29) and (30), simplifying, and interpreting the question properly, gives the following criterion:

$$(31) \qquad \max \phi = \phi_s = \min_j \left\{ \frac{\beta_o P_j}{\sum_i \sigma_{j_i} \beta_i P_j - \sigma_j} \text{ for denominator} > 0 \right\}$$

If the denominator is non-positive for all j, then ϕ can be made arbitrarily large and the present basis will still be optimum. Otherwise, the index s determines (perhaps not uniquely) a vector P_s to introduce which may allow ϕ to be increased again. The regular primal criterion for choosing an index r is used to maintain feasibility if a change of basis is to be made.

I might remark in passing that the use of the product form of the inverse in machine computations is ideal for computing the row vector $\bar{\beta}_0$ in (29) or the sum vector $\sum_i \sigma_{j_i} \beta_i$ for use in the denominator of (31).

The algorithm for Pr. 4 is the same as the one for the regular simplex method once the specified amount ϕ of σ_j has been added to each a_{oj} and the correction (29) made in the inverse. For Pr. 3, we need only to substitute the computations involved in (31) for the pseudo-operations (i) and (ii), i.e. we replace the first simplex criterion with (31). This is only to be expected, of course, for once an optimum solution has been obtained to the original problem, the first simplex criterion has nothing more to offer — its function is completed.

In summary, let me again re-emphasize that organizing computational schemes in terms of fairly comprehensive pseudo-operations, such as have been indicated, is advantageous for the coder, the mathematician, and the economist. There is, of course, nothing new in the notion either of regional coding or of abstract or pseudo-code. My complaint is that the organization of codes into regions or blocks — and the devices used to couple them together and execute them — is all too often based on short-view expediency or established usage rather than on the true nature of the problems for which the program is designed. Furthermore, I suspect that this is also true for many problem formulations, apart from whether machine codes are written or not. Mathematicians, economists, and others who are potential customers for the services of a machine computing group should be interested in being as much as possible in rapport with all those engaged in solving their problems. All variations to date on the basic simplex algorithm, for instance, can be expressed in terms of just a few of our pseudo-operations, very few more than we have already mentioned. Consequently, if everyone's work is organized and planned along these lines, then modifications involve only a reshuffling of large blocks of already proven code. Furthermore, this sort of breakdown enhances the clarity of one's insight into the theory which underlies all these linear programming techniques — namely that of a system of linear inequalities in non-negative variables and its dual. (Actually, in applying the simplex method, the systems are sets of linear equalities.) For example, the parameters Θ and ϕ which we have discussed are really variables which are involved implicitl — but intimately — in any optimization problem with linear restraints in non-negative variables. When working with feasible solutions of the primal system, the variable Θ is made as large as it is possible to do and still maintain feasibility; the variable ϕ plays the same rôle for feasible solutions of the dual. In fact, as already noted, the change in the maximand x_0 for the former case is $-\Theta\delta_s \geq 0$, whereas the change in the minimand z — which is really the same variable in the complete dualized system — is given by $\phi v_r \leq 0$ in the latter case. Both Θ and ϕ are quotients with the pivot element, y_{rs}, for denominator in either the regular primal or straight dual algorithms. When both systems are feasible, i.e. when the whole dualized system is optimal, then any desirable change in basis can only give another optimal solution, both Θ and ϕ being zero and y_{rs} being any non-zero number. If changes are made in the right-hand side Q or the cost row ρ_0, then parametric programming provides slightly different formulas for θ or ϕ to maintain optimality, as discussed above.

To conclude, the complete dualized system is displayed in a form consistent with the preceding notation and arrangement of the original problem. For simplicity, it has been assumed that the basis B consists of the first m+1 columns of P. It is hoped that a careful perusal of this tableau will be as helpful to those readers who have not yet considered it as it has been to me in observing the remarkable relationships that exist in the type of problems with which we have been concerned. A full explanation of these relationships which the tableau is intended to display can be found in reference [4].

THE COMPLETE DUALIZED SYSTEM FOR THE SIMPLEX METHOD.

(column multipliers)

$$\begin{array}{c|ccccc:ccc|cc}
 & v_o & v_1 & v_2 & \cdots & v_m & 0 & \cdots & 0 & & \\
\hline
 & 1 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & = & z \\
\hline
 & & & \parallel & & & & \parallel & & & \parallel \\
\hline
\beta_{00} & 1 & a_{o1} & a_{o2} & \cdots & a_{om} & a_{o,m+1} & \cdots & a_{on} & & 0 \\
\beta_{01} & & a_{11} & a_{12} & \cdots & a_{1m} & a_{1,m+1} & \cdots & a_{1n} & & b_1 \\
\vdots & & \vdots & \text{(Primal basis)} & & & & & \vdots & = & \vdots \\
\beta_{0m} & & a_{m1} & a_{m2} & \cdots & a_{mm} & a_{m,m+1} & \cdots & a_{mn} & & b_m \\
\hline
\delta_1 & & -1 & & & & & & & & -x_1 \\
\delta_2 & & & -1 & & & & & & = & -x_2 \\
\vdots & & & & \ddots & & & & & & \vdots \\
\delta_m & & & & & -1 & & & & & -x_m \\
\hdashline
\delta_{m+1} & & & & & & -1 & & & & -x_{m+1} \\
\vdots & & & & & & & \ddots & & = & \vdots \\
\delta_n & & & & & & & & -1 & & -x_n \\
\hline
\end{array}$$

(row multipliers)

Note: The dual basis (for $m+1$ space) consists of all rows in the big main $(m+1)X(n+m+1)$ matrix except those multiplied by $\delta_1, \ldots, \delta_m$ (all zero). Making a change in the primal basis — P_s replacing P_{j_r} — implies a corresponding change in the dual basis — the row multiplied by δ_{j_r} replacing the one multiplied by δ_s, assuming the matrix P has rank $m+1$ and that one starts with a basic solution to the primal system.

REFERENCES

1. Dantzig, George B., et. al. RAND Research Memoranda

a. Dantzig, G. B.; Orden, Alex; Wolfe, Philip, "The Generalized Simplex Method for Minimizing a Linear Form under Linear Inequality Restraints," RAND RM-1264, 6 Jan. 1954.

b. ________________; ________________; "Duality Theorems," RAND RM-1265, 30 October 1953.

c. ________________; "Computational Algorithm of the Simplex Method," RAND RM-1266, 26 October 1953.

d. ________________; "On Algebraic Proof of the Min Max Theorem," RAND RM-1267, 18 December 1953.

e. ________________; Orchard-Hays, Wm., "Alternate Algorithm for the Revised Simplex Method (using a product form for the inverse)," RAND RM-1268, 19 November 1953.

f. Orchard-Hays, Wm., "The RAND Code for the Simplex Method" (Preliminary information), RAND RM-1269, 20 Jan. 1954.

g. Dantzig, G. B., "The Dual Simplex Algorithm," RAND RM-1270 4 Jan. 1954.

2. Charnes, A; Cooper, W. W.; Henderson, A., "An Introduction to Linear Programming," John Wiley & Sons, Inc., 1953.

3. Hoffman, A. J., "Cycling in the Simplex Algorithm," NBS report #2974, 16 Dec. 1953.

4. Orchard-Hays, Wm., "Background, Development, and Extensions of the Revised Simplex Method," RAND paper P-518, 30 April 1954.

5. "Mathematical Tables and Other Aids to Computation," VIII, No. 46, April '54.