J. Soc. Indust. Appl. Math.
Vol. 8, No. 4, December, 1960
Printed in U.S.A.

THE CUTTING-PLANE METHOD FOR SOLVING CONVEX PROGRAMS*

J. E. KELLEY, Jr.†

1. Introduction. Although generally quite difficult to solve, constrained minima problems are of perennial interest. There has been relatively little success in finding general computational techniques for handling them. However, useful techniques have been developed for certain small classes of these problems. One interesting class involves minimizing a continuous convex function on a closed convex set. It is known as the convex programming problem and has been the subject of numerous studies in recent years.[1] The main reason for success in this area appears to be that, with convex functions, all local minima are global minima.

In this paper we propose a method for solving convex programs which appears to be computationally efficient and works under rather general assumptions. The method is an outgrowth of a study of the Chebyshev curve fitting problem [7] and has previously been presented orally [5, 6]. Following the same path, Cheney and Goldstein [1] have arrived independently at the same method. As they point out, the basic idea involved can be traced back to Remez (see references in [1]) and has been used, in one form or another, by a number of authors.

The form of the convex programming problem to which we will address ourselves is to minimize a linear form on the compact convex set R.[2] That R is assumed to be compact is, for all practical purposes, no real restriction when the problem has a finite minimum. For if R were only closed, then we could restrict our inquiry to a compact convex subset of R which contains the minimum point.

The algorithm we develop for solving this problem involves solving a sequence (generally infinite) of linear programs. Of course, in practice the process is truncated after a suitable (finite) number of steps to obtain an approximate solution.

A connection with Gomory's integer linear programming algorithm

* Received by the editors May 4, 1959 and in revised form July 20, 1960.
This work was done while the author was with Remington Rand UNIVAC.

† Mauchly Associates, Inc., Ambler, Pennsylvania.

[1] A good survey of the work done up through early 1957 is contained in [10]. More recent work and allied topics may be found in [11] and in the references of [1, 9].

[2] This problem is equivalent to minimizing a bounded convex function, $F(x)$, on the compact convex set H. The former problem reduces to the latter with $R = H$ since the linear form is also convex. Conversely, let $R = \{(x, y) \mid M \geqq y \geqq F(x), x \in H\}$, where M exceeds the maximum of $F(x)$ for $x \in H$. Then the latter problem has the form of miniminizing the linear form y on R.

[3, 4] is shown with the result that the two algorithms may be suitably combined to solve integer convex programs.

Finally, some computational considerations are discussed and an example given to illustrate the computational process.

2. The fundamental convergence theorem. Let $G(\boldsymbol{x})$ be a continuous convex function defined on the n-dimensional compact polyhedral convex set $S = \{\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \geqq \boldsymbol{b}\}$, where $\boldsymbol{A}$ is an $m \times n$ matrix, $\boldsymbol{x}$ is an $n \times 1$ column vector of variables and $\boldsymbol{b}$ is an $m \times 1$ column vector. We assume that at every point $\boldsymbol{t}$ in S there exists an extreme support, $y = p(\boldsymbol{x}; t)$, to the graph of $G(\boldsymbol{x})$ with the property that, for some finite constant K, $\| \nabla p(\boldsymbol{x}; \boldsymbol{t}) \| \leqq K$ for all $\boldsymbol{x} \in S$. We will assume that the gradient vector is a row vector.[3] Let $\boldsymbol{cx}$ be a linear form, where $\boldsymbol{c}$ is an $n \times 1$ row vector and $\| \boldsymbol{c} \| < \infty$. Let $R = \{\boldsymbol{x} \mid G(\boldsymbol{x}) \leqq 0\}$ be nonempty and $R \subset S$.

We may state the convex programming problem formally as follows: *Find a vector* $\boldsymbol{\tau}$ *such that* $f = \boldsymbol{c\tau} = \min \{\boldsymbol{cx} \mid \boldsymbol{x} \in R\}$.

Since R is compact the minimum exists and is finite.

We now derive an algorithm for solving this problem. Let $\boldsymbol{t}$ be an arbitrary point in $S - R$. An extreme support to the graph of $G(\boldsymbol{x})$ at this point may be written in the form

$$p(\boldsymbol{x}; \boldsymbol{t}) = G(\boldsymbol{t}) + \nabla p(\boldsymbol{x}; \boldsymbol{t})(\boldsymbol{x} - \boldsymbol{t}) = y.$$

When $G(x)$ is "smooth", $y = p(\boldsymbol{x}; \boldsymbol{t})$ is simply the tangent hyperplane to the graph of $G(\boldsymbol{x})$ at $\boldsymbol{x} = \boldsymbol{t}$. We note that since $G(\boldsymbol{x})$ is convex, $p(\boldsymbol{x}; \boldsymbol{t}) \leqq G(\boldsymbol{x})$ for all $\boldsymbol{x} \in S$. Thus, if $\boldsymbol{x}$ is in R, $G(\boldsymbol{x}) \leqq 0$, so that $p(\boldsymbol{x}; \boldsymbol{t}) \leqq 0$. On the other hand, since $\boldsymbol{t}$ is not in R, $p(\boldsymbol{t}; \boldsymbol{t}) = G(\boldsymbol{t}) > 0$. Thus, the set R and the point $\boldsymbol{t}$ lie on opposite sides of the hyperplane $p(\boldsymbol{x}; \boldsymbol{t}) = 0$.

Now note that if we have a sequence $\{S_k\}$ of convex sets with the property that $S_k \subset S_{k-1}$ and $f_k = \min \{\boldsymbol{cx} \mid \boldsymbol{x} \in S_k\}$ then $f_k \geqq f_{k-1}$. Keeping this fact in mind let $S_0 = S$ and let $\boldsymbol{t}_0$ be a point in S_0 that minimizes $\boldsymbol{cx}$. (We assume that $\boldsymbol{t}_0$ is in $S_0 - R$ for otherwise $\boldsymbol{t}_0$ is a solution to the problem.) Further, let

$$S_1 = S_0 \cap \{\boldsymbol{x} \mid p(\boldsymbol{x}; \boldsymbol{t}_0) \leqq 0\}.$$

From what was said above, $\boldsymbol{t}_0$ is not in S_1 but $R \subset S_1$. We may thus find a point $\boldsymbol{t}_1$ in S_1 different from $\boldsymbol{t}_0$ that minimizes $\boldsymbol{cx}$ and $f_1 \geqq f_0$.

[3] An extreme support to the graph of $G(\boldsymbol{x})$ is an $(n+1)$-dimensional hyperplane that intersects the boundary of the convex set $P = \{(\boldsymbol{x}, y) \mid \boldsymbol{x} \in S, y \geqq G(\boldsymbol{x})\}$ and does not cut the interior of P. For every $\boldsymbol{x} \in S$ there exists an extreme support to the graph of $G(\boldsymbol{x})$ since it is convex. Thus, $y = p(\boldsymbol{x}; \boldsymbol{t})$ is a linear function and $\nabla p(\boldsymbol{x}; \boldsymbol{t})$ is simply the row vector of coefficients of the x-variables.

In general, let

$$S_k = S_{k-1} \cap \{x \mid p(x; t_{k-1}) \leqq 0\}$$

and let t_k be the point that minimizes cx on S_k. In this way we obtain sequences $\{t_k\}$ and $\{f_k\}$, and wish to know if $\{t_k\}$ contains a subsequence that converges to a point τ, in R. If there is such a converging subsequence, then it follows from the method of computation that the monotone increasing sequence $\{f_k\}$ converges to f and τ is the desired optimal solution.

To prove that $\{t_k\}$ contains the desired subsequence we proceed as follows:[4] Note first that if t_k minimizes cx on S_k then it must satisfy the inequalities

$$G(t_i) + \nabla p(t_k ; t_i)(t_k - t_i) \leqq 0 \qquad (0 \leqq i \leqq k-1).$$

Further, if $\{t_k\}$ is to have a subsequence which converges to a point in R, $\{G(t_k)\}$ must possess a subsequence which converges to zero. If the desired convergence does not occur, then there exists an $r(>0)$ independent of k such that

$$r \leqq G(t_i) \leqq \nabla p(t_k ; t_i)(t_i - t_k) \leqq K \| t_i - t_k \|$$

for $0 \leqq i \leqq k-1$. It follows that for every subsequence, $\{k_p\}$, of indices

$$\| t_{k_q} - t_{k_p} \| \leqq r/K \qquad (q < p)$$

so that $\{t_k\}$ does not contain a Cauchy subsequence. But this is impossible since S is compact. Therefore, $\{t_k\}$ contains a subsequence which converges to a point τ in S. Since it also follows that the corresponding subsequence of $\{G(t_k)\}$ converges to zero, τ must be in R.

In summary, we have just proved the following

THEOREM. *Let $G(x)$ be a continuous convex function defined on the n-dimensional compact convex set S such that at every point $t \in S$ there exists an extreme support, $y = p(x; t)$, to the graph of $G(x)$ with the property that, for some finite constant K, $\| \nabla p(x; t) \| \leqq K$ for all $x \in S$. Further, let cx be a linear form such that $\| c \| < \infty$ and let $R = \{x \mid G(x) \leqq 0\} \subset S$, with R nonempty. If $t_k \in S_k$ is such that*

$$ct_k = \min \{cx \mid x \in S_k\} \qquad (k = 0, 1, \cdots) \tag{1}$$

where $S_0 = S$ and

$$S_k = S_{k-1} \cap \{x \mid p(x; t_{k-1}) \leqq 0\},$$

then the sequence $\{t_k\}$ contains a subsequence that converges to a point τ, in R with $c\tau \leqq cx$ for all $x \in R$.

---

[4] I am indebted to P. Wolfe for suggesting this form of the convergence proof.

3. Integer convex programming. There is an interesting similarity between Gomory's integer linear programming method [3, 4] and that of the present paper that suggests how the two algorithms may be combined to solve integer convex programs. The *integer linear programming problem* asks to find a lattice point in the closed polyhedral convex set R^* that minimizes a given linear form. The *integer convex programming problem* asks the same question but without the restriction that R^* be polyhedral.

Gomory's algorithm and that of the present paper may be viewed geometrically as follows: Assuming that the problem in question has a bounded minimum, we imbed a portion of R^* containing a minimum point in a compact polyhedral convex set S_0 and minimize the linear form on S_0. This is a typical linear programming problem. We now cut off a portion of S_0 containing the just determined minimum point by passing a hyperplane (cutting plane) between the minimum point and $R(=S_0 \cap R^*)$ obtaining a new polyhedral convex set S_1. The linear form is now minimized on S_1. This process of making cuts and minimizing the linear form on the resulting polyhedral convex set is continued until in a finite (generally infinite) number of steps the solution to the integer linear (convex) programming problem is obtained.

The only essential difference between the integer linear and convex programming algorithms is the way in which the cutting planes are selected. We can make use of this fact to solve integer convex programming problems as follows[5]: Again assume the problem has a bounded minimum and imbed a portion of the feasible solution set R^* in a compact polyhedral convex set S_0. Now proceed by the integer linear programming algorithm to obtain the "best" lattice point in S_0. If this point is also in R $(= S_0 \cap R^*)$ then the process terminates. If not, introduce a cutting plane which passes between the just determined lattice point and R to eliminate it from further consideration. Begin again with the integer linear programming algorithm to obtain another lattice point. Eventually, since S_0, being compact, contains a finite number of lattice points, we must arrive at the "best" lattice point in R after a finite number of steps or obtain information that the problem is not feasible.

The only difficulty encountered in implementing this procedure is how to select a cutting plane to eliminate an unwanted lattice point. Except for a technicality (the integer linear programming algorithm requires that the linear form being minimized and the linear constraints defining $S_0, S_1, \cdots$ have integer or, by implication, rational coefficients and constant terms) we could use $p(x; t) \leqq 0$ for this purpose as in §2, where t is the lattice point being eliminated. Clearly, $p(x; t)$ may involve irrationals.

[5] This idea has also occurred to P. Wolfe.

However, we may use $p(\boldsymbol{x}; \boldsymbol{t})$ to obtain a "rational" cutting plane in the following way: Let

$$p(\boldsymbol{x}; \boldsymbol{t}) = \sum_{j=1}^{n} a_j x_j - b \leqq 0$$

and let $a_j^{(k)}$, $1 \leqq j \leqq n$, and $b^{(k)}$ be the decimal representations of a_j, $1 \leqq j \leqq n$, and b, respectively, out to k decimal places. Since S_0 is compact there exists $\lambda(> 0)$ such that $| x_j | \leqq \lambda$, $1 \leqq j \leqq n$, for all $\boldsymbol{x} = (x_j) \in S_0$. Therefore,

$$\left| \sum_{j=1}^{n} (a_j - a_j^{(k)}) x_j - (b - b^{(k)}) \right| \leqq n\lambda \max_j | a_j - a_j^{(k)} | + | b - b^{(k)} | = \mu^{(k)}.$$

Let $\bar{\mu}^{(k)}$ be the decimal representation of $\mu^{(k)}$ to k decimal places plus the quantity 10^{-k}. Clearly, $\bar{\mu}^{(k)}$ is rational and not less than $\mu^{(k)}$. Since, as is obvious, $\bar{\mu}^{(k)} \leqq \bar{\mu}^{(k-1)}$ and $\lim_k \bar{\mu}^{(k)} = 0$,

$$\sum_{j=1}^{n} a_j^{(k)} x_j - b^{(k)}$$

converges uniformly with k to $p(\boldsymbol{x}; t)$ on S_0. It follows that the halfspace defined by

$$\sum_{j=1}^{n} a_j^{(k)} x_j - b^{(k)} + \bar{\mu}^{(k)} \leqq 0 \tag{2}$$

contains the set $S_0 \cap \{\boldsymbol{x} \mid p(\boldsymbol{x}; \boldsymbol{t}) \leqq 0\}$ and, in consequence, also R. Also, since $p(\boldsymbol{t}; \boldsymbol{t}) > 0$, there must exist a finite value of k for which

$$\sum_{j=1}^{n} a_j^{(k)} t_j - b^{(k)} + \bar{\mu}^{(k)} > 0,$$

where $\boldsymbol{t} = (t_j)$. The inequality (2) for this value of k defines a cutting plane with rational coefficients and constant term which will eliminate $\boldsymbol{t}$ from further consideration and allow the integer linear programming algorithm to proceed.

The discussion of the last paragraph is somewhat academic since, in practice, usually only a rational approximation, $p^*(\boldsymbol{x}; \boldsymbol{t})$, to $p(\boldsymbol{x}; \boldsymbol{t})$ is known. In the absence of better information $p^*(\boldsymbol{x}; \boldsymbol{t}) = 0$ would probably be taken as the desired cutting plane provided it actually passed between $\boldsymbol{t}$ and R. In case $p^*(\boldsymbol{t}; \boldsymbol{t}) \leqq 0$ one must attempt to obtain a better approximation to $p(\boldsymbol{x}; \boldsymbol{t})$, determine a cutting plane by means of some other technique or stop the computation. If $p^*(\boldsymbol{t}; \boldsymbol{t}) > 0$ then, to help insure that a part of R is not eliminated from further consideration, the value of its constant term can be reduced slightly by an amount $\mu^*(< p^*(\boldsymbol{t}; \boldsymbol{t}))$. However, we are not guaranteed that the halfspace defined by $p^*(\boldsymbol{x}; \boldsymbol{t}) - \mu^* \leqq 0$ contains all of R. At the moment we have no alternative procedure.

Of course, these remarks apply equally well to both convex and integer convex programming.

In view of the foregoing and recent work in integer linear programming it should be possible to devise a complication of the cutting plane technique to solve convex programs where only some of the variables are required to be integers.

It is of incidental interest to note that while Gomory's algorithm reduces to the Euclidean algorithm when applied to a special case, our method is formally analogous to Newton's method for finding roots.

4. Computational considerations. Unfortunately, except for special forms of $G(\boldsymbol{x})$, $\boldsymbol{t}_k$ is almost never in R for any finite k. Further, we have been unable to give a useful upper bound on the deviation from optimality. One is tempted to stop the computation when $G(\boldsymbol{t}_k)$ first becomes smaller than some preassigned tolerance. This criterion seems to have worked out satisfactorily in the few examples that have been tried. However, one can conceive of cases where, although $G(\boldsymbol{t}_k)$ is very small, still $\boldsymbol{t}_k$ is arbitrarily removed from the minimum point.

Although one can concoct more complicated criteria for terminating the process, such complications are probably not worth the effort. A more difficult problem is to obtain accuracy near the termination of the process. When the process nears termination the successive cutting planes generally tend to become more and more parallel to one another. As this occurs, the determination of successive $\boldsymbol{t}_k$ becomes more and more difficult due to the limited precision with which the computations must be performed. Thus, there is a point where continued computation would cause the process to degenerate completely, oscillate or converge on the wrong solution. Some computational studies are required to resolve this difficulty.

The ease with which one can approximate the solution of a convex program depends primarily on the facility with which $\nabla p(\boldsymbol{x}; \boldsymbol{t}_k)$ and the solution to (1) can be determined. If $G(\boldsymbol{x})$ is "smooth" then it is clear that $\nabla p(\boldsymbol{x}; \boldsymbol{t}) = \nabla G(\boldsymbol{t})$. However, in the contrary case difficulties may arise depending upon how $G(\boldsymbol{x})$ is specified. Fortunately most problems are stated so that $G(\boldsymbol{x})$ takes the form

$$G(\boldsymbol{x}) = \max_i g_i(\boldsymbol{x})$$

for $\boldsymbol{x} \in S$, where $g_i(\boldsymbol{x})$ is differentiable, convex and defined on some set S_i^* containing S, and i runs over some index set.

Whenever $G(\boldsymbol{x})$ has the above form the components of $\nabla p(\boldsymbol{x}; \boldsymbol{t})$ may be computed as follows: If $\boldsymbol{t}_k \in S$ then $G(\boldsymbol{t}_k) = g_h(\boldsymbol{t}_k)$ for some value of h. At the point $\boldsymbol{x} = \boldsymbol{t}_k$ let

$$\frac{\partial p}{\partial x_j} = \frac{\partial g_h}{\partial x_j} \qquad 1 \leqq j \leqq n.$$

If $G(\boldsymbol{t}_k) = g_h(\boldsymbol{t}_k)$ for several h, it is indifferent to the convergence of the process which of these values of h is selected. Indeed, for these values of h

one may substitute any convex linear combination of the $\nabla g_h(t_k)$ for $\nabla p(x; t_k)$.

Assuming now that there is no essential difficulty in determining $\nabla p(x; t_k)$ we proceed to consider how to obtain a solution to (1). Rewriting (1) obtains

$$(3) \qquad \begin{cases} Ax \geqq b \\ -\nabla p(x; t_i)x \geqq G(t_i) - \nabla p(x; t_i)t_i \quad (0 \leqq i \leqq k-1) \\ cx = \text{minimum.} \end{cases}$$

It is clear that (3) is a linear program and can be solved by standard techniques. If solved in the form given, some variation of Lemke's method [8] is probably best; if the dual problem is solved instead, some method like the revised simplex method [2] is probably best, the solution to the primal problem being obtained as a by-product of the calculation. Since most presently programmed computational schemes are variants of the revised simplex method, the cutting plane method is most easily implemented by using the dual of (3). This dual program may be written as follows:

$$(4) \qquad \begin{gathered} uA - \textstyle\sum_{i=0}^{k-1} v_i \nabla p(x; t_i) = c \\ u \geqq 0, \qquad v_i \geqq 0 \qquad (0 \leqq i \leqq k-1) \\ ub + \textstyle\sum_{i=0}^{k-1} v_i[G(t_i) - \nabla p(x; t_i] = \text{maximum} \end{gathered}$$

where u is an m-dimensional row vector.

By adding a relatively few instructions to existing codes the transition from step k to step $k+1$, which involves the introduction of a new cutting plane, may be accomplished with a minimum of effort. This transition involves adding but one new variable in (4). When this is done the current (at the kth step) solution to (4) is generally not optimal. However, we may use this solution as a starting solution to obtain the optimal solution required (at the $(k+1)$th step). A subroutine for computing $\nabla p(x; t_k)$ and $G(t_k) - \nabla p(x; t_k)t_k$, and instructions for calling on them at the proper time are all that need be added to most existing computer codes.

How to select S so that the minimum of cx on the closed set R^* is contained in S is a difficult problem in general. Indeed, the problem may not have a finite minimum. However, it is not uncommon that the nature of a specific problem may indicate the impossibility of any coordinate of the solution vector x becoming numerically greater than a stated upper bound. Such is the case with most practical applications. When such a bound, λ, is known we may use the simple expedient of taking $S = \{x \mid |x_j| \leqq \lambda, 1 \leqq j \leqq n\}$, where x_j is the jth coordinate of x. If any x_j attains its upper or lower bound in the final solution, the formulation of the problem may be suspect.

From a computational viewpoint it is inefficient to carry along from the start all the constraints defining S. They should only be introduced as needed as the computation proceeds. A constraint is needed whenever (4) has an unbounded maximum for some value of k.

Although we made the proof of our theorem depend on it, it is not necessary to carry along all cutting planes through the whole computation. Suppose we plan to end the computation when $G(\boldsymbol{t}_k) \leqq \epsilon$. We will call $\boldsymbol{t}_k$ an ϵ-effective solution to the convex program. If k became at all large before an ϵ-effective solution were found it would unduly magnify the computation time. After a while many of the constraints become redundant and should not be carried along through the remainder of the computation.

There are a number of alternatives open, one of which is the following: Suppose that for some set of points $\boldsymbol{t}_0, \cdots, \boldsymbol{t}_{k-1}$ in S, $G(\boldsymbol{t}_k) = M$. If $M \leqq \epsilon$, then $\boldsymbol{t}_k$ is ϵ-effective and the computation stops. However, if $M > \epsilon$ we content ourselves with obtaining only some percentage improvement in the solution instead of proceeding by our theorem directly to an ϵ-effective solution. Thus we would follow the procedure of the theorem until we find a $k = \alpha$ such that $G(\boldsymbol{t}_\alpha) \leqq \max(\epsilon, \theta M)$, where $0 \leqq \theta \leqq 1$. By our theorem this point must eventually be reached.

At this point we eliminate all cutting planes that do not pass through the point $\boldsymbol{t}_\alpha$, we rename the $\boldsymbol{t}$ points of the remaining cutting planes, and start again to improve the current solution by some percentage. Clearly, this procedure eventually yields an ϵ-effective solution but should require handling fewer cutting planes at any one time. This is a distinct advantage when an electronic computer is used.

5. An example. In order to illustrate the computational procedure for the cutting plane method consider the following example: *Find a vector* $\boldsymbol{x} = (x_1, x_2)$ *such that* $f = x_1 - x_2$ *is a minimum subject to*

$$G(\boldsymbol{x}) = 3x_1^2 - 2x_1x_2 + x_2^2 - 1 \leqq 0.$$

It is easy to see that the boundary of the region over which we are to minimize f is an ellipse. Thus, the problem is a convex program. Further, the minimum takes place at the point $\boldsymbol{x} = (0, 1)$ and $\min f = -1$.

In terms of the notation of §2 we write

$$\begin{aligned} R &= \{\boldsymbol{x} \mid G(\boldsymbol{x}) \leqq 0\}, \\ S_0 &= \{(x_1, x_2) \mid -2 \leqq x_1, x_2 \leqq 2\}, \\ \boldsymbol{t}_k &= (t_1^{(k)}, t_2^{(k)}), \\ p(\boldsymbol{x}; \boldsymbol{t}_k) &= G(\boldsymbol{t}_k) - \nabla p(\boldsymbol{x}; \boldsymbol{t}_k)(x_1 - t_1^{(k)}, x_2 - t_2^{(k)}), \\ \nabla p(\boldsymbol{x}; \boldsymbol{t}_k) &= (6t_1^{(k)} - 2t_2^{(k)}, -t_1^{(k)} + 2t_2^{(k)}), \\ f_k &= t_1^{(k)} - t_2^{(k)}. \end{aligned}$$

What we have done here is imbed R in the square defined by S_0. Initially we solve the linear program

$$-2 \leqq x_1 \leqq 2$$

$$-2 \leqq x_2 \leqq 2$$

$$x_1 - x_2 = \text{minimum.}$$

The solution to this problem is found to be $\boldsymbol{x} = (-2, 2)$ with $f_0 = -4$. Therefore we set $\boldsymbol{t}_0 = (-2, 2)$. The new constraint required to eliminate $\boldsymbol{t}_0$ from further consideration is computed as indicated above. We obtain

$$p(\boldsymbol{x}; \boldsymbol{t}_0) = -16x_1 + 8x_2 - 25 \leqq 0.$$

S_1 becomes $S_0 \cap \{\boldsymbol{x} \mid p(\boldsymbol{x}; \boldsymbol{t}_0) \leqq 0\}$. We are now required to solve the linear program

$$-2 \leqq x_1 \leqq 2$$

$$-2 \leqq x_2 \leqq 2$$

$$-16x_1 + 8x_2 \leqq 25$$

$$x_1 - x_2 = \text{minimum.}$$

The solution to this linear program is found to be $\boldsymbol{x} = (-0.5625, 2)$ with $f_1 = -2.5625$. Therefore we set $\boldsymbol{t}_1 = (-0.5625, 2)$. We now continue as before. The results of several steps are tabulated below:

k	$p(\mathbf{x}; \mathbf{t}_k)$	$t_1^{(k)}$	$t_2^{(k)}$	f_k	$G(\mathbf{t}_k)$
0	$-16.00000x_1+8.00000x_2-25.00000$	−2.00000	2.00000	−4.00000	23.00000
1	$-7.37500x_1+5.12500x_2-\ 8.19922$	−0.56250	2.00000	−2.56250	6.19922
2	$-2.33157x_1+3.44386x_2-\ 4.11958$	0.27870	2.00000	−1.72193	2.11978
3	$-4.85341x_1+2.73459x_2-\ 3.43067$	−0.52970	0.83759	−1.36730	1.43067
4	$-2.63930x_1+2.42675x_2-\ 2.47792$	−0.05314	1.16024	−1.21338	0.47793
5	$-0.41071x_1+2.11690x_2-\ 2.48420$	0.42655	1.48499	−1.05845	0.48419
6	$-1.38975x_1+2.07205x_2-\ 2.13155$	0.17058	1.20660	−1.03603	0.13154
7	$-1.97223x_1+2.04538x_2-\ 2.04657$	0.01829	1.04098	−1.02269	0.04656
8	$-2.67809x_1+2.01305x_2-\ 2.06838$	−0.16626	0.84027	−1.00653	0.06838
9		−0.07348	0.92972	−1.00321	0.01723

REFERENCES

1. E. W. Cheney and A. A. Goldstein, *Newton's method of convex programming and Tchebycheff approximation*, Numer. Math., 1 (1959), pp. 253–268.
2. G. B. Dantzig, A. Orden and P. Wolfe, *The generalized simplex method for minimizing a linear form under linear inequality constraints*, Pacific J. Math., 5 (1955), pp. 283–295.
3. R. E. Gomory, *An algorithm for integer solutions to linear programs*, Princeton

IBM Mathematics Research Project, Technical Report No. 1, Princeton University, November 17, 1958.

4. ———, *Outline of an algorithm for integer solutions to linear programs*, Bull. Amer. Math. Soc., 64 (1958), pp. 275–278.
5. J. E. KELLEY, JR., *A computational approach to convex programming* (Abstract), Econometrica, 27 (1959), p. 276.
6. ———, *A general technique for convex programming* (Abstract), *The Rand Symposium on Mathematical Programming*, Rand Report No. R-351, The RAND Corporation, Santa Monica, Cal., 1960, p. 101.
7. ———, *An application of linear programming to curve fitting*, this Journal, 6 (1958), pp. 15–22.
8. C. E. LEMKE, *The dual method of solving the linear programming problem*, Naval Res. Logist. Quart., 1 (1954), pp. 36–47.
9. J. B. ROSEN, *The gradient projection method for nonlinear programming, Part I. Linear constraints*, this Journal, 8 (1960), pp. 181–217.
10. P. WOLFE, *Computational techniques for non-linear programs*, Princeton University Conference on Linear Programming, March 13–15 1957.
11. ——— (Editor), *The Rand Symposium on Mathematical Programming*, Rand Report No. R-351, The RAND Corporation, Santa Monica, Cal., 1960.