AD-A063 965 WISCONSIN UNIV-MADISON MATHEMATICS RESEARCH CENTER F/G 12/1
EXPLOITING STRUCTURE IN FIXED-POINT COMPUTATION.(U)
AUG 78 M J TODD DAAG29-75-C-0024
UNCLASSIFIED MRC-TSR-1867 NL

1 OF 1
ADA
063865

END
DATE
FILMED
3 79
DDC

(14) MRC-TSR-

(9) MRC Technical Summary Report #1867

(6) EXPLOITING STRUCTURE IN FIXED-POINT COMPUTATION.

(10) Michael J. Todd

LEVEL

(15) DAAG29-75-C-0024, NSF-ENG76-08749

Mathematics Research Center
University of Wisconsin—Madison
610 Walnut Street
Madison, Wisconsin 53706

(11) Aug[illegible]78

(12) 30 p.

(Received July 5, 1978)

Approved for public release
Distribution unlimited

Sponsored by

U. S. Army Research Office
P. O. Box 12211
Research Triangle Park
North Carolina 27709

and

National Science Foundation
Washington, D. C.
20550

UNIVERSITY OF WISCONSIN - MADISON
MATHEMATICS RESEARCH CENTER

EXPLOITING STRUCTURE IN FIXED-POINT COMPUTATION

Michael J. Todd

Technical Summary Report #1867

August 1978

ABSTRACT

We consider the recent algorithms for computing fixed points or zeros of continuous functions from R^n to itself that are based on tracing piecewise-linear paths in triangulations. We investigate the possible savings that arise when these fixed-point algorithms with their usual triangulations are applied to computing zeros of functions f with special structure: f is either linear in certain variables, separable, or has Jacobian with small bandwidth. Each of these structures leads to a property we call modularity; the algorithmic path within a simplex can be continued into an adjacent simplex without a function evaluation or linear programming pivot. Modularity also arises without any special structure on f from the linearity of the function that is deformed to f.

In the case that f is separable we show that the path generated by Kojima's algorithm with the homotopy H^2 coincides with the path generated by the standard restart algorithm of Merrill when the usual triangulations are employed. The extra function evaluations and linear programming steps required by the standard algorithm can be avoided by exploiting modularity.

AMS(MOS) Subject Classification: 90C99

Key Words: Solving systems of equations, fixed-point algorithms, triangulations, structure

Work Unit Number 5 - Mathematical Programming and Operations Research

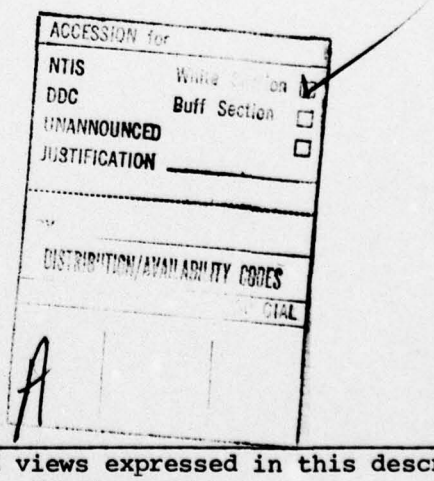
---

Sponsored by the United States Army under Contract No. DAAG29-75-C-0024 and by the National Science Foundation under Grant No. ENG76-08749.

SIGNIFICANCE AND EXPLANATION

Problems of solving systems of equations in several variables arise frequently in applied mathematics - in solving discretized versions of boundary value problems, in optimization and in economics. A class of algorithms for such problems has been developed in recent years based on tracing piecewise-linear paths in a triangulation of the domain of interest. These algorithms are generally called fixed-point algorithms because they guarantee convergence when seeking a fixed point of a continuous function from a compact convex set into itself. Besides strong global convergence properties, these algorithms possess quadratic convergence to a solution given sufficient smoothness conditions.

The main drawback of these algorithms is the large number of function evaluations required compared to other methods that may not be as robust. Here we alleviate this problem by showing how to exploit special structure in the function f whose zero is sought - either linearity in certain variables, separability, or having a Jacobian matrix whose nonzero entries are concentrated around the diagonal.



The responsibility for the wording and views expressed in this descriptive summary lies with MRC, and not with the author of this report.

EXPLOITING STRUCTURE IN FIXED-POINT COMPUTATION

Michael J. Todd

1. Introduction.

We consider fixed-point algorithms [2,3,4,5,10,15,18] for computing a zero of a continuous function $f : R^n \to R^n$. Such algorithms deform a simple function $f^0 : R^n \to R^n$ into f or into a piecewise-linear approximation to f and trace the zeroes of the resulting homotopy. Our aim is to see how such algorithms can exploit special structure in f and f^0.

While our approach is valid for other algorithms, in particular the homotopy method of Eaves [2] and Eaves and Saigal [4] and the algorithm of Garcia [5], we shall confine ourselves to the restart method of Merrill [10]. Here f^0 is an affine function with a unique zero at x^0, say. Consider the homotopy $h : R^n \times [0,1] \to R^n$ defined by $h(x,t) = tf(x) + (1-t)f^0(x)$. Let T be a special triangulation of $R^n \times [0,1]$; that is, all vertices of T lie in $R^n \times \{0,1\}$. Next let ℓ be a piecewise-linear approximation to h with respect to T. In other words, ℓ agrees with h on the vertices of T and is affine on each simplex of T.

Merrill's algorithm traces the piecewise-linear path of zeroes of ℓ starting with the known zero $(x^0, 0)$. Under certain conditions, this path ends at a point $(x^1, 1)$ and x^1 is thus a zero of a piecewise-linear approximation to f. For details on this path tracing, see Merrill [10] or Todd [18]. One can then restart the algorithm with a special triangulation T' of smaller mesh and a new function f^0 whose unique zero is x^1.

Suppose the algorithmic path meets an $(n+1)$-simplex $\sigma = \langle y^0,\ldots,y^{n+1}\rangle$ of T. Then within this simplex the path is linear. Assume that as the path is traced one encounters the face of σ opposite $\bar{y}^j$. The algorithm requires that we find the $(n+1)$-simplex σ' of T on the other side of this face, i.e. $\sigma' = \langle \bar{y}^0,\ldots,\bar{y}^{j-1}, \hat{y}^j, \bar{y}^{j+1},\ldots,\bar{y}^{n+1}\rangle \in T$, $\sigma' \neq \sigma$. We next compute $\ell(\hat{y}^j)$ and hence the algorithmic path within σ'. Our interest is in the case where ℓ is affine in $\sigma \cup \sigma'$; hence $\ell(\hat{y}^j)$ and the algorithmic path in σ' can be predicted from information known at σ - indeed the path in σ' is merely an extension of the

Sponsored by the United States Army under Contract No. DAAG29-75-C-0024 and by the National Science Foundation under Grant No. ENG76-08749.

straight line path in σ. In this case we say ℓ is modular at $\vec{y}$ with respect to σ.

In section 2 we describe two triangulations of $R^n \times [0,1]$ commonly used in fixed-point computation, for which modularity is easily exploited. In terms of these triangulations the definition of modularity takes an especially simple form reminiscent of the definition of modularity for functions on a lattice. In section 3 we show how modularity follows from three structures on f: linearity in certain variables, separability, or the property of having a Jacobian with small bandwidth. Section 4 describes how we exploit modularity in the restart algorithm.

Sections 5, 6 and 7 discuss further the special structures above. In section 5 we show how linearity arises naturally in the zero finding problem (or, more generally, complementarity problem) arising from a nonlinear programming problem. Section 6 contains the surprising result that Kojima's algorithm [8] for the separable case using his homotopy H^2 generates a path identical to that of the standard restart algorithm. The extra linear programming pivots and function evaluations required by the latter may all be avoided by exploiting modularity. Finally, section 7 discusses how function evaluations can be reduced if the Jacobian of f has small bandwidth - our approach here follows Curtis, Powell and Reid [1].

2. Triangulations.

For simplicity we confine our attention to triangulations with grid size 1. When used in fixed-point computation, these triangulations are generally scaled and translated in their first n coordinates. Identify $R^n \times [0,1]$ with $\{\bar{x} \in R^{n+1} | 0 \le \bar{x}_{n+1} \le 1\}$. Removal of the bar from a vector in $R^n \times [0,1]$ denotes its projection into R^n. Let $\bar{u}^1,\ldots,\bar{u}^{n+1}$ be the unit vectors in R^{n+1}, so that $u^1,\ldots,u^n$ are the unit vectors in R^n.

Now let $\bar{y} \in R^n \times \{0\}$ have y_i an integer for $1 \le i \le n$. Let π be a permutation of $\{1,2,\ldots,n+1\}$. Then $k_1(\bar{y},\pi)$ denotes the simplex $\langle \bar{y}^0,\ldots,\bar{y}^{n+1} \rangle$, where

$$\bar{y}^0 = \bar{y} \; ; \; \bar{y}^i = \bar{y}^{i-1} + \bar{u}^{\pi(i)}, \; 1 \le i \le n+1 \; . \tag{1}$$

The special triangulation $\tilde{K}_1$ is the set of all such $k_1(\bar{y},\pi)$'s [18, chapter III].

Next let $\bar{y} \in R^n \times \{0\}$ have y_i an odd integer for $1 \le i \le n$. Let π be a permutation of $\{1,2,\ldots,n+1\}$ and let $\bar{s} \in R^n \times \{1\}$ be a sign vector - $\bar{s}_i = \pm 1$ for $1 \le i \le n$. Then $j_1(\bar{y},\pi,\bar{s})$ denotes the simplex $\langle \bar{y}^0,\ldots,\bar{y}^{n+1} \rangle$, where

$$\bar{y}^0 = \bar{y} \; ; \; \bar{y}^i = \bar{y}^{i-1} + \bar{s}_{\pi(i)}\bar{u}^{\pi(i)}, \; 1 \le i \le n+1. \tag{2}$$

The special triangulation $\tilde{J}_1$ is the set of all such $j_1(\bar{y},\pi,\bar{s})$'s [18, chapter III].

The property that allows us to efficiently exploit structure in these triangulations is that each vertex of a simplex differs from its predecessor in just one coordinate. We always assume the vertices of a simplex in $\tilde{K}_1$ or $\tilde{J}_1$ are ordered as above to simplify the use of this property.

Let $\sigma = \langle \bar{y}^0,\ldots,\bar{y}^{n+1} \rangle \in T$, $T = \tilde{K}_1$ or $\tilde{J}_1$, and suppose that we wish to know $\hat{y}^j$ so that when $\hat{y}^j$ replaces $\bar{y}^j$ in σ, another simplex in T results. (We say $\hat{y}^j$ is the replacement for $\bar{y}^j$ in σ.) Consider first $\tilde{K}_1$. We have

$$\begin{aligned} j = 0 &: \hat{y}^0 = \bar{y}^{n+1} - \bar{y}^0 + \bar{y}^1, \; \sigma' = \langle \bar{y}^1,\ldots,\bar{y}^{n+1}, \hat{y}^0 \rangle \\ 0 < j < n+1 &: \hat{y}^j = \bar{y}^{j-1} - \bar{y}^j + \bar{y}^{j+1}, \; \sigma' = \langle \bar{y}^0,\ldots,\hat{y}^j,\ldots,\bar{y}^{n+1} \rangle \\ j = n+1 &: \hat{y}^{n+1} = \bar{y}^n - \bar{y}^{n+1} + \bar{y}^0, \; \sigma' = \langle \hat{y}^{n+1}, \bar{y}^0,\ldots,\bar{y}^n \rangle \; . \end{aligned} \tag{3}$$

In each case, we list the vertices on the right in the natural ordering given by (1).

Now consider $\tilde{J}_1$. We have

$$j = 0 : \hat{y}^0 = -\bar{y}^0 + 2\bar{y}^1$$

$$0 < j < n+1 : \hat{y}^j = \bar{y}^{j-1} - \bar{y}^j + \bar{y}^{j+1} \tag{4}$$

$$j = n+1 : \hat{y}^{n+1} = 2\bar{y}^n - \bar{y}^{n+1} \quad .$$

In each case, the natural ordering in (2) is unchanged - we have $\sigma' = \langle \bar{y}^0, \ldots, \hat{y}^j, \ldots, \bar{y}^{n+1} \rangle$.

For these triangulations, modularity takes an especially simple form. First let $T = \tilde{K}_1$. Suppose $0 < j < n+1$ and let $h = \pi(j)$, $i = \pi(j+1)$. Then $\bar{y}^j = \bar{y}^{j-1} + \bar{u}^h$, $\bar{y}^{j+1} = \bar{y}^{j-1} + \bar{u}^h + \bar{u}^i$, and $\hat{y}^j = \bar{y}^{j-1} + \bar{u}^i$. It is easy to see that ℓ is modular at $\bar{y}^j$ with respect to σ if $n+1 \notin \{h,i\}$ and either $\pi^{-1}(n+1) < j$ and

$$f(y^{j-1} + u^h) + f(y^{j-1} + u^i) = f(y^{j-1}) + f(y^{j-1} + u^h + u^i) \tag{5}$$

or $\pi^{-1}(n+1) > j$ and (5) holds with f^0 replacing f.

The similarity of (5) to the condition for modularity for a function defied on a lattice prompts our nomenclature. (The condition given above is almost necessary as well as sufficient for modularity. Any modularity with $j \in \{0,n+1\}$ or $n+1 \in \{h,i\}$ can only be due to a fortuitous relationship between f and f^0.)

Now suppose $T = \tilde{J}_1$. If $0 < j < n+1$, a sufficient condition for modularity is similar to that given above with $\bar{s}_h u^h$ and $\bar{s}_i u^i$ replacing u^h and u^i in (5). Now suppose $j = 0$ and let $i = \pi(1)$. If $i = n+1$ then $\bar{y}^0$ is the only vertex of σ in $R^n \times \{0\}$. Thus the end point $(x^1,1)$ has been found and no function evaluation or linear programming pivot is required. Hence suppose $i \neq n+1$. Then ℓ is modular at $\bar{y}^0$ in σ iff

$$f^0(y^0 - u^i) + f^0(y^0 + u^i) = 2f^0(y^0) \quad . \tag{6}$$

Finally, if $j = n+1$ let $h = \pi(n+1)$ - since y^{n+1} is to be removed, we can assume $h \neq n+1$. Then ℓ is modular at y^{n+1} in σ iff

$$f(y^{n+1} - u^h) + f(y^{n+1} + u^h) = 2f(y^{n+1}) \quad . \tag{7}$$

Henceforth, for simplicity in notation we deal only with the case that $T = \tilde{K}_1$. The corresponding results for $\tilde{J}_1$ are immediate. As we shall see, there are advantages to $\tilde{J}_1$ - in particular, since f^0 is affine, (6) always holds. From now on, we shall also abbreviate "ℓ is modular at $\overrightarrow{y^j}$ with respect to σ" by "$\overrightarrow{y^j}$ is modular in σ" with ℓ understood.

3. Examples of Modularity.

Here we show how modularity follows from three special structures on f. In addition, modularity arises naturally from the linearity of the artificial function f^0.

3.1. Linearity. Identify $R^p \times R^q$ with R^n, where $p+q = n$, and suppose f is such that $f(v,w)$ is affine in $w \in R^q$ for each fixed $v \in R^p$. Section 5 gives an application of such a problem to nonlinear programming. We then have

Lemma 3.1. Suppose $\sigma = \langle \bar{y}^0, \ldots, \bar{y}^{n+1} \rangle = k_1(\bar{y}, \pi)$, and f is as above. Then $\bar{y}^j$ is modular in σ if $0 < j < \pi^{-1}(n+1) - 1$ or if $\pi^{-1}(n+1) < j < n+1$ and both $h = \pi(j)$ and $i = \pi(j+1)$ are greater than p.

Proof. If $j < \pi^{-1}(n+1) - 1$ then $n+1 \notin \{h,i\}$ and $\pi^{-1}(n+1) > j$. Thus we only need to check that (5) holds with f^0 replacing f. But this follows from the linearity (in the affine sense) of f^0.

If $j > \pi^{-1}(n+1)$ then again $n+1 \notin \{h,i\}$ and we must verify (5). But the projections of y^{j-1}, $y^{j-1} + u^h$, $y^{j-1} + u^i$ and $y^{j-1} + u^h + u^i$ on their first p coordinates are the same. Hence (5) follows from the linearity of f in w for fixed v.

Note that (6) also holds if f^0 is affine, and (7) holds if $h > p$. Hence the use of $\tilde{J}_1$ allows additional exploitation of linearity.

3.2. Separability. The function f is called separable if there exist functions $f^i : R \to R^n$, $1 \le i \le n$, such that $f(x) = \sum_i f^i(x_i)$. Special algorithms to compute zeroes of such functions have been studied by Kojima [8]. One of these will be discussed in section 6.

Lemma 3.2. Suppose $\sigma = \langle \bar{y}^0, \ldots, \bar{y}^{n+1} \rangle = k_1(\bar{y}, \pi)$ and f is separable. Then $\bar{y}^j$ is modular in σ if $0 < j < \pi^{-1}(n+1) - 1$ or $\pi^{-1}(n+1) < j < n+1$.

Proof. For the first case the reasoning follows that in lemma 3.2. Suppose $\pi^{-1}(n+1) < j < n+1$ and let $h = \pi(j)$ and $i = \pi(j+1)$. We must check (5). Consider the equation

$$f^m((y^{j-1} + u^h)_m) + f^m((y^{j-1} + u^i)_m) = f^m(y_m^{j-1}) + f^m((y^{j-1} + u^h + u^i)_m) \quad .$$

For $m \notin \{h,i\}$ all four terms are equal and the equation is valid. For $m = h$, the first and

last terms are equal and so are the middle two - again the equation is valid. Similarly, it holds for $m = i$. Summing over m gives (5).

3.3. Functions with small bandwidth. Suppose the pth component function f_p of f depends on x_q only for $|p-q| < k$. Then we say f has bandwidth $2k-1$. If f is continuously differentiable and its Jacobian matrix has bandwidth m, then so does f.

Lemma 3.3. Suppose $\sigma = \langle \bar{y}^0, \ldots, \bar{y}^{n+1} \rangle = k_1(\bar{y}, \pi)$ and f has bandwidth m. Then $\bar{y}^j$ is modular in σ if $0 < j < \pi^{-1}(n+1) - 1$ or if $\pi^{-1}(n+1) < j < n+1$ and $|\pi(j) - \pi(j+1)| \geq m$.

Proof. We only need to verify (5) for $|h-i| \geq m$. For each $1 \leq p \leq n$, the pth component function f_p is affected by at most one of x_h and x_i. Hence (5) is valid for f_p since the terms are equal in pairs. Since p was arbitrary the lemma is proved.

Section 7 discusses further how the occurrence of modularity can be promoted and function evaluations circumvented when f has small bandwidth.

4. Exploiting Modularity.

Let $\sigma = \langle \bar{y}^0, \ldots, \bar{y}^{n+1} \rangle$ be a simplex of $\tilde{K}_1$ encountered by the algorithm, and for each j let $a^j \in R^{n+1}$ be the column vector $(1, \ell(\bar{y}^j))$. Then σ was generated because some face of σ, say that opposite $\bar{y}^k$, was known to contain a zero of ℓ. If B is the matrix $[a^0, \ldots, a^{k-1}, a^{k+1}, \ldots, a^{n+1}]$, then there is a solution $z \geq 0$ to $Bz = \bar{u}^{-1}$. (In fact, to circumvent the problems of degeneracy, we will only generate this face if B^{-1} has lexicographically nonnegative rows. For simplicity, however, we assume nondegeneracy throughout - we suppose B is nonsingular and the first column of its inverse is positive.)

We also have available to us the inverse B^{-1}. The analysis below assumes that B^{-1} is available explicitly. It is possible, however, to assume that we use the product form of the inverse and still exploit modularity.[1] A factored form of the inverse, such as $B = LU$, is not suitable for the use of our ideas.

The standard restart algorithm then proceeds as follows. Let $\bar{b} = B^{-1}\bar{u}^{-1}$, $\bar{a} = B^{-1}a^k$ and find j such that $\bar{a}_j > 0$ and $\bar{b}_j/\bar{a}_j = \min\{\bar{b}_r/\bar{a}_r \mid \bar{a}_r > 0\}$. Here and throughout this section, $\bar{b}$ and $\bar{a}$ have coordinates indexed $0, \ldots, k-1, k+1, \ldots, n+1$, to correspond to the indices of the columns of B. Then the other face of σ containing a zero of ℓ is the face opposite $\bar{y}^j$. Make a linear programming pivot step to obtain $\tilde{B}^{-1}$ where $\tilde{B}$ is the basis obtained from B by replacing a^j with a^k. Then find the replacement $\hat{y}^j$ for $\bar{y}^j$ in σ, compute $\ell(\hat{y}^j)$ and introduce $\hat{a}^j = (1, \ell(\hat{y}^j))$ into the basis $\hat{B}$, continuing the process.

Now suppose that $\bar{y}^j$ is modular in σ. Then we can save a function evaluation since $\hat{a}^j = a^{j-1} - a^j + a^{j+1}$. However we can do more - we need not make the pivot step to obtain $\tilde{B}^{-1}$.

Indeed, suppose first that $j \notin \{k-1, k+1\}$. Let $\hat{B} = [a^0, \ldots, \hat{a}^j, \ldots, a^{k-1}, a^{k+1}, \ldots, a^{n+1}]$ be the basis obtained from B by replacing a^j with $\hat{a}^j$. Then the inverse of $\hat{B}$ can be obtained from B^{-1} as follows. The $(j+\varepsilon)$-th row of $\hat{B}^{-1}$ is the $(j+\varepsilon)$th row of B^{-1} plus the jth row of B^{-1}, $\varepsilon = \pm 1$. The jth row of $\hat{B}^{-1}$ is the negative of the jth row of B^{-1}. All these relationships follow trivially from the equation $\hat{a}^j = a^{j-1} - a^j + a^{j+1}$. Similar changes are made to $\bar{a}$ and $\bar{b}$. The total work involved is $2n + 6$ additions and $n+3$ sign changes.

---

[1] I am grateful to Professor Romesh Saigal for this suggestion.

Note that the first column of $\hat{B}^{-1}$ is not positive - its jth entry is negative. However, since $\bar{a}_j$ was positive, it is now negative; if we express $\bar{u}^1$ in terms of $\hat{B}$ and a^k, and increase the weight on a^k, the weight on $\hat{a}^j$ increases to zero and beyond. We may therefore simulate the entry of the column $\hat{a}^j$ into the basis $\tilde{B}$ by entering a^k into the basis $\hat{B}$.

We therefore recompute the minimum ratio $\min\{\bar{b}_r/\bar{a}_r/\bar{a}_r > 0\}$. Notice that only two of these quotients (corresponding to $r = j-1$ and $r = j+1$) have changed. Suppose the minimum ratio occurs for the index j'. If $\bar{y}^{j'}$ is not modular in the new simplex $\sigma' = \langle \bar{y}^0,\ldots,\hat{y}^j,\ldots,\bar{y}^{n+1}\rangle$ we perform a normal pivot step, replacing $a^{j'}$ with a^k. If, however, $\bar{y}^{j'}$ is modular in σ' and $j' \notin \{k-1, k+1\}$ we may perform the same trick again. Hence several pivot steps may be saved (or rather, replaced by trivial pivot steps requiring minimal arithmetic) if several modular vertices are encountered.

We must now deal with the case where $j \in \{k-1, k+1\}$.[1] Suppose $j = k-1$. Then $B^{-1}\hat{a}^j = B^{-1}a^{k-2} - B^{-1}a^{k-1} + B^{-1}a^k = e^{k-2} - e^{k-1} + \bar{a}$, where e^i denotes a unit vector with coordinates indexed $0,\ldots,k-1, k+1,\ldots,n+1$ and ith coordinate equal to one. We now want to express $\bar{u}^1$ in terms of $a^0,\ldots,a^{k-2}, \hat{a}^{k-1}, a^k,\ldots,a^{n+1}$, *or equivalently,* $\bar{b}$ in terms of $e^0,\ldots,e^{k-2}, \bar{a} + e^{k-2} - e^{k-1}, \bar{a}, e^{k+1},\ldots,e^{n+1}$. Suppose the appropriate weights are $\lambda_0,\ldots,\lambda_{n+1}$. Then we have

$$\begin{aligned}
&\bar{a}_i(\lambda_{k-1} + \lambda_k) + \lambda_i = \bar{b}_i \ , \ i \notin \{k-2, k-1\} \ ; \\
&\bar{a}_{k-2}(\lambda_{k-1} + \lambda_k) + \lambda_{k-1} + \lambda_{k-2} = \bar{b}_{k-2} \ ; \qquad (8) \\
&\bar{a}_{k-1}(\lambda_{k-1} + \lambda_k) - \lambda_{k-1} = \bar{b}_{k-1} \ .
\end{aligned}$$

We know a nonnegative solution with $\lambda_{k-1} = 0 (\lambda_k = \bar{b}_{k-1}/\bar{a}_{k-1})$ and we want to increase λ_{k-1}. Notice that $\bar{a}_{k-1}$ is positive, since $j = k-1$ was chosen in the minimum ratio test. Hence $\lambda \equiv \lambda_{k-1} + \lambda_k$ increases with λ_{k-1}, from the last equation. Rewriting (8) we obtain

$$\begin{aligned}
&\bar{a}_i\lambda + \lambda_i = \bar{b}_i \ , \ i \notin \{k-2, k-1\} \ ; \\
&(\bar{a}_{k-2} + \bar{a}_{k-1})\lambda + \lambda_{k-2} = (\bar{b}_{k-2} + \bar{b}_{k-1}) \ ; \qquad (9) \\
&(\bar{a}_{k-1} - 1)\lambda + \lambda_k = \bar{b}_{k-1} \ .
\end{aligned}$$

[1] The rest of the section is very technical. The reader may skip to section 5 without loss of continuity.

We now use (9) to perform a minimum ratio test to determine which vertex next leaves the current simplex σ'. If this leaving vertex is $\bar{y}^{k-2}$ we find the band of non-unit columns in the representation of the current basis in terms of B widening. Let us therefore progress to the general case, where $\bar{y}^{k+1},\ldots,\bar{y}^{s}$ have been replaced (in that order) by $\hat{y}^{k+1},\ldots,\hat{y}^{s}$, and $\bar{y}^{k-1}, \bar{y}^{k-2},\ldots,\bar{y}^{r}$ have been replaced (in that order) by $\hat{y}^{k-1}, \hat{y}^{k-2},\ldots,\hat{y}^{r}$. Of course, when any such $\bar{y}^{j}$ is replaced, we assume that it was modular in the current simplex, or the process terminates.

We therefore want to express $\bar{b}$ in terms of the columns of

$$[e^{0},\ldots,e^{r-1}, \bar{a}+e^{r-1} - e^{k-1},\ldots,\bar{a}+e^{k-2} - e^{k-1}, \bar{a}, \bar{a}+e^{k+2} - e^{k+1},\ldots$$
$$\ldots,\bar{a}+e^{s+1} - e^{k+1}, e^{s+1},\ldots,e^{n+1}] \quad . \tag{10}$$

If the corresponding weights are $\lambda_0,\ldots,\lambda_{n+1}$, we have

$$\begin{aligned}
&\bar{a}_i(\lambda_r+\cdots+\lambda_s) + \lambda_i = \bar{b}_i, \ i \notin \{r-1,\ldots,s+1\} \ ; \\
&\bar{a}_{r-1}(\lambda_r+\cdots+\lambda_s) + \lambda_{r-1} + \lambda_r = \bar{b}_{r-1} \ ; \\
&\bar{a}_i(\lambda_r+\cdots+\lambda_s) + \lambda_{i+1} = \bar{b}_i, \ r \le i < k-1 \ ; \\
&(\bar{a}_{k-1} - 1)(\lambda_r+\cdots+\lambda_{k-1}) + \bar{a}_{k-1}(\lambda_k+\cdots+\lambda_s) = \bar{b}_{k-1} \ ; \\
&\bar{a}_{k+1}(\lambda_r+\cdots+\lambda_k) + (\bar{a}_{k+1} - 1)(\lambda_{k+1}+\cdots+\lambda_s) = \bar{b}_{k+1} \ ; \\
&\bar{a}_i(\lambda_r+\cdots+\lambda_s) + \lambda_{i-1} = \bar{b}_i, \ k+1 < i \le s \ ; \\
&\bar{a}_{s+1}(\lambda_r+\cdots+\lambda_s) + \lambda_s + \lambda_{s+1} = \bar{b}_{s+1} \ .
\end{aligned} \tag{11}$$

Suppose $\hat{y}^{r}$ is the new vertex and the weight λ_r is to be increased. Adding equations indexed r through k-1, we get $(\bar{a}_r+\cdots+\bar{a}_{k-1})(\lambda_r+\cdots+\lambda_s) - \lambda_r = (\bar{b}_r+ \ +\bar{b}_{k-1})$. As we shall see, the fact that $\bar{y}^{r}$ left the simplex implies that $(\bar{a}_r+\cdots+\bar{a}_{k-1})$ is positive. Hence we can increase $\lambda = \lambda_r+\cdots+\lambda_s$ instead of λ_r. Adding the equations indexed k+1 through s we obtain

$$(\bar{a}_{k+1}+\cdots+\bar{a}_s)(\lambda_r+\cdots+\lambda_s) - \lambda_s = (\bar{b}_{k+1}+\cdots+\bar{b}_s) \quad .$$

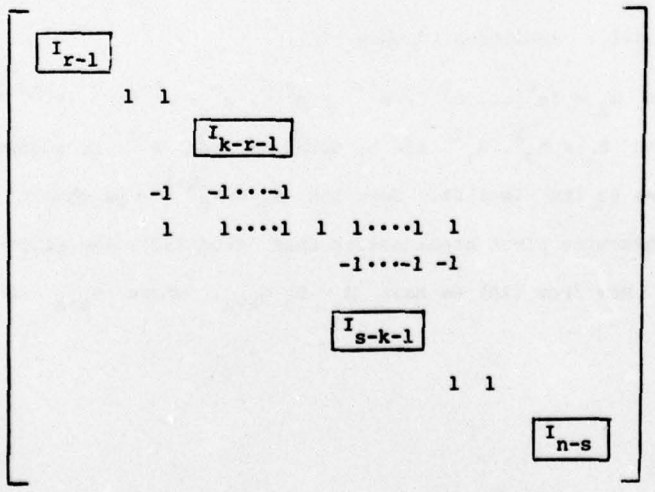
Now (11) can be written as

$$
\begin{aligned}
&\bar{a}_i\lambda + \lambda_i = \bar{b}_i \ , \ i \notin \{r-1,\ldots,s+1\} \ ; \\
&(\bar{a}_{r-1}+\cdots+\bar{a}_{k-1})\lambda + \lambda_{r-1} = (\bar{b}_{r-1}+\cdots+\bar{b}_{k-1}) \ ; \\
&\bar{a}_i\lambda + \lambda_{i+1} = \bar{b}_i \quad r \le i < k-1 \ ; \\
&(\bar{a}_{k-1} + \bar{a}_{k+1} - 1)\lambda + \lambda_k = \bar{b}_{k-1} + \bar{b}_{k+1} \ ; \qquad (12) \\
&\bar{a}_i\lambda + \lambda_{i-1} = \bar{b}_i \ , \ k+1 < i \le s \ ; \\
&-(\bar{a}_{k+1}+\cdots+\bar{a}_s)\lambda + \lambda_s = -(\bar{b}_{k+1}+\cdots+\bar{b}_s) \ ; \\
&(\bar{a}_{k+1}+\cdots+\bar{a}_{s+1})\lambda + \lambda_{s+1} = (\bar{b}_{k+1}+\cdots+\bar{b}_{s+1}) \ .
\end{aligned}
$$

We now use (12) to perform a minimum ratio test to see which $\bar{y}^j$ or $\hat{y}^j$ leaves the current simplex σ'. There are several cases.

Case 1. $\bar{y}^j$ or $\hat{y}^j$ is not modular in σ'. We wish to obtain $\tilde{B}^{-1}$, where $\tilde{B}$ is $\tilde{A} = [a^0,\ldots,a^{r-1}, \hat{a}^r,\ldots,\hat{a}^{k-1}, a^k, \hat{a}^{k+1},\ldots,\hat{a}^s, a^{s+1},\ldots,a^{n+1}]$ with the column indexed j removed. We write $\tilde{B} = \tilde{A}^{\sim j}$. The columns of matrices of size $(n+1) \times (n+2)$ will always be indexed $0,\ldots,n+1$; naturally the columns of $\tilde{B}$ are indexed $0,\ldots,j-1, j+1,\ldots,n+1$.

Case 1(a). $j \notin \{r-1,\ldots,s+1\}$. First do a standard linear programming pivot step to obtain B_1^{-1}, where $A_1 = [a^0,\ldots,a^{n+1}]$ and $B_1 = A_1^{\sim j}$. Now note from (10) that $\tilde{A} = A_1 P_{krs}$, where P_{krs} is the matrix



Hence $\tilde{B} = B_1 P^{\sim jj}_{krs}$, where $P^{\sim jj}_{krs}$ is P_{krs} with row and column j removed. It follows that $\tilde{B}^{-1} = (P^{\sim jj}_{krs})^{-1} B_1^{-1}$, and it only remains to invert $P^{\sim jj}_{krs}$. But this is easily done. Let Q_{krs} be the inverse of P_{krs}; we have Q_{krs} explicitly as

$$\begin{bmatrix} I_{r-1} & & & & & \\ & 1\ 1\cdots1\ 1 & & & & \\ & -1\cdots-1\ -1 & & & & \\ & & I_{k-r-1} & & & \\ & & & 1\ 1\ 1 & & \\ & & & & I_{s-k-1} & \\ & & & & -1\ -1\cdots-1 & \\ & & & & 1\ 1\cdots1\ 1 & \\ & & & & & I_{n-s} \end{bmatrix}$$

Then $(P^{\sim jj}_{krs})^{-1} = Q^{\sim jj}_{krs}$ and thus $\tilde{B}^{-1} = Q^{\sim jj}_{krs} B_1^{-1}$. Obtaining $\tilde{B}^{-1}$ from B_1^{-1} requires only $(s-r+4)(n+1)$ additions/subtractions.

Case 1(b). $j \in \{r+1,\ldots,k-1\}$. First do a standard linear programming pivot step to obtain B_1^{-1}, where $B_1 = A_1^{\sim j-1}$. Then an analysis identical to that above shows that $\tilde{B}^{-1} = Q^{\sim j,j-1}_{krs} B_1^{-1}$, where $Q^{\sim j,j-1}_{krs}$ is Q_{krs} with row j and column $j-1$ removed.

Case 1(c). $j \in \{k+1,\ldots,s-1\}$: analogous to case 1(b).

Case 1(d). $j = r-1$. Let $A_2 = [a^0,\ldots,a^{r-2}, a^{r-1} - a^{k-1}, a^r - a^{k-1},\ldots,a^{k-2} - a^{k-1}, a^{k-1}, a^k,\ldots,a^{n+1}]$. Then, with $B_1 = A_2^{\sim k}$, B_1^{-1} can be obtained from B^{-1} by adding the $(r-1)$st through $(k-2)$nd rows to the $(k-1)$st. Next let $B_2 = A_2^{\sim k-1}$ - we obtain B_2^{-1} from B_1^{-1} by a standard linear programming pivot step; notice that, from (12), the pivot element $(\bar{a}_{r-1}+\cdots+\bar{a}_{k-1})$ is positive. Now from (10) we have $\tilde{B} = B_2 D_{krs}$, where D_{krs} is the matrix

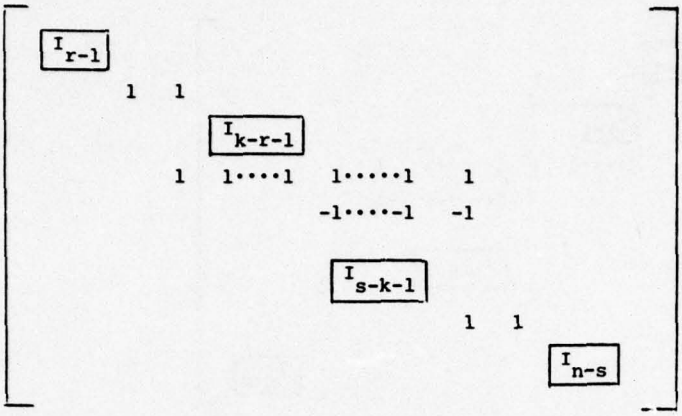
$$\begin{bmatrix} I_{r-1} & & & & \\ & I_{k-r} & & & \\ & 1\cdots1 \;\; 1 & 1\cdots\cdots1 \;\; 1 & & \\ & & -1\cdots\cdots-1 \;\; -1 & & \\ & & & I_{s-k-1} & \\ & & & & 1 \;\; 1 \\ & & & & & I_{n-s} \end{bmatrix}$$

with inverse

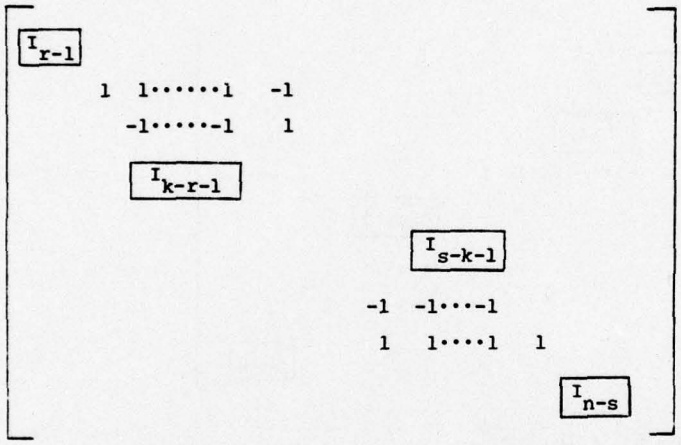
$$\begin{bmatrix} I_{r-1} & & & & \\ & I_{k-r} & & & \\ & -1\cdots-1 \;\; 1 \;\; 1 & & & \\ & & & I_{s-k-1} & \\ & & -1 \;\; -1\cdots-1 & & \\ & & 1 \;\; 1\cdots1 \;\; 1 & & \\ & & & & I_{n-s} \end{bmatrix}$$

Hence we easily obtain $\tilde{B}^{-1} = D_{krs}^{-1} B_2^{-1}$.

Case 1(e). $j = k$. Let $A_2 = [a^0,\ldots,a^{k-1}, a^k - a^{k-1}, a^{k+1} - a^{k-1}, a^{k+2},\ldots,a^{n+1}]$. Then, with $B_1 = \tilde{A}_2^k$, B_1^{-1} is obtained from B^{-1} by adding the $(k+1)$st row to the $(k-1)$st. Next, let $B_2 = \tilde{A}_2^{k-1}$ - since $\bar{a}_{k-1} + \bar{a}_{k+1} - 1$ is positive from (12), this is a standard linear programming pivot step. Now $\tilde{B} = B_2 E_{krs}$, where E_{krs} is the matrix



with inverse



Thus $\tilde{B}^{-1} = E_{krs}^{-1} B_2^{-1}$ is easily obtained.

Case 1(f). $j = s$. Let $A_2 = [a^0,\ldots,a^{k-1}, a^k, a^{k+1}, a^{k+2} - a^{k+1},\ldots,a^s - a^{k+1}, a^{s+1},\ldots, a^{n+1}]$. With $B_1 = A_2^{\tilde{k}}$, B_1^{-1} is obtained by adding the (k+2)nd through sth rows of B^{-1} to its (k+1)st row. Then a standard linear programming pivot step gives us $(A_2^{\tilde{k+1}})^{-1}$. From this it is easy to obtain $\tilde{B}^{-1}$; the argument is analogous to case 1(d).

Case 1(g). $j = s+1$. Analogous to that above; we start now by adding the (k+2)nd through (s+1)st rows of B^{-1} to its (k+1)st row.

We have finally completed the analysis of the nonmodular case. Fortunately, the case remaining is much simpler.

Case 2. $\bar{y}^j$ or $\hat{y}^j$ is modular in the current simplex σ'.

Case 2(a). $j \notin \{r-1,\ldots,s+1\}$. In the original B, replace a^j by $\hat{a}^j = a^{j-1} - a^j + a^{j+1}$; the update of B^{-1} is trivial. Correspondingly update $\bar{b}$ and $\bar{a}$ and return to the minimum ratio test of (12) - note that only two ratios have changed.

Case 2(b). $j = r-1$. Note that (12) then implies that $(\bar{a}_{r-1}+\cdots+\bar{a}_{k-1})$ is positive, thus justifying our claim above (12) that $(\bar{a}_r+\cdots+\bar{a}_{k-1})$ was positive at the previous step. In this case, r is decreased by 1 and we return to the minimum ratio test of (12). Only two ratios (for $r-2$ and r) need be calculated.

Case 2(c). $j = s$. Then s is decreased by 1 and we return to (12) - only the ratios for $s-1$ and $s+1$ are updated.

Case 2(d). $j = s+1$. We increase s by 1 and return to (12). The ratios for s and $s+2$ are recomputed.

Case 2(e). $j \in \{r+1,\ldots,s-1\}$. The new column, assuming $j < k$, is $\hat{\hat{a}}^j = \hat{a}^{j-1} + \hat{a}^{j+1} - \hat{a}^j$. Expressed in terms of the basis B, it is $B(\bar{a} + e^{j-2} - e^{j-1} + e^j - e^{k-1})$. It might be possible to handle this case and derive the appropriate counterpart to (12) to make the minimum ratio test. On the other hand, the complications of case 1 suggest that we avoid these difficulties. Certainly if case 2(e) occurred several times in succession, the resulting basis would be very hard to deal with. Hence we suggest ignoring the modularity and returning to the appropriate subcase of case 1. At least one pivot has already been saved, and we know $\hat{\hat{a}}^j$.

We now give an estimate of the amount of work that is saved by exploiting modularity. We suppose that the difficulties of case 2(e) have been surmounted. Our estimate is based on the measure of directional density of a triangulation, studied in [17, 19]. Given $x, d \in R^n$, let $\phi(t) = (x+td, t) \in R^n \times [0,1]$. Let T be a triangulation of $R^n \times [0,1]$ and $\sigma_1, \sigma_2 \in T$. We write $\sigma_1 \overset{t}{i} \sigma_2$ if $\phi(t-\eta) \in \sigma_1$, $\phi(t+\eta) \in \sigma_2$ for small positive η, and if the vertex of σ_2 not in σ_1 lies in $R^n \times \{i\}$. Let $n_i = i + |\{t \in (0,1) \mid \sigma_1 \overset{t}{i} \sigma_2 \text{ for some } \sigma_1, \sigma_2 \in T\}|$. Then $n_0(n_1)$ measures the number of function evaluations of $f^0(f)$ and the associated number of linear programming pivot steps. Let $N_i(T,d)$ be the average of n_i when x is uniformly distributed in a cube of side 2, for $T = \tilde{K}_1$ and $\tilde{J}_1$. We then have

Theorem 4.1 [19]

(a) $N_0(\tilde{K}_1,d) = \sum_i (d_i^- + (d_i-1)^+) + \sum_{i<j} \frac{1}{2}|d_i-d_j|$

(b) $N_1(\tilde{K}_1,d) = 1 + \sum_i (d_i^+ + (d_i-1)^-) + \sum_{i<j} \frac{1}{2}|d_i-d_j|$

(c) $N_0(\tilde{J}_1,d) = \sum_i \frac{1}{2}(|d_i| + (d_i-1)^+ + (d_i+1)^-) + \sum_{i<j} \frac{1}{4}(|d_i+d_j| + |d_i-d_j|)$

(d) $N_1(\tilde{J}_1,d) = 1 + \sum_i \frac{1}{2}(|d_i| + (d_i-1)^- + (d_i+1)^+) + \sum_{i<j} \frac{1}{4}(|d_i+d_j| + |d_i-d_j|)$.

Here, for any real λ, $\lambda^+ = \max\{0,\lambda\}$ and $\lambda^- = \lambda^+ - \lambda$. Using the same arguments we can prove the result below. Of course, since $\phi(t)$ is straight we may call all vertices modular. However, when we say that so many pivots can be avoided by exploiting modularity, we mean only those which are guaranteed to be modular by the linearity of f^0 or the special structure of f.

Theorem 4.2

(i) By using the linearity of f^0, we can avoid $\sum_{i<j} \frac{1}{2}|d_i-d_j|$ of the pivots counted in (a) above, and $\sum_i \frac{1}{2}|d_i| + \sum_{i<j} \frac{1}{4}(|d_i+d_j| + |d_i-d_j|)$ of the pivots counted in (c) above.

(ii) If f is linear in variables $p+1,\ldots,n$, then we can avoid $\sum_{p<i<j} \frac{1}{2}|d_i-d_j|$ of the pivots counted in (b) above, and $\sum_{p<i} \frac{1}{2}|d_i| + \sum_{p<i<j} \frac{1}{4}(|d_i+d_j| + |d_i-d_j|)$ of the pivots counted in (d) above.

(iii) If f is separable, we can avoid in (b) and (d) the pivots counted in the sums $\sum_{i<j}$.

(iv) If f has bandwidth m, we can avoid in (b) and (d) that part of the pivots counted in the sums $\sum_{i<j}$ for which $|i-j| \geq m$.

5. An example of linearity.

Consider the nonlinear programming problem of minimizing $\theta(v)$ subject to $g(v) = 0$, where $\theta: R^p \to R$ and $g: R^p \to R^q$ are continuously differentiable. The problem of finding a stationary point of the Lagrangian of this program is that of finding a zero of f, where $f(v,w) = (\nabla\theta(v) + \nabla g(v)w, g(v))$. Here $w \in R^q$ - clearly for any fixed v, f is linear in w.

The inequality problem $(g(v) \leq 0)$ has traditionally been formulated for application of fixed-point algorithms as a zero-finding problem for a point-to-set mapping defined on R^p. The algorithms converge slowly in this case. If g is affine then the methods of Kojima [7] and Todd [20] provide fast algorithms.

We are here concerned with the case where g is not affine. For simplicity we consider only the equality-constrained problem above, but our analysis is easily extended to inequality or mixed constraints. Two papers have addressed this problem: Kojima [9] and Todd [20]. Both propose methods that use triangulations of $R^p \times R^q \times [0,1]$ (explicitly or implicitly) in which the mesh size for the w variables is large or infinite and does not shrink as the iterations progress. The reason, of course, is that a piecewise-linear approximation to f with respect to such a triangulation can be made arbitrarily close to f.

Fixed-point algorithms, besides yielding approximate fixed points or zeros, also provide approximations to the Jacobian of f [16, 13]. The exploitation of this information is the key to obtaining quadratic or faster convergence - see Saigal [12] and Saigal and Todd [14]. Unfortunately, the algorithms proposed in [9, 20] do not give good approximations to the partial derivatives of f with respect to the v-variables. Indeed, the approximation to $\partial f(v,w)/\partial v_j$ is the difference between the function values at the two vertices of the final simplex that differ by $\bar{u}^j$. If $\hat{w}$ is the value of w for these vertices, what is obtained is an approximation to $\partial f(v,\hat{w})/\partial v_j$, and $\hat{w}$ may be far from the approximate value of w.

We may obtain good approximations to the Jacobian of f by using a fine grid size for the w variables as well as the v variables. The disadvantage using the standard restart method is the large number of pivots to move from one region of w-space to another. In effect, one has a general $(p+q)$-dimensional problem. It is generally felt that the gain in smoothness compared to the standard p-dimensional formulation is not worth the corresponding increase in dimension.

However, the analysis of section 4 shows that a fine grid size can be employed for the w variables and most of the work of moving in the w-space can be eliminated by exploiting modularity. The advantage of smoothness may now compensate for the increase in dimension. In any case, the results of [12, 14] show that asymptotically quadratic convergence can be achieved under reasonable conditions; because of linearity in w, asymptotically each iterate requires only $p+1$ evaluations of $\nabla\theta$ and ∇g.

6. Separable functions.

Recall that $f:R^n \to R^n$ is separable if there exist $f^i:R \to R^n$, $1 \le i \le n$, such that $f(x) = \sum_i f^i(x_i)$. Clearly any linear function such as f^0 is separable. Kojima [8] has suggested two algorithms for computing a zero of such an f - here we restrict our attention to that using the homotopy H^2.

Using f, f^0 and the triangulation $\tilde{K}_1$ of $R^n \times [0,1]$ we construct the piecewise-linear function ℓ as in the introduction. On the other hand, for each i, $1 \le i \le n$, we can use f^i, f^{0i} and the triangulation $\tilde{K}_1$ of $R^1 \times [0,1]$ to construct $\ell^i(x_i,t)$. Define $\hat{\ell}(x,t) = \sum_i \ell^i(x_i,t)$.

Lemma 6.1. $\hat{\ell} = \ell$.

Proof. Suppose $\ell(x,t) = a \in R^n$. Then there are $\lambda_0,\ldots,\lambda_{n+1}$ and a simplex $\sigma = \langle \bar{y}^0,\ldots,\bar{y}^{n+1} \rangle \in \tilde{K}_1$ such that $(x,t) = \sum_{j=0}^{n+1} \lambda_j \bar{y}^j$ and

$$\sum_{j=0}^{n+1} \lambda_j f^j = a$$

$$\sum_{j=0}^{n+1} \lambda_j = 1 \qquad (13)$$

$$\lambda_j \ge 0,\ j = 0,\ldots,n+1$$

where $f^j = f(y^j)$ if $\bar{y}^j_{n+1} = 1$ and $f^j = f^0(y^j)$ otherwise. Suppose $\sigma = k_1(\bar{y},\pi)$. Let $I_1 = \{i \mid 1 \le i \le n,\ \pi^{-1}(i) < \pi^{-1}(n+1)\}$ and $I_2 = \{1,\ldots,n\} \setminus I_1$. For $i \in I_1$, define $z_i^0 = \bar{y}_i^0$, $z_i^1 = z_i^2 = \bar{y}_i^0 + 1$, $t_i^0 = t_i^1 = 0$ and $t_i^2 = 1$. Also define $\lambda_{i0} = \sum\{\lambda_j \mid 0 \le j < \pi^{-1}(i)\}$, $\lambda_{i1} = \sum\{\lambda_j \mid \pi^{-1}(i) \le j < \pi^{-1}(n+1)\}$ and $\lambda_{i2} = \sum\{\lambda_j \mid \pi^{-1}(n+1) \le j \le n+1\}$. For $i \in I_2$, define $z_i^0 = z_i^1 = \bar{y}_i^0$, $z_i^2 = \bar{y}_i^0 + 1$, $t_i^0 = 0$ and $t_i^1 = t_i^2 = 1$. Also define $\lambda_{i0} = \sum\{\lambda_j \mid 0 \le j < \pi^{-1}(n+1)\}$, $\lambda_{i1} = \sum\{\lambda_j \mid \pi^{-1}(n+1) \le j < \pi^{-1}(i)\}$ and $\lambda_{i2} = \sum\{\lambda_j \mid \pi^{-1}(i) \le j \le n+1\}$.

Consider the ith component of $(x,t) = \sum_{j=0}^{n+1} \lambda_j \bar{y}^j$. Because $\bar{y}_i^j = \bar{y}_i^0$ for $j < \pi^{-1}(i)$, $\bar{y}_i^j = \bar{y}_i^0 + 1$ for $j \ge \pi^{-1}(i)$, we obtain $x_i = \lambda_{i0} z_i^0 + \lambda_{i1} z_i^1 + \lambda_{i2} z_i^2$. Similarly $t = \lambda_{i0} t_i^0 + \lambda_{i1} t_i^1 + \lambda_{i2} t_i^2$. Now let $f^{ij} = f^i(\bar{y}_i^j)$ if $\bar{y}_{n+1}^j = 1$ and $f^{ij} = f^{0i}(\bar{y}_i^j)$

otherwise. Then (13) gives $\sum_{j=0}^{n+1} \lambda_j \sum_{i=1}^{n} f^{ij} = a$, and collecting terms yields $\sum_{i=1}^{n}(\lambda_{i0}g^{i0} + \lambda_{i1}g^{i1} + \lambda_{i2}g^{i2}) = a$, where $g^{i0} = f^{0i}(z_i^0)$, $g^{i2} = f^{i}(z_i^2)$ and $g^{i1} = f^{i}(z_i^1)$ if $t_i^1 = 1$, $g^{i1} = f^{0i}(z_i^1)$ otherwise. Together with the trivial facts that $\sum_{k=0}^{2} \lambda_{ik} = 1$, $\lambda_{ik} \geq 0$, this last equation says precisely that $\hat{\ell}(x,t) = a$.

The lemma gives immediately

Theorem 6.2. The path generated by Kojima's algorithm using homotopy H^2 and triangulation $\tilde{K}_1(\tilde{J}_1)$ for each component coincides with the path generated by Merrill's algorithm using triangulation $\tilde{K}_1(\tilde{J}_1)$.

Of course, many more pivots may be required in the latter algorithm. Indeed, Kojima's algorithm works with the linear system

$$\begin{aligned} \textstyle\sum_{i=1}^{n} \sum_{k=0}^{2} \lambda_{ik} g^{ik} &= 0 \\ \textstyle\sum_{k=0}^{2} \lambda_{ik} t_i^k &= t \\ \textstyle\sum_{k=0}^{2} \lambda_{ik} &= 1 \\ \lambda_{ik} &\geq 0 \ . \end{aligned} \tag{14}$$

During a single pivot in (14), it is possible that several λ_{i0}'s for $i \in I_1 \equiv \{i' | \lambda_{i0} < 1 - t\}$ change their relative magnitudes. From the correspondence between λ_j's and λ_{ik}'s in lemma 6.1, it can be seen that each such change requires a change in the permutation π and hence a pivot in (13).

However, the path of either algorithm is straight in a piece corresponding to a single pivot in (14); hence all the additional pivots correspond to modular vertices and can be performed by the techniques of section 4. With the exception of pivots of type case 2(e) of section 4, we can therefore simulate Kojima's algorithm in a standard restart algorithm - note that the latter requires only a linear system of order n+1 rather than 3n as in (14). Of course, n of the constraints of (14) are of generalized upper bound form.

The theorem also implies that the asymptotic results of quadratic convergence in Saigal [12], Saigal and Todd [14] apply to Kojima's algorithm also. No such results are available

for the homotopy H^3 [8] - the possibility of acceleration seems a powerful reason for choosing H^2 over H^3.

A slight modification is suggested for accelerating in the separable case. In the standard algorithm, having obtained an approximate zero x^1, we choose f^0 to have a zero at x^1. We also translate the triangulation so that $(x^1,0)$ is in the center of the face of the starting simplex $\langle \bar{y}^0,\ldots,\bar{y}^{n+1}\rangle$ opposite $\bar{y}^{n+1}$. This is done by arranging that $x^1 = \frac{1}{2n}(y^0 + y^n) + \frac{1}{n}\sum_{j=1}^{n-1} y^j$. Then if all zeroes of ℓ have projections within $\varepsilon/2n$ (in the ℓ_∞ norm) of x^1, only $n+1$ function evaluations and linear programming pivots are required to obtain x^2; here ε is the grid size of the triangulation $\tilde{K}_1$ or $\tilde{J}_1$ used. With f separable, it is preferable to translate the triangulation so that $x^1 = (\frac{1}{2} - \delta)(y^0 + y^n) + [2\delta/(n-1)]\sum_{j=1}^{n-1} y^j$, where δ is small and positive. In this case, as long as all zeroes of ℓ have projections within $(\frac{1}{2} - \delta)\varepsilon$ of x^1, only $n+1$ function evaluations and "nonmodular" linear programming pivot steps are required. Many more pivot steps may be needed, but they will all be of the trivial type considered in section 4. Thus the algorithms can be accelerated more safely in the separable case.

7. Functions with small bandwidth.

Recall that f has bandwidth $m = 2k-1$ if each component function f_i depends on x_j only for $|i-j| < k$. We assume $6k-4 \leq n$. Functions with small bandwidth arise naturally in discretizations of boundary value problems (see chapter 1 of [11]). In addition, if f is "sparse" (i.e. its Jacobian matrix is) we may permute the coordinates of the domain and range space to attempt to obtain a small bandwidth. See [6] for computational complexity results for minimizing the bandwidth by such permutations.

Our approach is based on that of Curtis, Reid and Powell [1]. Our aim is not just to avoid function evaluations but to promote the occurence of modularity. Fortunately these goals suggest the same strategy.

We would like to encounter simplices $k_1(\bar{y},\pi)$ such that adjacent members of π differ by at least m. While the algorithm is deterministic, we can at least control the starting simplex. We choose this to have associated permutation $\hat{\pi}$ = (1, m+1, 2m+1,...,2, m+2, 2m+2,...,...,m, 2m,...,n+1). Then the first step of the algorithm that does not transfer a vertex from $R^n \times \{0\}$ to $R^n \times \{1\}$ or vice versa entails a modular pivot. Indeed, since each simplex change affects the permutation only by an adjacent transposition, it is likely that several modular pivots will be encountered.

Under reasonable smoothness conditions on f, we know that asymptotically only $n+1$ simplices will be encountered. These all correspond to $k_1(\bar{y},\pi)$, with π containing the indices 1 through n in the same order as in $\hat{\pi}$, and $n+1$ moving from the last position to the first. In this case one can also economize on function evaluations.

Of course, if scalar function evaluations are much cheaper than vector evaluations one may economize at each step. If $\bar{y}^j$ is to leave the simplex and be replaced by $\hat{y}^j$, then $f(\hat{y}^j)$ can be obtained from $f(y^{j-1})$ or $f(y^{j+1})$ by making only the m scalar function evaluations in which it is know to differ. Frequently, however, economies in common expressions and the need to control subroutine calls suggest that vector function evaluations are more efficient. In this case one may proceed as follows.

Starting with $\langle \bar{y}^0, \ldots, \bar{y}^{n+1} \rangle = k_1(\bar{y}, \hat{\pi})$ we first calculate $f(y^{n+1})$. If, as hoped, $\bar{y}^n$ leaves the simplex, it is replaced by $\hat{y}^n$; we need $f(y^{n+1} - u^{\pi(n)}) = f(y^{n-1})$. Instead we calculate $f(y^{\pi^{-1}(m)-1})$. This evaluation gives us automatically $f(y^{\pi^{-1}(m)})$, $f(y^{\pi^{-1}(m)+1}), \ldots,$ $f(y^{n-1})$. If, as we again hope, $\hat{y}^n$ displaces $\bar{y}^{n-1}$, then we need $f(y^{n-2})$; but this is already known. It follows that if the sequence of simplices is as hoped (and asymptotically guaranteed) we need only evaluate f at $y^0 = y^{\pi^{-1}(1)-1}, y^{\pi^{-1}(2)-1}, \ldots, y^{\pi^{-1}(m)-1}$ and y^{n+1}, a total of $m+1$ function evaluations. In any case, we have not made any extra evaluations.

It is clear that, at a general stage of the algorithm, we may also try to guess the next few vertices that will be generated and make a function evaluation that will give us the value of the function at each of these vertices. Also, the more general grouping idea of Curtis, Powell and Reid [1] can be exploited in an analogous way - asymptotically only $m+1$ function evaluations are necessary each cycle, where here m is the number of groups.

REFERENCES

[1] A. R. Curtis, M. J. D. Powell, and J. K. Reid, "On the Estimation of Sparse Jacobian Matrices," J. Inst. Maths. Applics. 13 (1974), 117-119.

[2] B. C. Eaves, "A Short Course in Solving Equations with PL Homotopies," in Nonlinear Programming, Proceedings of the Ninth SIAM-AMS Symposium in Applied Mathematics, R. W. Cottle and C. E. Lemke (eds.), SIAM, Philadelphia, 1976.

[3] B. C. Eaves, and H. E. Scarf, "The Solution of Systems of Piecewise Linear Equations," Mathematics of Operations Research 1 (1976), 1-27.

[4] B. C. Eaves and R. Saigal, "Homotopies for Computation of Fixed Points on Unbounded Regions," Mathematical Programming 32(1972), 225-237.

[5] C. B. Garcia, "A Global Existence Theorem for the Equation $Fx = y$," Center for Mathematical Studies in Business and Economics Report 7527, University of Chicago, Chicago, Illinois (June 1975).

[6] M. R. Garey, R. L. Graham, D. S. Johnson and D. E. Knuth, "Complexity Results for Bandwidth Minimization," SIAM Journal on Applied Mathematics 34 (1978), 477-495.

[7] M. Kojima, "Computational Methods for Solving the Nonlinear Complementarity Problem," Keio Engineering Report 27 (1974), no. 1, Keio University, Yokohma, Japan.

[8] M. Kojima, "On the Homotopic Approach to Systems of Equations with Separable Mappings," Math. Prog. Study, 7 (1978), 170-184.

[9] M. Kojima, "A Modification of Todd's Triangulations J_3," Research Report B-45, Department of Information Sciences, Tokyo Institute of Technology, Tokyo.

[10] O. H. Merrill, "Applications and Extensions of an Algorithm that Computes Fixed Points of Certain Upper Semi-Continuous Point to Set Mappings," Ph.D. Dissertation, Department of Industrial Engineering, University of Michigan (1972).

[11] J. M. Ortega and W. C. Rheinboldt, Iterative Solutions of Nonlinear Equations of Several Variables, Academic Press, New York, 1970.

[12] R. Saigal, "On the Convergence Rate of Algorithms for Solving Equations that are Based on Methods of Complementary Pivoting," Mathematics of Operations Research 2 (1977), 108-124.

[13] R. Saigal, "On Piecewise Linear Approximations to Smooth Mappings," Northwestern University, Evanston, Illinois (November 1977).

[14] R. Saigal and M. J. Todd, "Efficient Acceleration Techniques for Fixed-Point Algorithms," to appear in SIAM Journal on Numerical Analysis, 1978.

[15] H. E. Scarf and T. Hansen, Computation of Economic Equilibria, Yale University Press, New Haven (1973).

[16] M. J. Todd, "Orientation in Complementary Pivot Algorithms," Mathematics of Operations Research 1 (1976), 54-66.

[17] M. J. Todd, "On Triangulations for Computing Fixed Points," Mathematical Programming 10 (1976), 322-346.

[18] M. J. Todd, The Computation of Fixed Points and Applications, Springer-Verlag, Berlin-Heidelberg, 1976.

[19] M. J. Todd, "Improving the Convergence of Fixed Point Algorithms," Mathematical Programming Study 7 (1978), 151-169.

[20] M. J. Todd, "New Fixed-Point Algorithms for Economic Equilibria and Constrained Optimization," Technical Report No. 362, School of Operations Research, Cornell University, Ithaca, New York (September 1977).

MJT/jvs

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER #1867	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) EXPLOITING STRUCTURE IN FIXED-POINT COMPUTATION		5. TYPE OF REPORT & PERIOD COVERED Summary Report - no specific reporting period
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Michael J. Todd		8. CONTRACT OR GRANT NUMBER(s) DAAG29-75-C-0024 ENG76-08749
9. PERFORMING ORGANIZATION NAME AND ADDRESS Mathematics Research Center, University of Wisconsin 610 Walnut Street Madison, Wisconsin 53706		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Work Unit Number 5 - Mathematical Programming and Operations Research
11. CONTROLLING OFFICE NAME AND ADDRESS See Item 18.		12. REPORT DATE August 1978
		13. NUMBER OF PAGES 25
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

U. S. Army Research Office
P. O. Box 12211
Research Triangle Park
North Carolina 27709

and

National Science Foundation
Washington, D. C.
20550

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Solving systems of equations, fixed-point algorithms, triangulations, structure

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

We consider the recent algorithms for computing fixed points or zeros of continuous functions from R^n to itself that are based on tracing piecewise-linear paths in triangulations. We investigate the possible savings that arise when these fixed-point algorithms with their usual triangulations are applied to computing zeros of functions f with special structure: f is either linear in certain variables, separable, or has Jacobian with small bandwidth. Each of these structures leads to a property we call modularity; the algorithmic path within a simplex can be continued into an adjacent simplex without a function

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

evaluation or linear programming pivot. Modularity also arises without any special structure on f from the linearity of the function that is deformed to f.

In the case that f is separable we show that the path generated by Kojima's algorithm with the homotopy H^2 coincides with the path generated by the standard restart algorithm of Merrill when the usual triangulations are employed. The extra function evaluations and linear programming steps required by the standard algorithm can be avoided by exploiting modularity.