POLYNOMIALS, CHAO-SEADER AND NEWTON RAPHSON - THE USE OF PARTIALLY ORDERED PIVOT SEQUENCES

by

Thomas J. Berna & Arthur W. Westerberg

DRC-06-1-79

January 1979

Department of Chemical Engineering
Carnegie-Melion University
Pittsburgh, PA 15213

This work was funded by NSF Grant ENG-76-80149.

ABSTRACT

Imbedded within a typical K-value routine, such as Chao-Seader, there is usually a cubic polynomial in compressibility factor. In general the Newton Raphson method, which linearizes and solves iteratively an entire set of nonlinear equations, is an unacceptable method if an imbedded function has two or more roots in the feasible region, as occurs here. Corrective action requires temporary replacement of the polynomial by one or more of the inequality constraints on the higher derivatives while converging toward the solution. This paper shows that, by restricting the permitted pivot sequence developed to solve the entire set of linearized equations, one can very simply incorporate and release these constraints without redeveloping the pivot sequence, a costly procedure when using sparse matrix codes. The ideas generalize.

SCOPE

One possible method for solving a flash problem is to gather together all the equations, including those arising in the evaluation of physical properties, and solve them using a Newton-Raphson based method. The equations can be written as $f(x) = 0$ where f and x are vectors of length n. The Newton-Raphson method is an iterative scheme which linearizes f about the current point, x(k) and solves for x(k+l) so as to drive f(x(k+l)) to zero. Thai: is$_s$ it solves the following for x(k+l):

$$\left(\frac{\partial f}{\partial x^T}\right)(x(k+1) - x(k)) = - f(x(k))$$

Two advantages associated with using a Newton-Raphson based method are that the method is easy to use and that it converges very quickly (quadratically) when it does converge. Two problems associated with the Newton-Raphson method are that it requires all first derivatives to be continuous near the solution and that the method indiscriminately converges to any root when multiple roots exist.

When solving a flash problem, one finds imbedded in the equation set for estimating vapor fugacity coefficients an equation of state for the fluid. The Chao-Seader method uses the Redlich-Kwong equation of state to estimate K-values. The Redlich-Kwong equation is a cubic polynomial in compressibility factor z, and, when two phases can occur, it has three roots. The lowest one is the liquid phase compressibility factor, the upper is the gas phase compressibility, while the middle one has no physical meaning.

While it would be desirable to use a Newton-Raphson based method to solve a flash problem, the fact that the first derivative changes sign for the cubic polynomial requires that we modify the iteration scheme. This is true for any problem where one or more functions exhibit multiple roots. The purpose of this paper is to demonstrate a rather simple strategy which will permit one to use the Newton-Raphson method for such problems. It involves automatically turning on and off, as needed, the appropriate inequality constraints to guide the Newton-Raphson method to the correct root. A minor change is needed for selecting the pivot sequence used to solve the linear equations developed for each iteration. The restricted pivot sequence together with a simple strategy to "clip[11] certain variables which are wandering through constraints completes the approach.

CONCLUSIONS AND SIGNIFICANCE

A simple strategy is presented which permits simulation problems involving functions having multiple roots (such as polynomials) to be solved via the Newton-Raphson method provided only that

1. The particular root of the function is uniquely characterized by the signs of a set of inequalities on higher derivatives.
2. The highest derivative considered does not change sign over the search space.

Results on three test problems demonstrate the effectiveness of the approach with convergence rates found which are characteristic of the Newton-Raphson method. The key feature of the method is that it precludes the need to redevelop the pivot sequence for the linear equations solved at each iteration when the inequality constraints are active and used temporarily to replace the original function.

For cubic polynomials, one can analytically locate the inflection points, and in many cases, the technique described here will seem too complicated for use with these problems. The significance of the method presented is that it permits one to use a gradient-based iteration scheme on systems of nonlinear functions where some of the derivatives change sign in the feasible region. It becomes most effective when used with higher order polynomials and functions involving exponential terms.

1. Chao-Seader and Newton-Raphson

In the course of solving Chao-Seader flash problems, one discovers imbedded in the equations the Redlich-Kwong equation of state, a cubic polynomial in the compressibility factor z. Usually this polynomial has three roots, the lowest one being the liquid compressibility factor and the uppermost the gas compressibility factor. The middle root has no physical interpretation. Figure la illustrates.

The root desired when calculating the vapor fugacity coefficient is the gas compressibility factor root which we see is uniquely characterized by three conditions

$$P_3(z) - 0$$

$$P\hat{}z) * 0 \qquad \text{(PI)}$$

$$p_3''(z) \geq 0$$

The middle root has $p^{3}(z) < 0$ and the lowest root has $p^{3}(z) < 0$. In principle then, we can find the desired root by placing these two additional constraints on our problem and solving, ufeing traditional approaches. (For example, we could solve using penalty functions (see Fiacco and McCormick[2]) or by using an active set strategy (see Abadie[l]).) Our goal, however, is to use a Newton Raphson based iteration scheme, and we would prefer not to have to reformulate the problem every time we move one of the constraints from the active set to the inactive set or vice versa.

Consider the following scheme for solving, where we are using the Newton Raphson method. We consider only the three equations

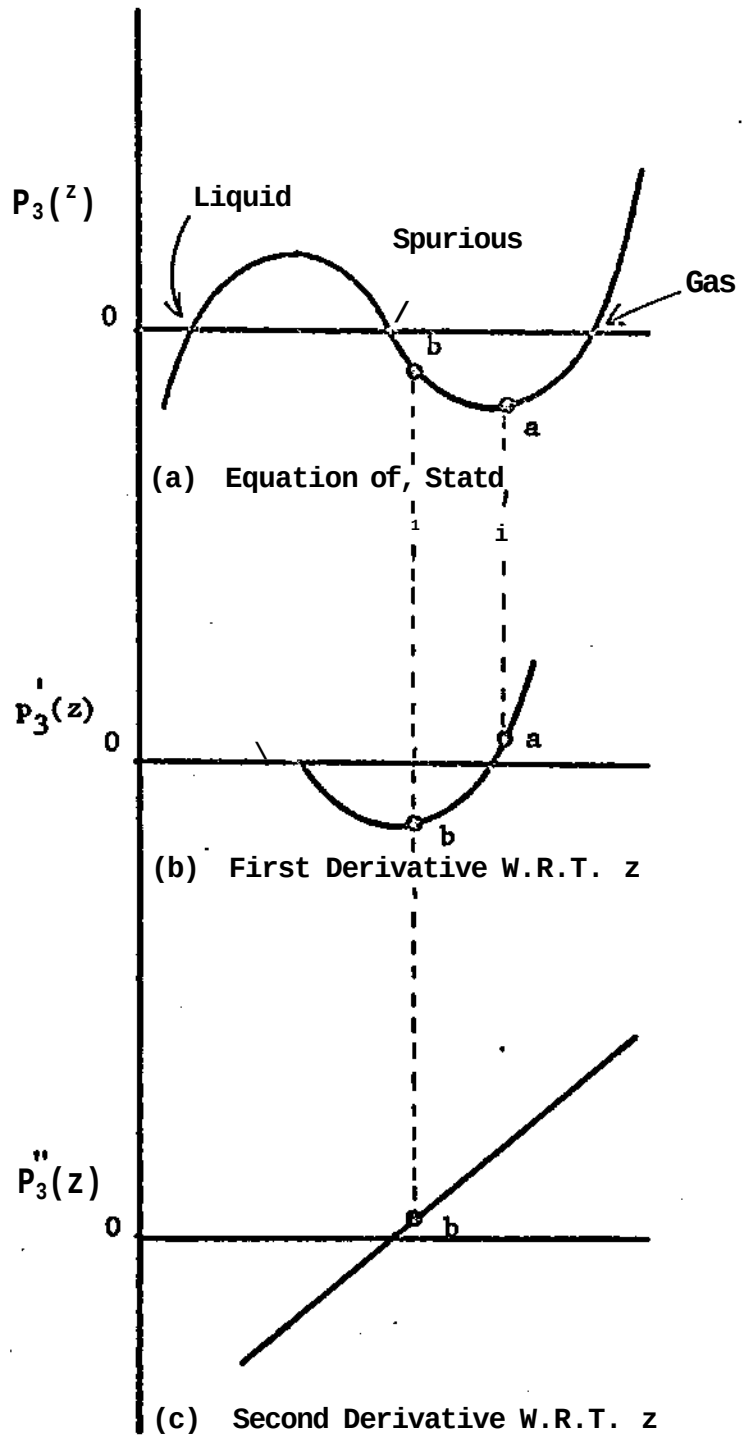

Figure 1. A Cubic Equation of State and its Derivatives

$$f_1 \equiv P_3(z) = 0$$

$$f_2 = s_2 - (P_3(2) - \cdot) = 0 \qquad \text{(P2)}$$

$$f_3 = s_3 - (p^f;(z) - \alpha) = 0$$

$$s_2,\ s_3 \geq 0$$

Here s_2 and s_3 are slack variables which are bounded below at zero, and $c > 0$ is a fixed small number. Note this formulation is essentially equivalent to our first problem except for the occurrence of $f_i > 0$. If f_2 is satisfied with $s_2 = 0$, then $p_3(z) = e$ results, and we are slightly to the right of the minimum (i.e. at point "a") in Figure la. At this point the derivative of f_1 with respect to z is not zero, a possibility which can cause computational problems if allowed to occur. An efficient procedure for computing x(k+l) using the Newton Raphson method is to perform an L/U factorization of the Jacobian matrix $\left(\frac{\partial f}{\partial x^T} \right)$ A description of this procedure is included as an appendix for those who are not familiar with the technique. The method is entirely equivalent to solving by Gaussian elimination where the factor L is the matrix equivalent of the forward elimination and matrix U the resulting matrix used in the backward substitution step. Let us examine the effects of using L/U factorization (i.e. Gaussian elimination) as it applies to problem (P2) . The Jacobian matrix $\partial f / \partial x^T$ has the following zero/nonzero pattern

	Az	As_2	As_3
f_1	⊗		
f_2	x	⊗	
f_3	x		⊗

any any self-respecting sparse matrix code would choose the circled pivot sequence as it causes no fill-in to occur. This pivot sequence says we should first use the linearized equation for f_1 to find Az, then f_2 to find As_2 and finally f_3 to find As_3.

Let us now observe the consequences of selecting this pivot sequence combined with the intuitive strategy to clip s_2 and s_3 whenever they are out of bounds as we seek the solution to our problem. We note that Az is found from the linearized equation for f_1 irrespective of the values obtained for As_2 and As_3. Thus, if we are near to and are approaching the middle root in Figure la, we shall continue to approach it. Near this root we shall find both As_2 and As_3 being clipped because solving f_2 and f_3 will result in both s_2 and s_3 being negative. Note that if we were to compute Az based on the linearized equation for f_2 we would approach point "a" in Figure la. Once at point "a" we could use f_1 to compute Az, and this procedure would bring us to the desired root. The procedure for doing this is as follows:

Step 1

Using a "constraint replacement" strategy, discard momentarily $f_1(z) = 0$ and introduce the active constraint $s_2 = 0$, giving us the problem

$$s_2 = 0$$

$$s_2 - (p \wedge z) - e) = 0$$

$$s_3 " (P \wedge 2) - c) = 0$$

Solving this problem will lead to point "a" in Figure la.

Step 2

Recover the original equation set and solve for Az using the Newton-Raphson equation. From point "a", we clearly move to the correct root of $P_3(z) = 0$.

The difficulty with the above approach is that we must reformulate our problem twice in the course of solving. Also, we had to discard temporarily an equality constraint of the original problem. Problem reformulation is at best tedious and at worst extremely time consuming, and we would like to avoid it.

Now consider the following very unnatural pivot sequence for our original equation set (P2)

	Δz	Δs_3	Δs_2
f_3	(x)	x	
f_2	x	(*)	x
f_1	x	*	(*)

where the asterisks indicate that the location will fill-in during the course of the L/U factorization (i.e. Gaussian elimination). We first pivot on the coefficient of Δz in the linearized equation for f_3. Eliminating the nonzeros in column 1 below f_3 will cause the $(f_2, \Delta s_3)$ and $(f_1, \Delta s_3)$ elements to become nonzero. Thus the coefficient of Δs_3 can be used as a pivot in f_2. The $(f_1, \Delta s_2)$ fills in similarly when we pivot the second equation.

In solving we first do the forward elimination just described, getting finally the upper triangular structure which corresponds to the U matrix in the L/U factorization

	Δz	Δs_3	Δs_2
f_3	x	x	
f_2	0	x	x
f_1	0	0	x

Performing a backward substitution, we now solve the last equation for As_9 and check if this perturbation will cause S_2 to become negative. If so, we can immediately select As_2 so s_2 is zero. Now the middle euqation is solved for As_3 in terms of whatever value we have given As_2 — that is, whether it came from the last equation or by being set to a value to prevent s_1 becoming negative. Again if As_vwould cause S_3 to become negative we alter its value to cause s_3 to be zero. Lastly we solve for Az in terms of As_3-

Examining the behavior of this scheme, we find a very different result from before. If we are in the vicinity of the middle root in Figure la, the slack variables s_2 and s_3 would be negative, as both the slope and the curvature of the cubic equation are negative at this point. Thus we will find that in the backward substitution step both As_1 and As_3 will be clipped, that is, set to $-s_2$ and $-s_3$, respectively. The only equation satisfied of the three will be f_3 which will, when solved with $s_3 = 0$, move us away from the middle root and to the point "b" in Figure la. At this point As_3 is no longer clipped, and both equations f_3 and f_2 will be satisfied, with $s_3 > 0$ and $s_2 = 0$. Being satisfied with $s_2 = 0$, f_2 moves us to point "a". At this point all three equations are active, and we will move directly to the desired upper root.

We note that we never reformulated the pivot sequence, and we can readily verify that this behavior of moving ultimately to our desired root occurs no matter where we start with our initial guess for z. What we did was choose an initial pivot sequence so if As_3 was clipped then equation f_3 was not allowed to be violated, while f_2 and f_1 were. Also, if only As_1 was clipped, then equations f_3 and f_2 were not allowed to be violated. The above is accomplished by forcing a partial ordering onto the (allowed) original pivot sequence.

If we clip a variable on the back substitution step as indicated, the pivot equation and potentially any below it may no longer be satisfied at the resulting solution point. However, any occurring above the pivot equation can receive any value for the clipped variable and will be exactly satisfied. The ordering is such that we cannot select s_3 as a pivot variable until equation f_3 is already pivoted, and we cannot select s_1 until both f_3 and f_2 are pivoted. We note this ordering by listing for each variable the equations which cannot be violated if the variable is clipped as indicated (IE = Inviolate Equation).

$$IE(s_3) = \{f_3\}$$

$$IE(s_2) = \{f_2, f_3\}$$

Developing the pivot sequence in a sparse matrix code can then proceed by whatever scheme is desired but modified to refuse s_3 as a pivot until after equation f_3 is pivoted, etc. Note the above IE sets force f_3 to be the first equation selected, and they force Az to be the pivot variable. (Neither As_0 nor As_0 can be used until after f_0 is used, therefore if f_0 is not selected, neither As_2 nor As_3 can be used in the first two pivots.) Equally f_2 and As_1 are forced to be the second pivot equation and variable, respectively, leaving As_2 and f_1 for last. It is straightforward to develop an algorithm to discover when a pivot is being forced without examining all possible sequences.

2. The General Problem

We can now examine the general implications of the observations made in Section 1. We are interested in finding values for each of the n components of the vector x which will drive the n functions f(x) to zero. Some of the components of x are subject to upper and/or lower bounds; we let S represent the set of all values for x which satisfy these inequality constraints. Finally, the convergence criterion will be that the sum of the squares of the residuals, represented by $f^T f$ be less than some small positive number, 6. The mathematical expression which describes this problem is:

$$\underset{x}{\text{Min}} \{qp | qp = f^T f;\ x \in S\} \qquad \text{(P3)}$$

The Newton-Raphson method for solving (P3) is to generate a sequence {x(k)} which converges to $\hat{x}$, where, at the k^{th} iteration, x(k-fl) is found by solving

$$\left[\frac{\partial f}{\partial x^T} \right]_{x(k)} (x(k+1) - x(k) = - fx(k)) \qquad (1)$$

The matrix $\left(\frac{\partial g_A}{\partial x} \right)_{x(k)}$ *s t^ie Jac°b*-an matrix of partial derivatives. The only inequality constraints we mentioned when formulating (P3) were simple bounds on some of the variables. In general, the original problem may also contain a number of nonlinear inequality constraints and some linear inequalities involving more than one component of x. In solving (P3) we find it convenient to transform these inequalities into equality constraints with slack variables. These equality constraints are then added to the vector g so that all constraints in S take the form of simple bounds on the independent variables. For discussion purposes, any variable subject to urp=r

or lower vounds will be called a bounded variable (BV). Any BV outside or on the boundary of S will be called an "active[11] bounded variable (ABV). While it may seem somewhat cumbersome at first to talk about IE'S, BV[1]s and ABV's, we are convinced that this nomenclature simplifies the overall discussion. For the reader's convenience these abbreviations are summarized in Table 1.

Often in the course of a Newton Raphson search, equation (1) predicts a value for x(k+l) which lies outside of S. Our main objective is to develop a computationally convenient and effective procedure for handling this situation. Figure 2 illustrates the case where equation (1) gives x(k+l) at point A. We require that x(k+l) always lie within S; one reason is that some functions (e.g. $\sqrt{x}$) are simply not defined for x outside of S (e.g. x < 0). One strategy is to replace $i \wedge 0$ or $f_3 = 0$ by $s \wedge 0$ giving x(k+l) at B or E (in Figure 2), respectively, problems with this procedure center around deciding which constraint to release and how to avoid cycling among the held and released constraints. Another strategy is to find the intersection of the Newton Raphson step and the violated constraint. This procedure places x(k+l) at point C in Figure 2. Still another strategy simply "clips[11] any variable which is projected through one of its bounds. The result of using this strategy is to place x(k+l) at point D in Figure 2. Note that this procedure represents an orthogonal projection into S from A and gives the best location for x(k+l) in the lease squares sense.

In the course of solving a problem such as a Chao-Seader flash problem, it becomes desirable to label certain equations as "inviolate equations[11] (IE[f]s); the consequence is that orthogonal projection back into S

Table 1. Definitions Associated with Partially Ordered Pivot Sequences

Symbol	Description
$IE(x_j)$	The set of Inviolate Equalities associated with the variable x_j. When developing a pivot sequence, one must pivot all IE'S in $IE(x_j)$ before x_j may be used as a pivot.
IE	Inviolate Equality. Whenever the Newton based search procedure predicts a point outside of S, we would like simply to perform an orthogonal projection back into S. This is possible in most cases, but when IE's are present the projection must lie along the linearized IE'S.
BV	Bounded Variable. Any variable subject to upper and/or lower bounds may be "clipped[11] during the Newton-Raphson iteration scheme; therefore, we flag these variables by calling them BV's.
ABV	Active Bounded Variable. This term refers to a BV which is about to be clipped; that is, any BV whose next value is predicted outside of S is termed an ABV.

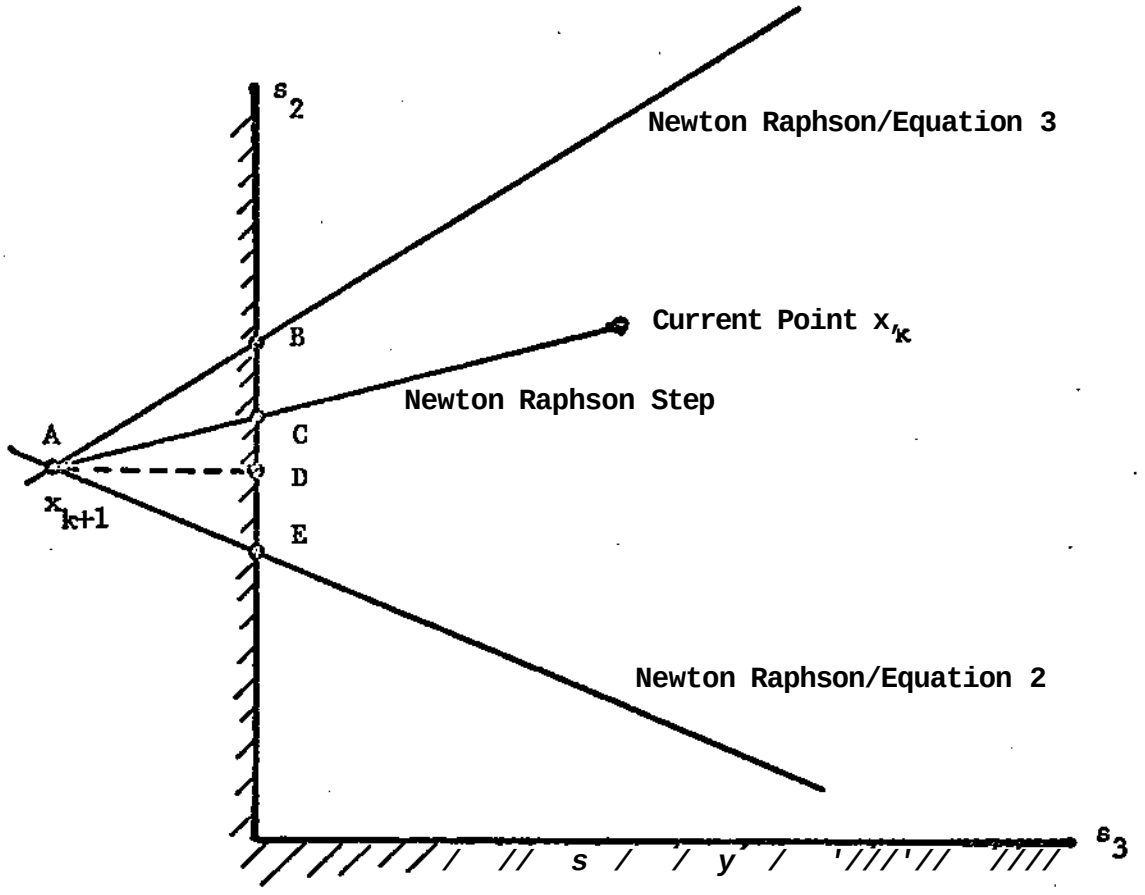

Figure 2. possible Strategies to Move Back into the Feasible Region

is not permitted in certain coordinate directions. Projection must lie along an IE (inviolate equation) whenever an associated ABV (active bounded variable) occurs in that equation. For example, if x is outside S and f_1 and f_2 belong to IE(x), then projection back into '" must lie along the linearized constraints $f_1 = 0$ and $f_2 = 0$. We use orthogonal projection where possible and projection along IE's where required. For problem (P2), f_i lE(s_j) so we must move to point B in Figure 2. In the next paragraph, we formally state the algorithm.

Step 0; Initialization

i) Set k = 0, x = x(0)

ii) Set IEFLAG (i,j) = 1 if f_i e IE(x_j) and set to zero otherwise for all i,j - l,*#-,n.

Step 1: Compute Residuals and Check for Convergence

i) Compute f(x(k)) and $f^T f$

ii) If ($f^T f$ £ 4) go to Step 4

iii) If (k > (maximum number of iterations allowed)) go to Step 3

Step 2: Compute x_{k+1}

i) Compute $\frac{\partial f}{\partial x^T}$ at x(k)

ii) Factor $\frac{\partial f}{\partial x^T}$ into L and U making sure that x_j is not pivoted until all rows in IE(x_j) have been pivoted

iii) Solve Lq = -f by forward substitution

iv) Perform backward substitution to solve Uy = q.

When performing the back substitution use y_j if IEFLAG (i,j) is zero, and use z_j otherwise. The vector y contains the undipped Newton step and z contains the same step except that it is clipped where necessary to keep x(kfl) inside the feasible region.

v) $x(k+1) = x(k) + z$

vi) $k = k + 1$

vii) Go to Step 1

Step 3: Convergence Failed

Print error message and go to Step 5

Step 4: Convergence Achieved

Print final results and go to Step 5

Step 5: Stop

We shall explain the purpose of introducing the two vectors y and z in Step 2(iv) by the following example. Reconsider problem (P2) in linearized form. Only this time include two additional variables x_4 and x_5 and two linearized constraints f_4^A and f_5^A in the following sequence and with the following total zero/nonzero pattern

	Δz	Δs_3	Δx_4	Δs_2	Δx_5
f_3	⊛	x			
h	x	⊛		x	
f_4^A	x	*	⊛	*	x
f_1	x	*	x	⊛	x
f_5^A	x	*	x	*	⊛

Again the asterisks '*' indicate locations which will fill in during the forward Gaussian elimination step if the pivot sequence illustrated is used.

Suppose we have completed the forward elimination step and have just reached function f_1 on the backward substitution step. We discover As_2 has to be clipped and do so. The effect is, of course, to override the value which would have resulted from using f_1. f^5 has already been used to obtain a value for Ax_5 based on equation f_1 so it too will no longer hold. We now have to backsubstitute using f_4 to calculate Ax_4. Do we use As_2 which f_1 gives or the clipped As_2? We see that f_4 and f_5 are really innocent bystanders to our treachery and are in this example entirely equivalent in original zero/nonzero pattern. It seems therefore f_4 has a choice whereas f_5 was given no such opportunity. Adopting the procedure in Step 2(iv), f_4 and f_5 are treated equally, both using the value for As_2 given by using f_1. Only those equations in IE(s_2), namely f_4 and f_5, must use the clipped value for As_2 which again Step 2(iv) does. So Step 2(iv) is there to treat all non-inviolate equations equally whether they appear before or after the clipping of a bounded slack variable in the pivot sequence and to treat all inviolate equations as proposed earlier. Ax_4 and Ax_5 values are not affected by the clipping and thus the clipping is like an orthogonal projection from the solution point back to the feasible region with respect to f_4 and f_5. On the other hand f_4 and f_5 are held inviolate with respect to clipping As_2; that is, we must move back to the feasible region along these constraints.

3, Multiple-Root Functions

In the previous section we presented an algorithm for carrying out the modified Newton-Raphson iteration scheme described earlier in this paper. Before this algorithm can be applied, we need a procedure for setting up the sets of appropriate inviolate equalities which will guarantee that this new algorithm will converge to the desired root of any function

which exhibits multiple roots. Even if a function has only one real root in the region of interest, some of its derivatives may change sign, thus raising havoc with a standard Newton-Raphson iteration scheme. In order to apply our method two conditions are necessary:

1. The desired root is uniquely characterized by the signs of the first P derivatives, and the user knows these signs a priori.
2. The sign of the P + 1^{st} derivative is constant for all feasible values of $\underline{x}$. The user need not know this sign.

The second condition can always be met for polynomials, but not all functions have a derivative which does not change sign over the feasible region of the independent variable, e.g. sin(x), $x\varepsilon[2\pi]$. For many realistic problems, P is small and the first condition does not represent a major difficulty. Given a set of m functions which satisfy conditions 1 and 2, we can use the following algorithm to define the appropriate sets of inviolate equalities (IE's).

Step 0: Initialization

i) Label all variables which are subject to upper and/or lower bounds as bounded variables (BV's).

ii) Set k = m and i = 1

Step 1: Fill $IE(x_j)$ for the P derivatives of f_i as follows

i) If the value of P corresponding to f_i is zero, go to Step 2.

ii) For $\ell = 1,\cdots,P$ write the correct inequalities $\pm f_i^{(\ell)} \geq \pm \epsilon$.

iii) For $t = 1,2,\#\#\#,P$ introduce slack variables $x^{\wedge}_{\ldots+\prime}$ to generate equality constraints of the form $f_{,k+\prime}:fj^{\acute{}}$ - « $\pm x,_{k+\&}$ i = 0

$x_{k+\&}$ - is a BV subject to a lower bound of zero.

iv) For $I = 1,2,**',P$ for each bounded variable x_j in constraint $f_{k+\&}$ add f_{k+X} •" f_{k+P} to the set $IE(x_j)$.

v) Add f_{k+1} ••• f_{k+P} to $IE(x_j)$ for all bounded variables x_j which appear in f_i.

Step 2: Move to the Next Function

i) Set k = k + P; set i = i + 1

ii) If (i £ m) go to Step 1

Step 3; Stop

i) Set n = k

ii) STOP

The effect of using the two algorithms presented here is to guarantee that the variable is moved in the proper direction (in spite of the fact that some derivatives are changing sign) with each Newton Raphson step. The IE corresponding to the highest derivative will be pivoted before its slack variable or that associated with any derivatives of lower order. In this way, if any of the slack variables are clipped, all higher derivatives will still be satisfied. In contrast, equations corresponding to derivatives of lower order will not be satisfied, but it does no good to satisfy these equations until those of higher order are all satisfied.

Clearly this strategy hinges only on the fact that we have a set of equalities for which a hierarchy exists for satisfying them. That is, there is one constraint which, if not satisfied, makes the remaining constraints meaningless — thus they can be momentarily discarded. Once satisfied, then another constraint becomes the dominating one. For a polynomial whose general shape is known at the solution, we can always find such a sequence to force us to the desired root.

4. Examples

In the next examples, all variables were constrained between -100 and 100. All pivots smaller in absolute value than 0.0001 were considered to be zero, and e was chosen as 0.1. As long as the IE's were pivoted first, the procedure converged for all legitimate pivot strategies and starting points, of which a wide variety were tried.

Example 1

We first illustrate the effectiveness of the procedure by applying it to a cubic polynomial with constant coefficients. We desire the uppermost root of

$$f_1: \quad x_1^3 + x_1^2 - 5x_1 - 10 = 0$$

In setting up this problem we follow the algorithm given in Section 3. Initially m = 1, but we need to introduce two constraints on the derivatives of f_1:

$$f_1' \geq \epsilon$$

$$f_1'' \geq \epsilon$$

We set m = 3 and introduce two slack variables corresponding to the two functions generated by the above inequality constraints

$$f_2: 3x_1^2 + 2x_1 - 5 - x_2 - 6 = 0$$

$$f_3: 6x_1 + 2 - x_3 - e = 0$$

$$x_2, x_3 \geq 0$$

Step l(iv) creates the following IE sets

$$IE(x_2) = \{f_2, f_3\}$$

$$IE(x_3) = \{f_3\}$$

As indicated by the discussion in Section 1, the only allowable pivot sequence is

Row:	f_3	f_2	f_1
Column:	x_1	x_3	x_2

Table 2 compares the results obtained using the standard Newton-Raphson technique with orthogonal projection (that is, the solution point is found and then all variables which are out of bounds are simply placed back onto their bounds) against the results obtained by using the procedure recommended here. When the superfluous constraint $x_1 \geq 0$ is added to the problem, the first technique is unable to move away from the origin. It is clear from Table 2 that the partially ordered pivoting combined with clipping of ABV's during the backward substitution step causes the problem to converge quickly. This result is because clipping $\Delta x_2 = -9$ to zero during backward substitution in the first Newton-Raphson step caused Δx_3 (and then Δx_1) to become large enough to satisfy the constraint on the second derivative.

Table 2« Comparison of Pivoting Strategies for *Example* 1

Newton Raphson with Orthogonal Projection

Iteration	x_1	x_2	x_3	f_1	f_2	f_3
0	0	0	0	-10	-5	2
1	-2	0	0	- 4	3	-10
2	-.67	0	0	-6.5	-5	- 2
3	-1.97	0	0	-3.91	2.70	-9.82
4	- .52	0	0	-7.27	-5.23	-1.12
5	-1.91	0	0	-3.77	2.12	-9.46
6	- .13	0	1.22	-9.34	-5.21	0.00
7	-1.92	0	0	-3.97	2.22	-9.52

Partially Ordered Pivoting with ABV Clipping

Iteration	x_1	x_2	x_3	f_1	f_2	f_3
0	0	0	0	-10	-5	2
1	2.5	0	17	- .63	18.75	0
2	2.53	19.36	17.21	- .05	.10	-.03
3	2.533	19.31	17.20	.003	.004	-.002

Example 2

Here ve wish to find the larger root of a quadratic polynomial with variable coefficients.

Using the algorithm in Section 3, we first set m = 5. The equations are:

$$L_1: \quad ax_1^2 + bx_1 + c = 0$$

$$f_2: \quad ax_1 - x_2 = 0$$

$$f_3: \quad b + x_1 x_2 = 0$$

$$V \quad cx_1 + x_2 = 0$$

$$f_5: \quad x_1 + x_2 - 1 = 0 \quad .$$

The bounds on the variables are:

$$0 \leq x_1, x_2 \leq 1$$

$$0 \wedge a$$

$$c \pounds 0$$

Following the algorithm in Section 3 we find that only $\pounds_1$ has a non-zero value for P; i.e., we are going to add a constraint on the first derivative of f. For f_1, P = 1 (first derivative only) so we introduce the following constraint to guarantee convergence to the upper root of $\pounds_1$.

$$f_6: \quad 2ax_1 + b - x_3 - e = 0$$

$$x_3 \geq 0$$

$$IE(x_3) = \{f_6\}$$

$$IE(a) = \{f_6\}$$

$$IE(x_1) = \{f_6\}$$

The only restriction on the pivot sequence then is that f_6 be pivoted before either x_0, a or x-. Note the variable b, which also appears in f,, is not bounded so no inviolate equation set is needed for b.

Table 3 gives the results obtained for this system of equations. The starting point chosen gives a singular Jacobian matrix of rank 5 initially. Our convention is to set the perturbation variable corresponding to a column which cannot be pivoted to zero. When this procedure is used, the standard Newton-Raphson technique with orthogonal projection is unable to move away from the origin.

Example 3

Finally, we consider a cubic polynomial with variable coefficients. We wish to find the largest root. This example is the most closely related to the Chao-Seader problem of the three examples. The original set of equations is:

$$f_1: \quad x_1^3 + bx_1^2 + cx_1 + d = 0$$

$$f_2: \quad a - 2b = 0$$

$$f_3: \quad 2a + c = 0$$

$$f_4: \quad 4b - 5d = 0$$

$$f_5: \quad c + dx_1 + 4 = 0$$

$$a, x_1 \neq 0$$

$$d \neq 0$$

Table 3. Results for Example 2

Iter	a	b	c	x_1	x_2	x_3	f_1	f_2	f_3	f_4	f_5	f_6
0	0	0	0	0	0	0	0	0	0	0	-1	-.1
1	0	.1	0	1	0	0	.1	-1.	.1	0	0	0
2	2.1	-.1	-1.	3#-08	1	3.1	-1.	2.1	-1	1	0	-4.2
3	1.55	-.5	0	.5	.5	1.5	.14	.28	-.25	.5	0	-.55
4	2.032	-.25	-.5954	.7073	.2977	2.309	.23	-.097	-.041	-.121	0	.19
5	1.766	-.2326	-.5561	.6441	.3559	1.912	.027	-.015	-.003	-.002	0	.031
6	1.760	-.2311	-.5681	.6377-	.3623	1.913	10^{-4}	IO^{-5}	io^{-5}	io^{-5}	0	lo^{-5}

Again, constraints will only be written on derivatives of f_1, so only f_1 has a value for P greater than zero. Here P = 2 because we require that both the first and second derivatives be non-negative

$$f_6: \quad 3ax_1^2 + 2bx_1 + c - x_2 - \epsilon$$

$$f_7: \quad 6ax_1 + 2b - x_3 - \epsilon$$

$$x_2, x_3 \geq 0$$

The bounded variables a, x_1, x_2, x_3 and d appear in f_1, f_6 and f_7 so we get the following IE's

$$IE(a) = \{f_6, f_7\}$$

$$IE(x_1) = \{f_6, f_7\}$$

$$IE(x_2) = \{f_6, f_7\}$$

$$IE(x_3) = \{f_7\}$$

$$IE(d) = \{f_6, f_7\}$$

The only restrictions on the pivot sequence are

i) f_7 must be pivoted before x_1, x_2, x_3, a or d

ii) f_6 must be pivoted before x_1, x_2, a or d.

Table 4 gives the results obtained using the procedure recommended in this paper. As in Example 2, the standard Newton-Raphson technique with orthogonal projection is unable to move away from the origin.

Table 4. Results for Example 3

Iter.	a	b	c	d	x_x	x_2	x_3	^	f_2	f_3	f^	f_5	f_{fi}	$f_?$
0	0	0	0	0	0	0	0	-.1	0	0	0	4.0	-.1	0
1	0	.05	.1	0	.1	0	0	.0105	-.1	.1	.2	4.1	.01	0
2	1.961	.9804	-.1449	.7843	0	0	3.037	.7843	0	3.777	0	3.855	-.2449	-1.176
3	.9604	.4802	-1.921	.3841	2.651	3.177	32.05	16.56	0	0	0	1.061	17.60	-15.91
4	1.548	.7291	-2.916	.5833	1.446	12.04	17.61	2.303	0	0	0	.2399	-3.791	-3.598
5	1.621	.8103	-3.241	.6483	1.140	4.846	12.90	.4060	0	0	0	.02	-2.475	-.2989
6	1.651	.8255	-3.302	.6604	1.056	3.846	12.02	.037	0	0	0	.001	.015	-.015
7	1.654	.8269	-3.308	.6616	1.046	3.755	11.94	10^{-4}	0	0	0	io^{-5}	io^{-4}	io^{-4}

REFERENCES

1. Abadie, J. and J. Carpentier, "Generalization of the Wolfe Reduced Gradient Method to the Case of Nonlinear Constraint,[11] in Optimization, Academic Press, New York, pp 37-47 (1969).

2. Fiacco, A. and G. McCormick, Nonlinear Programming; Sequential Unconstrained Minimization Techniques, John Wiley, New York (1968),

3. Reid, J. (ed), Large Sparae Sets of Linear Equations, Academic Press, London (1971).

ACKNOWLEDGEMENT

This work was funded by NSF Grant ENG 76-80149.

APPENDIX: L/U FACTORIZATION

An efficient technique for solving a system of linear equations of the form Ax = b is to factor the coefficient matrix A into the product of a lower triangular matrix L and an upper triangular matrix U. This factorization process is very similar to performing a Gaussian elimination. The L/U factorization algorithm we used in our test problems is given below. We initialize IP and JP to satisfy the restrictions imposed by the sets of IE^fs, and the only reason for changing the ordering is to avoid a zero point. This algorithm is for illustrative purposes only.

Step 0: Initialization

i) Set $k_p = 0$. Define row and column number index sets IP and JP, respectively, to contain a pivot sequence which satisfies the restrictions imposed by the sets of IE^fs.

ii) Set $n_o = 0$. (n_o will count the number of columns which become essentially all zero before a pivot element can be found in them. These "nonpivot" columns will correspond to perturbation variables whose values will be left at zero. The matrix is singular, and this approach is the one most commonly adopted to handle this case.)

Step 1: Find Next Pivot

i) Set $k_p = k_p + 1$

ii) Set $j \leftarrow JP(k_p)$

iii) Set $k_\pm = k_p$

iv) Set $i = IP(k_\pm)$

v) If $|a_{ij}| < e$ (a small number), reject as next pivot and go to Step (vi)• Otherwise interchange $IP(k_i)$ and $IP(k_p)$ row numbers and go to Step 2.

vi) Set $k_i = kj + 1$. If $k_i < n - n_o$ go to Step (iv) . Otherwise continue.

vii) No pivot is found in this column. Set $i = IP(k_p)$. Move all row and column numbers from position $k_p + 1$ forward one position in IP and JP. Then put IP(n) = i, JP(n) = j and $n_o + 1$.

viii) If $k_p < n - n_o$ return to Step (ii). Otherwise go to Step 3.

Step 2: Perform Elimination Using Current Pivot

i) Set $P = a_{\pm j}$. Set $i = IP(k_p)$

ii) For $Jl - k_p + l, \ldots, n - n_o$

— Set j - JP(J6)

- Set $a_{\pm J} - a_{\pm j}/P$

iii) Set $i^1 = IP(k_p)$, $j^f - JP(k_p)$

iv) For $m = k_p + l, \ldots, n - n_o$

For $n = k_p + l, \ldots, n - n_o$

— Set i = IP(m)

— Set j = JP(n)

— Set $a_{ij} = a_{ij} - a_{ij'} \cdot a_{i'j}$

v) Return to Step 1.

Step 3; Factorization Complete; STOP

The original matrix now has the non-zeros in L stored on and below the diagonal iri pivot sequence. The non-zeros for U are stored above the diagonal, except for the diagonal elements which are all unity.

Once the coefficient matrix has been factored, solving a linear system is accomplished in two steps. Note that the original right hand side should be rearranged to conform to the row pivot sequence and the solution vector should be rearranged to conform to the column pivot sequence. We first solve

$$L\ y = b$$

for y. This is accomplished by working from the first row of L down to the last row. We then solve

$$U\ x = y$$

to obtain the desired solution x arranged in column pivot sequence. This second step is most easily accomplished by working from the last row of U upward. The variables corresponding to zero pivots can be set to any value; we choose to set them to zero.