INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING, VOL. 37, 2093–2107 (1994)

THE MONOMIAL METHOD: EXTENSIONS, VARIATIONS, AND PERFORMANCE ISSUES

SCOTT A. BURNS*

University of Illinois at Urbana—Champaign 104 S. Mathews #117 Urbana, IL 61801 U.S.A.

SUMMARY

The monomial method solves systems of non-linear algebraic equations by constructing a sequence of approximating monomial (single-term polynomial) systems, much as Newton's method generates a sequence of linear systems to do this. Since the monomial system becomes linear through a logarithmic transformation of variables, the monomial method can be considered to be an alternative linearization scheme. Although the monomial method is closely related to Newton's method, it exhibits many special invariance properties not shared by Newton's method that enhance performance. This paper first briefly reviews the monomial method and its special properties. Two new versions of the algorithm are presented, both of which are simplified and computationally more efficient to implement in comparison to the original algorithm. The monomial method is also extended to apply to certain non-algebraic systems. Since the monomial method can be interpreted as Newton's method applied to a three-part reformulation of the algebraic system, graphical experiments are presented which investigate the role that each part of the reformulation plays in contributing to the enhanced performance. Finally, instances in which difficulties have arisen using the monomial method are discussed.

1. INTRODUCTION

When a closed-form solution of a system of non-linear equations is impossible or impractical, numerical methods often provide a means of approximating one or more solutions. Newton's method is a well-known method for doing this. By replacing the non-linear system with a sequence of easier-to-solve linear systems, a solution is approached iteratively from a user-provided starting point. When multiple solutions exist, they can sometimes be found by initiating Newton's method from several different starting points.

Linearization is a fundamental way of locally approximating non-linear phenomena, and is pervasive throughout numerical solution techniques. Newton's method uses the linear part of the Taylor series expansion, evaluated at the operating point, to approximate each non-linear equation. The monomial method is based on an alternative to this type of linearization.

Suppose that instead of approximating the non-linear system, $F_k(x_1, x_2, \ldots, x_N) = 0$, $k = 1, \ldots, N$, with a linear system, $\sum_{j=1}^{N} M_{jk} x_j = b_k$, we approximate it with a monomial system, specifically a system of positive single-termed polynomials of the form $C_k \prod_{j=1}^{N} x_j^{A_{jk}} = 1$. At once, it is obvious that this monomial system becomes linear when transformed into log space using $z_j = \ln x_j$. This linear system, $\sum_{j=1}^{N} A_{jk} z_j = -\ln C_k$, is easily solved, and, as with Newton's method, the solution provides the operating point for the subsequent iteration.

At first glance, this 'monomial method' might be considered as just a curious alternative to

* Associate Professor

CCC 0029–5981/94/122093–15$9.00

Received 17 December 1992
Revised 13 October 1993

Newton's method, with no obvious indication that it should perform significantly better or worse. Yet, over the past 15 years, the author has observed that quite often the monomial method strikingly outperforms Newton's method. A recent project concerning the computer graphics-based visualization of the performance of numerical methods[1] provided clues that led to the discovery of special invariance properties of the monomial method. These special properties are not shared by Newton's method and are responsible, at least partially, for the enhanced performance of the monomial method. These special properties are described in a previous work.[2] This paper is concerned with (1) refining and qualifying some of these special properties, (2) reporting recently discovered previous work in the area, (3) extending the monomial method to apply to certain non-algebraic systems, (4) presenting two new simplified algorithms that are computationally simpler and more efficient to implement, and (5) discussing instances where severe difficulties have arisen using the monomial method.

2. PREVIOUS WORK

The idea of approximating functions by monomials is not new. Duffin *et al.*[3] suggested this as a means of expanding the applicability of geometric programming beyond polynomial functions with positive coefficients ('posynomials'). Their approximation was expressed in terms of partial derivatives of the function being approximated. A few years later, Duffin introduced a process called 'condensation',[4] which is mathematically equivalent to the previous monomial approximation, but is applied specifically to posynomials through the application of the weighted arithmetic-geometric mean inequality.[5,6] Condensation has traditionally been applied to optimization problems to approximate polynomial functions with posynomial functions, to reduce the 'degree of difficulty' of geometric programs to make them easier to solve by dual-based techniques, or to solve primal geometric programs directly without the use of the dual.[7-10] The author has applied condensation techniques to incorporate equality constraints within the framework of generalized geometric programming.[11]

Rijckaert and Martens[12] first suggested applying condensation specifically to solve systems of non-linear equations. They did not advance the idea as a general-purpose method for solving systems of equations, but, instead, incorporated it as part of a larger optimization technique. The author has recently learned of four M.S. and Ph.D. dissertations at the Colorado School of Mines that treat the idea of successive monomial approximations as a general-purpose technique for solving systems of non-linear equations. Initially, Tulk[13] called it the 'transformed Newton's method' and formed his monomial approximation to match the partial derivative form of Duffin *et al.*[3] He observed that it outperformed Newton's method in limited numerical experiments with simple test problems. Later, Baker[14] presented the same algorithm, but expressed in terms of condensation, and reported very good numerical results with a discounted cash flow rate of return problem and an algebraic chemical equilibrium problem. Greening[15] showed that Tulk and Baker's methods were essentially the same, and proved the method would converge under certain circumstances by relating it to Newton's method. Wall[16,17] proved that the method is guaranteed to converge from any starting point for algebraic chemical equilibrium problems.

In 1987, Peterson[18] investigated the solution of a single posynomial equation in one variable. He showed that at most two solutions are possible and presented a Newton–Fourier solution technique applied to the 'logposynential' form of the equation that guarantees monotonic convergence to both solutions. The Newton part of his technique is essentially the same as the monomial method. The Fourier modification assures the monotonic convergence, but the results do not generalize to the multi-dimensional case nor to the non-posynomial case.

3. A BRIEF REVIEW OF THE MONOMIAL METHOD

The monomial method is applicable to the general class of N equations in N independent variables of the form

$$\sum_{i=1}^{T_k} \left(s_{ik} c_{ik} \prod_{j=1}^{N} x_j^{a_{ijk}} \right) = 0, \quad k = 1, \ldots, N \tag{1}$$

The coefficients, c_{ik}, are positive values, each multiplied by a sign term, s_{ik}, which has a value of either $+1$ or -1. The exponents, a_{ijk}, are real values, unrestricted in sign. The variables, x_j, are required to be positive. Any algebraic system can be cast in the form (1), but additional equations and variables may need to be introduced.[19] Without delving into details that are available elsewhere,[2] the linear system produced by the logarithm of the approximating monomial system is

$$\sum_{j=1}^{N} A_{jk} z_j = -\ln C_k, \quad k = 1, \ldots, N \tag{2}$$

The matrix A_{jk} comprises weighted sums of the exponents, a_{ijk}, of the form

$$A_{jk} = \sum_{i=1}^{T_k} s_{ik} a_{ijk} \delta_{ik} \tag{3}$$

The vector $\ln C_k$ comprises weighted sums of the coefficients, c_{ik}, of the form

$$\ln C_k = \sum_{i=1}^{T_k} s_{ik} \delta_{ik} \ln (c_{ik}/\delta_{ik}) \tag{4}$$

These two equations for A_{jk} and $\ln C_k$ are more compact than those presented previously,[2] the difference being the use of the sign quantity, s_{ik}, to combine previously separate sums.

The weights, δ_{ik}, are uniquely determined by the current operating point, $\bar{x}$. To calculate the weights, the terms of the original system (1) must first be partitioned into two groups: those with $s_{ik} = +1$ and those with $s_{ik} = -1$. We define functions P_k and Q_k to represent this partition so that the algebraic system (1) becomes $P_k - Q_k = 0$. Each of the functions P_k and Q_k is now a sum of positive terms. The weights are calculated as the fraction contributed by each term to its parent P_k or Q_k function, when evaluated at the operating point, $\bar{x}$. There is one weight corresponding to each term in the system (1).

Formally, if we let T_k^+ denote the set of positive terms of equation k and T_k^- the set of negative terms of equation k, then

$$\bar{P}_k = \sum_{i \in T_k^+} c_{ik} \prod_{j=1}^{N} \bar{x}_i^{a_{ijk}} \quad \text{and} \quad \bar{Q}_k = \sum_{i \in T_k^-} c_{ik} \prod_{j=1}^{N} \bar{x}_j^{a_{ijk}} \tag{5}$$

and

$$\delta_{ik} = \frac{c_{ik} \prod_{j=1}^{N} \bar{x}_j^{a_{ijk}}}{\bar{P}_k} \quad \text{for } i \in T_k^+ \quad \text{and} \quad \delta_{ik} = \frac{c_{ik} \prod_{j=1}^{N} \bar{x}_j^{a_{ijk}}}{\bar{Q}_k} \quad \text{for } i \in T_k^- \tag{6}$$

The overbar on P and Q indicates evaluation at the operating point, $\bar{x}$.

The solution of linear system (2), z, is a vector of transformed variables $z_j = \ln x_j$. The outcome of one iteration of the monomial method, which becomes the new operating point, is obtained by exponentiating each element of the solution vector, z.

4. SIMPLIFICATIONS

There is a considerable amount of unnecessary computation required by the constructions presented in the previous section. This section presents simplifications that reduce the effort required to construct the linear system and to make the transition from one linear system to the next.

Suppose that we identify each term of the algebraic system (1) by the symbol u_{ik}, so that

$$u_{ik} = c_{ik} \prod_{j=1}^{N} x_j^{a_{ijk}} \tag{7}$$

and $P_k = \sum_{i \in T_k^+} u_{ik}$ and $Q_k = \sum_{i \in T_k^-} u_{ik}$. If we now transform the variables with $x_j = e^{z_j}$, we have

$$u_{ik} = c_{ik} \prod_{j=1}^{N} (e^{z_j})^{a_{ijk}} = c_{ik} e^{\sum_{j=1}^{N} z_j a_{ijk}} \tag{8}$$

It is now apparent that the weights can be calculated efficiently from the transformed operating point, $\bar{z}$, which is obtained directly from the solution of the previous linear system. This avoids having to exponentiate $\bar{z}$ to get $\bar{x}$, and also avoids having to perform the many calculations of the form $x_j^{a_{ijk}}$ in equation (6), which can be relatively expensive for fractional exponents. Instead, a sum of products is exponentiated once for each term. From a computational point of view, this allows the linear system to be updated from one iteration to the next with far less calculation.

Also, the form of the linear system itself can be modified so that additional savings in computation result. It can be shown, by expanding and combining terms, that

$$\ln C_k = \ln(\bar{P}_k/\bar{Q}_k) - \sum_{j=1}^{N} A_{jk} \bar{z}_j \tag{9}$$

which, when substituted into equation (2), leads to the linear system

$$\sum_{j=1}^{N} A_{jk} \Delta z_j = -\ln(\bar{P}_k/\bar{Q}_k), \quad k = 1, \ldots, N \tag{10}$$

where $\Delta z_j = z_j - \bar{z}_j$. In this case, C_k (equation (4)) need not be computed at all to construct this linear system. Linear system (10) is more convenient to implement. but linear system (2) may still be preferred in certain cases, as discussed in the next section.

5. TWO SOLUTION ALGORITHMS

The monomial method algorithm can be performed in two different ways, depending upon the nature of the starting point used. In one case, a selection of $\bar{x}$ initiates the process; in the other case, a selection of weights, δ, initiates the process. The two algorithms operate differently in the first iteration only.

Many variants of Newton's method have been proposed to make it more robust, including such modifications as line search and step control. These modifications can be applied to the monomial method as well, but they are not addressed in this presentation.

Algorithm MM(x)
Step 1: Select an initial positive operating point, $\bar{x}$.
Step 2: Transform the starting point using $\bar{z}_j = \ln \bar{x}_j$.
Step 3: Evaluate each monomial term at $\bar{z}$ using $\bar{u}_{ik} = c_{ik} \exp(\sum_{j=1}^{N} \bar{z}_j a_{ijk})$.
Step 4: Evaluate $\bar{P}_k = \sum_{i \in T_k^+} \bar{u}_{ik}$ and $\bar{Q}_k = \sum_{i \in T_k^-} \bar{u}_{ik}$.

Step 5: Evaluate $\delta_{ik} = \bar{u}_{ik}/\bar{P}_k$ for $i \in T_k^+$ and $\delta_{ik} = \bar{u}_{ik}/\bar{Q}_k$ for $i \in T_k^-$.

Step 6: Evaluate $A_{jk} = \sum_{i=1}^{T_k} s_{ik} a_{ijk} \delta_{ik}$.

Step 7: Solve the linear system $\sum_{j=1}^{N} A_{jk} \Delta z_j = -\ln(\bar{P}_k/\bar{Q}_k)$ for the change in transformed variables, Δz.

Step 8: Calculate the new transformed operating point, $\hat{z}_j = \bar{z}_j + \Delta z_j$.

Step 9: Check for convergence, e.g. $\|\Delta z_j\| <$ tolerance and/or $\|\ln(P_k(\hat{z})/Q_k(\hat{z}))\| <$ tolerance. If so, then quit with $\bar{x} = \exp \hat{z}$ as the solution. Otherwise set $\bar{z} = \hat{z}$ and go to step 3.

Algorithm MM(δ)

Step 1: Select an initial positive weight, δ_{ik}, for each monomial term of system (1).

Step 2: Evaluate $A_{jk} = \sum_{i=1}^{T_k} s_{ik} a_{ijk} \delta_{ik}$ and $\ln C_k = \sum_{i=1}^{T_k} s_{ik} \delta_{ik} \ln(c_{ik}/\delta_{ik})$.

Step 3: Solve the linear system $\sum_{j=1}^{N} A_{jk} z_j = -\ln C_k$ for the transformed variables, z.

Step 4: Set $\bar{z} = z$ and begin the second iteration by entering Algorithm MM(x) at step 3.

The second algorithm does not require values of the independent variables to be selected to initiate the process. In some cases, more information may be known about what the relative weights of each term at a solution should be than what the values of the variables themselves at a solution should be. This knowledge may arise from an understanding of the physical system being modelled by the algebraic system. In this case, a better selection of starting point (one that is closer to a solution) can be made with Algorithm MM(δ).

When multiple solutions exist, they may be easier to identify using a carefully selected set of starting δ values rather than $\bar{x}$ values. For example, in the field of civil engineering structural optimization, it has been observed that when multiple optima exist, they each correspond to a different 'mode' of behaviour, which can be identified by patterns of dominant terms in the formulation.[20, 21] A solution with a specific mode of behaviour can be obtained by initiating Algorithm MM(δ) with an appropriate pattern of weights.

To demonstrate one iteration of Algorithm MM(x), consider the system of equations

$$\begin{aligned} 11\,664 x_1 x_2^{-4} + 54{\cdot}0 x_1^2 x_2^{-4} + 21{\cdot}6 x_1^{-2} - 10{\cdot}8 x_1^4 x_2^{-4} - 4{\cdot}32 = 0 \\ 6998{\cdot}4 x_1^4 x_2^{-7} + 18{\cdot}0 x_1^4 x_2^{-6} + 8398{\cdot}0 x_2^{-3} - 12{\cdot}96 x_1^4 x_2^{-4} - 5{\cdot}184 = 0 \end{aligned} \tag{11}$$

This problem represents the fully stressed design of a civil engineering frame structure.[2, 21] (Note that although this formulation has only integral exponents, the monomial method can treat arbitrary real-valued exponents.) The starting point is selected to be $\{\bar{x}_1 = 2,\ \bar{x}_2 = 10\}$ or $\{\bar{z}_1 = 0{\cdot}6931,\ \bar{z}_2 = 2{\cdot}303\}$. The values of each monomial term evaluated at $\bar{z}$ are $\{\bar{u}_{11} = 2{\cdot}333,\ \bar{u}_{21} = 0{\cdot}0216,\ \bar{u}_{31} = 5{\cdot}4,\ \bar{u}_{41} = 0{\cdot}01728,\ \bar{u}_{51} = 4{\cdot}32,\ \bar{u}_{12} = 0{\cdot}01120,\ \bar{u}_{22} = 0{\cdot}000288,\ \bar{u}_{32} = 8{\cdot}398,\ \bar{u}_{42} = 0{\cdot}02074,\ \bar{u}_{52} = 5{\cdot}184\}$. The positive and negative parts of each equation are $\{\bar{P}_1 = 7{\cdot}754,\ \bar{Q}_1 = 4{\cdot}337,\ \bar{P}_2 = 8{\cdot}410$ and $\bar{Q}_2 = 5{\cdot}205\}$. The fraction contributed by each term to its parent P or Q is $\{\delta_{11} = 0{\cdot}3008,\ \delta_{21} = 0{\cdot}002786,\ \delta_{31} = 0{\cdot}6964,\ \delta_{41} = 0{\cdot}003984,\ \delta_{51} = 0{\cdot}9961,\ \delta_{12} = 0{\cdot}001331,\ \delta_{22} = 0{\cdot}00003424,\ \delta_{32} = 0{\cdot}9986,\ \delta_{42} = 0{\cdot}003984$ and $\delta_{52} = 0{\cdot}9960\}$. The elements of the linear system are $\{A_{11} = -1{\cdot}102,\ A_{21} = -1{\cdot}199,\ A_{12} = -0{\cdot}01047$ and $A_{22} = -2{\cdot}989\}$ and $\{-\ln(\bar{P}_1/\bar{Q}_1) = -0{\cdot}5810$ and $-\ln(\bar{P}_2/\bar{Q}_2) = -0{\cdot}4798\}$. The solution of this linear system is $\{\Delta z_1 = 0{\cdot}3539$ and $\Delta z_2 = 0{\cdot}1593\}$. This corresponds to a new operating point of $\{\bar{z}_1 = 1{\cdot}047,\ \bar{z}_2 = 2{\cdot}462\}$ or $\{\bar{x}_1 = 2{\cdot}849,\ \bar{x}_2 = 11{\cdot}73\}$, which completes the first iteration.

To approach the issue of how much computation is required per iteration for the monomial method in comparison to Newton's method, one could attempt to count the various arithmetic operations involved. A much simpler observation will suffice, however. It is important to realize that as N grows, the computational effort required to solve the linear system rapidly dominates. Since the sparsity pattern of the A matrix will be identical to the Jacobian matrix (see equation

(23)), the computational effort required for one iteration of the monomial method and Newton's method approach one another as N grows large.

6. SPECIAL PROPERTIES

This section briefly reviews and expands on several of the unusual properties of the monomial method. The practical significance of these special properties is also discussed.

6.1. *Property 1 (self-scaling variables)*

The monomial method is 'scale invariant' to a positive diagonal transformation of variables in a stronger sense than Newton's method. As with Newton's method, if one iteration of the monomial method is performed in the space of the transformed variables, the outcome, when transformed back, will be identical to the outcome of one iteration without any scaling. Yet, scaling features are often found in Newton solvers, even though the scale invariance predicts that they will have no effect. The reason for the scaling is that Newton's method is scale invariant only when *exact arithmetic* is used and the same pivot sequence is selected during the linear solution process.[22] The scaling process will greatly alter the values of the individual elements of the linear system and will generally give rise to a different pivoting sequence. A poor choice of pivots may cause a substantial propagation of errors with finite-precision computations. To improve pivot selections, it is often recommended in practice that (1) the variables be scaled so that 'typical' values of each transformed variable are roughly equal, and (2) the equations of the linear system be 'equilibrated' so that the maximum absolute value of the coefficients of each row are roughly equal.

With the monomial method, the individual elements of the A matrix are *unchanged* by a scaling of the variables. This property overrides and nullifies any attempt to precondition the linear system by scaling the variables.

6.2. *Property 2 (self-equilibrating equations)*

The individual elements of the A matrix are bounded by

$$\min_{i\in T_k^+} a_{ijk} - \max_{i\in T_k^-} a_{ijk} \leqslant A_{jk} \leqslant \max_{i\in T_k^+} a_{ijk} - \min_{i\in T_k^-} a_{ijk} \tag{12}$$

and the row sums of the A matrix are bounded by

$$\min_{i\in T_k^+} n_{ik} - \max_{i\in T_k^-} n_{ik} \leqslant \sum_j A_{jk} \leqslant \max_{i\in T_k^+} n_{ik} - \min_{i\in T_k^-} n_{ik} \tag{13}$$

where n_{ik} is the homogeneity degree of term i of equation k. Since the range of exponent values found in typical systems of equations rarely spans more than an order of magnitude, it is apparent that this property equilibrates the linear system automatically.

The combination of Properties 1 and 2 yields a built-in self-scaling of the variables and equilibrating of the equations that 'floats' with the current operating point. For certain problems, these properties have been shown to dramatically improve the conditioning of the linear system produced by the monomial method in comparison to Newton's method.[2]

6.3. *Property 3 (factor invariance)*

The monomial method is completely invariant to the operation of multiplying any one or more of the non-linear equations in the original system by any one or more posynomial functions. All

elements of the linear system produced by the monomial method are unchanged by this operation; the sequence of iterates generated from any arbitrary starting point is unaffected.

As a special case, this property overrides and nullifies any attempt to scale each equation by a constant or by any power of x, rendering useless any attempt to improve performance by putting the system in 'standard form'. Contrast this to Newton's method, where multiplication of the equations by posynomials will generally introduce new solutions and/or singularities and significantly affect the numerical performance.

It has recently been discovered that this property must be qualified to a certain extent. When multiplying by a positive polynomial, no cancellation or combination of terms must take place between P and Q, otherwise the linear system may change as a result of the operation. For example, if the original equation is $x - x^{-1} = 0$ and it is multiplied by the posynomial $x^2 + 2$, the result is $(x^3 + 2x) - (x + 2x^{-1}) = 0$ or, after cancelling like terms, $x^3 + x - 2x^{-1} = 0$. The monomial method linear system is unchanged in the first case, i.e. $P = x^3 + 2x$ and $Q = x + 2x^{-1}$, but will be different in the second case, i.e. $P = x^3 + x$ and $Q = 2x^{-1}$. Note that this qualification does not impact the issue of putting the system into standard form because multiplying by simple powers of x will not cause like terms of opposite sign to appear.

6.4. *Property 4 (proportion invariance)*

A set of operating points that yield a common set of weights, δ (i.e. that preserve the proportionality of terms in the original non-linear system), has the property that all elements of the set generate the same sequence of iterates under the action of the monomial method.

As a special case, if the positive terms, P, and negative terms, Q, of each equation of the original non-linear system are individually homogeneous, to any degree of homogeneity (possibly different for each P and Q), then a set of operating points that form an origin-based ray will step to a common single point in one iteration of the monomial method, and this common point will be located at the intersection of the monomial surfaces that are each tangent to their corresponding $P - Q = 0$ surfaces at the point where the origin-based ray pierces the $P - Q = 0$ surface. Since intersections of the $P - Q = 0$ surfaces are where all solutions exist, starting points very distant from these surfaces will be drawn to them very rapidly in one iteration.

For systems that are not homogeneous, experiments show that the first iteration often exhibits this quick convergence to the vicinity of the $P - Q = 0$ surfaces anyway. This is perhaps due to the observation that for starting points distant from the origin, the highest-order terms in the equations dominate; conversely, for starting points very near the origin, the lowest-order terms dominate. This implies that the corresponding δ_{ik} will dwarf the others, and the system of equations will appear to the monomial method as being effectively homogeneous.

A simple matrix construction will determine whether or not there are sets of operating points that yield identical weights. Consider the exponent matrix, E, containing the exponents of each variable (one variable per column) in each term (one term per row). For the example presented in Section 5,

$$\begin{aligned} P_1 &= 11\,664 x_1 x_2^{-4} + 54{\cdot}0 x_1^2 x_2^{-4} + 21{\cdot}6 x_1^{-2} \\ Q_1 &= 10{\cdot}8 x_1^4 x_2^{-4} + 4{\cdot}32 \\ P_2 &= 6998{\cdot}4 x_1^4 x_2^{-7} + 18{\cdot}0 x_1^4 x_2^{-6} + 8398{\cdot}0 x_2^{-3} \\ Q_2 &= 12{\cdot}96 x_1^4 x_2^{-4} + 5{\cdot}184 \end{aligned} \tag{14}$$

the exponent matrix is

$$E = \begin{bmatrix} 1 & -4 \\ 2 & -4 \\ -2 & 0 \\ \hline 4 & -4 \\ 0 & 0 \\ \hline 4 & -7 \\ 4 & -6 \\ 0 & -3 \\ \hline 4 & -4 \\ 0 & 0 \end{bmatrix} \begin{matrix} > \\ > \\ \\ > \\ \\ > \\ > \\ \\ > \\ \\ \end{matrix} \begin{matrix} -1 & 0 \\ 4 & -4 \\ \\ 4 & -4 \\ \\ 0 & -1 \\ 4 & -3 \\ \\ 4 & -4 \\ \\ \end{matrix}, \quad S = \begin{bmatrix} -1 & 0 \\ 4 & -4 \\ 4 & -4 \\ 0 & -1 \\ 4 & -3 \\ 4 & -4 \end{bmatrix} \tag{15}$$

A solid line separates each equation and a dotted line separates the positive and negative terms in each equation. Within each group of terms of like sign, a pairwise difference of rows is calculated. This forms what is called the S matrix. It was shown previously that if the rank of S is less than the number of columns (number of variables), then sets of starting points producing common weights exist.[2] In this case, it is obvious that the rank of S is 2. Consequently, every operating point gives rise to a unique set of weights.

Even though the S matrix is not rank deficient in this case, the first iteration of this system still demonstrates the extremely rapid movement toward the solutions as described for the homogeneous case. To demonstrate this, a graphical construction is presented in Figure 1 that shows the mapping of a checkerboard of starting points on the x_1–x_2 plane spanning a region $0{\cdot}0 \leqslant x_1 \leqslant 900{\cdot}0$ and $0{\cdot}0 \leqslant x_2 \leqslant 600{\cdot}0$ (Figure 1(a)) under the action of one iteration of Newton's method (Figure 1(b)) and one iteration of the monomial method (Figure 1(c)). All four solutions of this non-linear system fall within the tiny rectangular region shown in the lower left-hand corner of Figure 1(a). Figure 1(b) shows that Newton's method attracts some regions toward a solution but also expels other regions that will eventually encounter a floating point overflow condition. Contrast this to Figure 1(c), which shows the extremely rapid convergence in the first iteration that is characteristic of the monomial method, even though the functions are not homogeneous, nor do they possess sets of starting points that preserve proportionality of terms. The entire checkerboard of starting points is mapped to the tiny cluster of dots shown in the lower left-hand corner of Figure 1(c). Further investigation is needed to identify what types of non-linear systems give rise to this behaviour generally.

7. ALTERNATE FORMS

The linear system generated at each iteration of the monomial method is

$$\sum_{j=1}^{N} A_{jk} \Delta z_j = -\ln(\bar{P}_k/\bar{Q}_k) \tag{16}$$

The elements of the A matrix are weighted sums of the exponents, a_{ijk}, appearing in the original algebraic system $P - Q = 0$. Several authors have noted that an identical A matrix results from expanding the logarithm of each posynomial by first-order Taylor series in terms of the variables

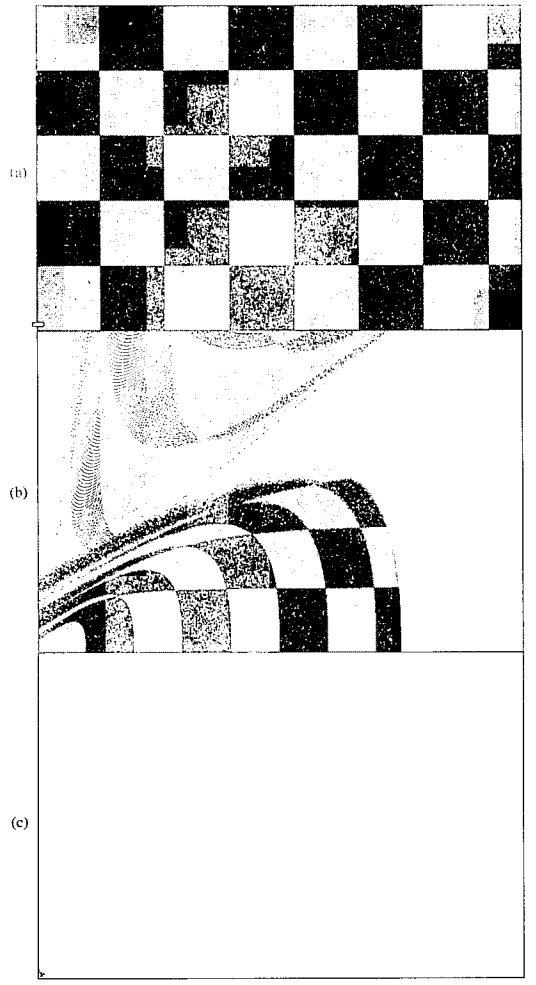

Figure 1. The mapping of a checkerboard of starting points (a) produced by one iteration of Newton's method (b) and one iteration of the monomial method (c) applied to the example problem (14)

$z_j = \ln x_j$ (see References 3, 11, 13, 15 and Lemma 4.3 in 23). The linear system can therefore be expressed in terms of partial derivatives of P and Q:

$$\sum_{j=1}^{N} \left(\frac{\bar{x}_j}{\bar{P}_k} \frac{\partial P_k}{\partial x_j}\bigg|_{\bar{x}} - \frac{\bar{x}_j}{\bar{Q}_k} \frac{\partial Q_k}{\partial x_j}\bigg|_{\bar{x}} \right) \Delta z_j = -\ln(\bar{P}_k/\bar{Q}_k) \tag{17}$$

This form is useful when the system being solved is not algebraic in form, but when positive and negative parts can be identified, e.g. $x^2 + 2 - e^x = 0$. Also, this alternate form may be useful when the system of equations being solved is not known explicitly, but when function evaluations can be performed to approximate the partial derivatives by finite differencing. A major difficulty is that the functions must be separated into strictly positive and strictly negative parts, which may not be possible when the system is now known in explicit form.

A third form of the monomial method linear system results when the system of equations, $P - Q = 0$, is recast as $R = 1$, where $R = P/Q$. By virtue of the rules of differentiation of a rational

function, the linear system (17) is equivalent to

$$\sum_{j=1}^{N} \frac{\bar{x}_j}{\bar{R}_k} \frac{\partial R_k}{\partial x_j}\bigg|_{\bar{x}} \Delta z_j = -\ln \bar{R}_k \tag{18}$$

Instead of finding simultaneous zeros of $P - Q$, we are finding simultaneous 'ones' of the strictly positive function R. This form suggests another solution strategy for general systems (including non-algebraic systems) of the form $F(x) = 0$. If we exponentiate this system, we have $e^F = 1$, which is positive for all values of F. Upon substituting $R = e^F$ into (18), we obtain

$$\sum_{j=1}^{N} \bar{x}_j \frac{\partial F_k}{\partial x_j}\bigg|_{\bar{x}} \Delta z_j = -\bar{F}_k \tag{19}$$

System (19) is very similar, but not identical, to Newton's method applied to $F(x) = 0$. Limited numerical experiments have been conducted using (19) applied to algebraic systems of the form $F = P - Q$. When applied to the example problem (14), the performance of (19) is found to be better than Newton's method but much worse than the monomial method. It is conjectured that there is something inherent in the form $P/Q = 1$ which is exploited by the monomial method, but which is absent in the form $e^{P-Q} = 1$.

One final form of linear system (16) is interesting because it relates the monomial method to Newton's method. Consider another recasting of the system $P - Q = 0$, obtained by taking the logarithm of $P/Q = 1$, which yields $\ln(P/Q) = 0$. If we identify this new form as $G(z) = 0$, expressing G as a function of the transformed variables $z = \ln x$, then the monomial method linear system is equivalent to

$$\sum_{j=1}^{N} \frac{\partial G_k}{\partial z_j}\bigg|_{\bar{z}} \Delta z_j = -\bar{G}_k \tag{20}$$

It is apparent that (20) is simply Newton's method applied to $G(z)$. Thus, the monomial method can be interpreted as Newton's method applied to the logarithm of the rational function P/Q, operated in the space of the variables $z = \ln x$. Although this observation is useful for understanding some theoretical aspects of the behaviour of the monomial method, the linear system (16) should be preferred in practice because it treats the system $P - Q = 0$ directly, without having to recast the equations, transform the variables, and then take derivatives. On the other hand, if a computer program that implements Newton's method is readily available, the monomial method can be performed without additional programming by reformulating the problem as $\ln(P_k(e^{z_j})/Q_k(e^{z_j})) = 0$ and using the Newton solver. Another advantage of using existing solver programs is that they often contain step control features that might enhance the performance of the monomial method in situations where it misbehaves, such as near singularities of the A matrix.

8. THE THREE-PART REFORMULATION

It was shown in the previous section that the monomial method can be viewed as Newton's method applied to a reformulation of the original algebraic system. The reformulation has three basic parts: a conversion to rational form, $P/Q = 1$, to which a logarithm is applied to give $\ln(P/Q) = 0$, and an exponential transformation of variables, $x = e^z$. This section investigates whether or not one portion of the reformulation is more responsible than the others for the enhanced performance of the monomial method. To investigate this question, graphical constructions were generated. Figure 2 presents 'basins of attraction' constructions for the example

problem (14) for the following formulation variations:

(a) Newton's method applied to $P(x) - Q(x) = 0$ (the standard treatment)
(b) Newton's method applied to $P(x)/Q(x) - 1 = 0$
(c) Newton's method applied to $\ln(P(x)/Q(x)) = 0$
(d) Newton's method applied to $P(e^z) - Q(e^z) = 0$

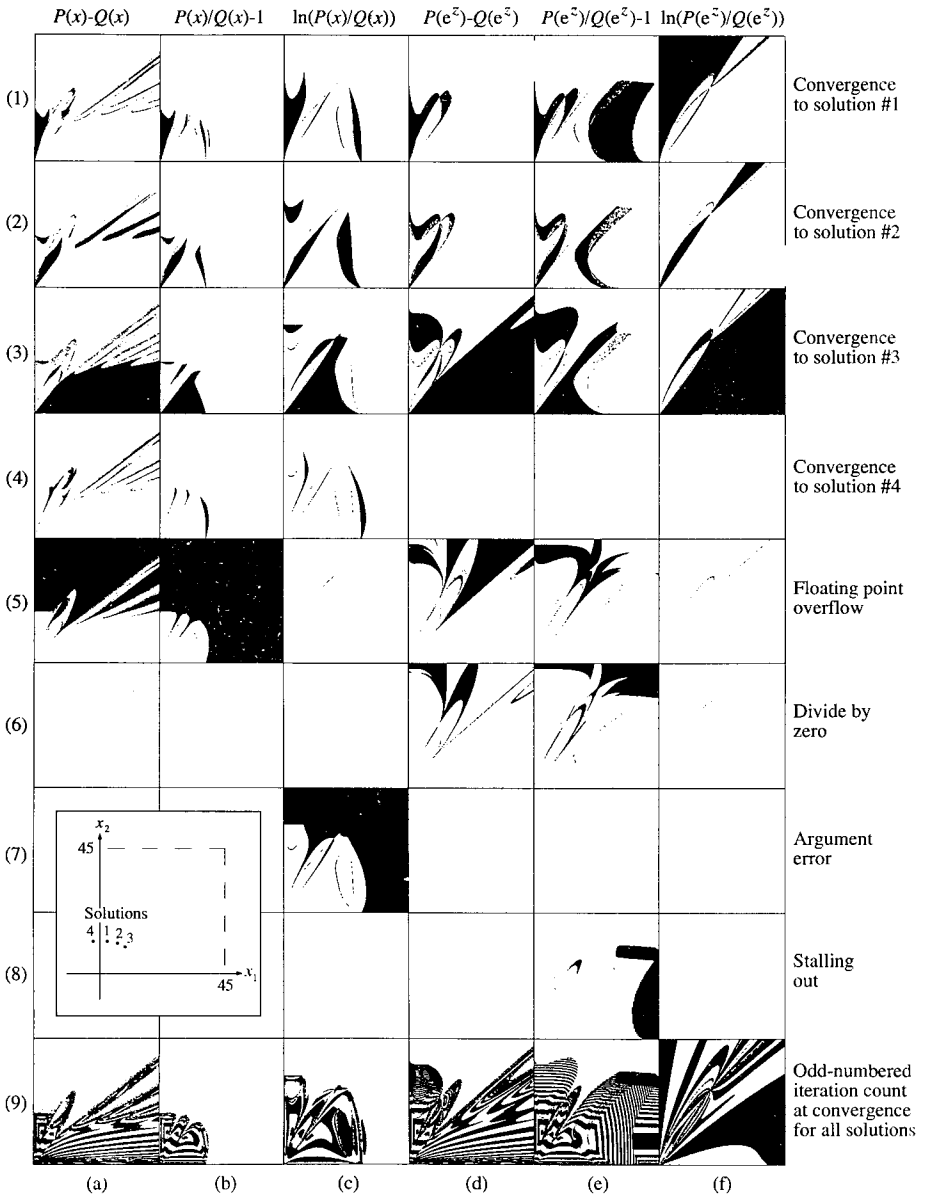

Figure 2. An array of basin of attraction constructions for Newton's method applied to six different formulations (columns a–f) of the example problem (14)

(e) Newton's method applied to $P(e^z)/Q(e^z) - 1 = 0$
(f) Newton's method applied to $\ln(P(e^z)/Q(e^z)) = 0$ (the monomial method)

Figure 2 is divided into an array of nine rows and six columns of graphical images. Each column of the array corresponds to one of the formulation variations above. Each row of the array corresponds to a different terminating event, i.e. the condition that caused the Newton iteration sequence to terminate. The first four rows are for normal termination—convergence to one of the four solutions. The next three rows are for floating point exception conditions, either overflow, divide by zero, or an argument error (in this case, a log taken of a negative number). The eighth row is for when movement of the operating point from one iteration to the next is less than a specified distance (0·001 in this case); this terminating condition is called 'stalling out'. The final row shows all of the starting points that converged to a solution, within a tolerance of 0·1, in an odd number of iterations. The widths of the resulting black and white bands give an indication of how fast convergence occurred.

Each image in Figure 2 is identified by a row number (1–9) and column letter (a–f). Image (n, α) depicts the set of starting points (in black) that led to the terminating event n when Newton's method was applied to formulation α. All images represent a tightly spaced lattice of 300×300 starting points spanning the portion of the design space bounded by $0{\cdot}0 \leqslant x_1 \leqslant 45{\cdot}0$ and $0{\cdot}0 \leqslant x_2 \leqslant 45{\cdot}0$. Column a is the standard Newton's method treatment of this problem; column f is the monomial method.

A very simple implementation of Newton's method is used to create all images. The 2×2 Jacobian matrix is inverted in closed form and multiplied by the vector of negative function values to get an explicit formula for the Newton step. No step control is performed. The four solutions are $\{(2{\cdot}92, 11{\cdot}72), (6{\cdot}45, 11{\cdot}15), (9{\cdot}12, 9{\cdot}71)$ and $(-1{\cdot}98, 11{\cdot}74)\}$. The calculations were performed on a Macintosh Quadra 700 under System 7 using single precision and version 2·4 of the Absoft MacFortran compiler.

Image (5, b) in Figure 2 indicates that the rational reformulation alone is detrimental to the performance for this particular problem. A much larger percentage of starting points encounter a floating point overflow condition. Taking the logarithm of this rational form causes a slight improvement in convergence near the solutions (images (1, c)–(4, c)), but another problem has appeared. Whenever an iterate steps into a region where P/Q is negative, an argument error occurs on the next iteration. This happens for nearly 80 per cent of the starting points (image (7, c)). Notice, however, that the speed of convergence is greatly improved by taking a logarithm, as evidenced by the wider bands in image (9, c).

The exponential transformation of variables, with or without the rational reformulation, gives rise to relatively little difference in performance. Divide-by-zero terminations are much more frequent and an interesting region of sensitive dependence on initial conditions has appeared in most images in column e (the speckled region). The banding in images (9, d) and (9, e) indicates that the exponential transformation does not seem to affect the speed of convergence. Instead, one of the most obvious effects of the exponential transformation is that no starting points converge to solution 4 since it has a negative component, one that is unrealizable for any value of z.

The author had previously claimed that because negative solutions were not attainable by the monomial method, there would be no corresponding basins of attraction for them.[2] Image (8, e) refutes this claim. Even though solution 4 cannot be achieved, its influence is still felt, causing a large number of starting points to be attracted to the point nearest to solution 4 in the positive quadrant, which is located at $(0+, 11{\cdot}74)$. Interestingly, this phantom solution does not have a large basin of attraction surrounding it, but instead strongly attracts a distant set of starting points which jump to near the origin on the first iteration and then are attracted up the x_2 axis until they reach the phantom attractor.

It appears that a combination of all three parts of the reformulation is necessary to produce the strikingly improved performance observed in column f. As others have noted, the combined action of the logarithm and the exponential transformation of variables seems to extract as much linearity out of the posynomial as possible.[15, 18] We see this more clearly through

$$\ln P_k = \ln \sum_{i=1}^{T_k} \left(c_{ik} \prod_{j=1}^{N} x_j^{a_{ijk}} \right) = \ln \sum_{i=1}^{T_k} \left(c_{ik} \prod_{j=1}^{N} (e^{z_j})^{a_{ijk}} \right) = \ln \sum_{i=1}^{T_k} e^{(\ln c_{ik} + \sum_j z_j a_{ijk})} \tag{21}$$

where the logarithm and exponential are separated only by a summation. The rational reformulation of the positive and negative terms as $P/Q = 1$, prior to taking the logarithm, maintains this near linearity as the difference of two nearly linear constructions:

$$\ln \frac{P_k}{Q_k} = \ln \sum_{i \in T_k^+} e^{(\ln c_{ik} + \sum_j z_j a_{ijk})} - \ln \sum_{i \in T_k^-} e^{(\ln c_{ik} + \sum_j z_j a_{ijk})} \tag{22}$$

The combined action of the logarithm and exponential is crucial to the performance characteristics of the monomial method in another respect. Note that by putting equation (17) in matrix form, the monomial method linear system may be expressed in terms of the Jacobian matrices of P and Q, J_P and J_Q, respectively, as

$$[D_{\bar{P}}^{-1} \bar{J}_P D_{\bar{X}} - D_{\bar{Q}}^{-1} \bar{J}_Q D_{\bar{X}}] \{\Delta \ln X\} = \{ - \ln (\bar{P}/\bar{Q})\} \tag{23}$$

where the Jacobian matrices are premultiplied and postmultiplied by diagonal matrices of inverse function values and operating point values, respectively. These diagonal matrices are equivalent to the process of scaling and equilibrating the Newton linear system as discussed in Section 6 to improve pivot selections. The premultiplying diagonal matrix, which is responsible for equilibrating the rows, is a direct result of the logarithm since $\mathrm{d}(\ln f)/\mathrm{d}x = (1/f)/\mathrm{d}f/\mathrm{d}x$. The postmultiplying diagonal matrix, which is responsible for scaling the variables, is a direct result of the exponential transformation of the variables, $x = e^z$, since $\mathrm{d}f/\mathrm{d}z = x\,\mathrm{d}f/\mathrm{d}x$. Thus, the combined action of the logarithm and exponential in the reformulation is responsible for the floating self-conditioning of the linear system that was discussed in Section 6.

9. DIFFICULTIES OBSERVED WITH THE MONOMIAL METHOD

The basins of attraction images of Figure 2 demonstrate that the monomial method dramatically outperforms Newton's method for the specific system of equations under consideration. This section presents another system of equations for which the monomial method performs very poorly. The system of equations is simple: the intersection of two circles

$$\begin{aligned} (x-1)^2 + (y-1)^2 &= 1 \\ (x-2)^2 + (y-2)^2 &= 1 \end{aligned} \tag{24}$$

Equations of this type are common in applications such as mechanical engineering mechanism design.

Figure 3 presents the basins of attraction constructions for this problem, presented in a fashion similar to Figure 2. Each image represents an array of 300×300 starting points spanning the region $0{\cdot}0 \leqslant x, y \leqslant 5{\cdot}0$. There are two solutions, (1, 2) and (2, 1), which have corresponding basins of attraction in rows 1 and 2 of this figure. Rows 3 and 4 present the starting points that encounter a floating point error condition that terminates the solution process. Clearly, the monomial method is unsuitable for this problem.

One explanation for the poor performance of the monomial method is that both equations have an identical exponent structure. The performance of the monomial method is closely tied to

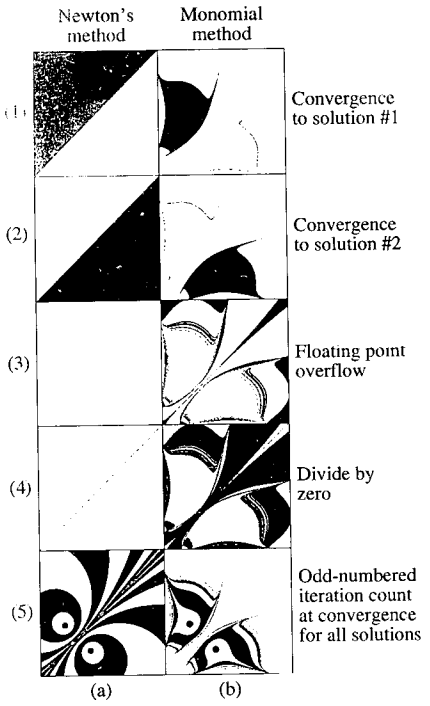

Figure 3. An array of basin of attraction constructions for Newton's method (column a) and the monomial method (column b) applied to the problem (24)

the exponent structure of the system of equations since the coefficients of the linear system are weighted sums of the exponents. When two equations have an identical exponent structure, the corresponding rows of the linear system will have a tendency to be linearly dependent. The only hope of breaking this linear dependence is with the weights, which differ only because the coefficients in the non-linear system differ. Apparently, the iterates in this example are being drawn into a region where the weights are unable to adequately break the linear dependence, leading to extreme ill-conditioning in the linear system and subsequent numerical difficulties.

This poor performance seems to extend to 'translation systems' in general, i.e. when two or more equations differ only as a result of a translating change of co-ordinates of the form $x'_j = x_j + \alpha_j$. This same phenomenon is present in systems of linear equations; a linear system becomes singular when two equations are translations of one another (parallel). Equation (22) indicates that the monomial method is attempting to extract as much linearity out of the non-linear system as it can. In doing so, it is apparently sensitive to phenomena analogous to linear dependence in the non-linear system.

10. CONCLUSIONS

The new algorithms for the monomial method have been presented. They are simpler and computationally more efficient to implement in comparison to the original algorithm. Several variations have also been presented which apply to a wider class of problems. The monomial method was shown to be equivalent to a Newton treatment of a three-part reformulation, and the role that each part plays in improving performance was investigated.

The monomial method has performed exceptionally well when applied to a structural engineering design problem. Other investigators have reported excellent performance with systems related to chemical equilibria and economics. In contrast, it was shown that the monomial method performs very poorly with certain problems related to mechanism design. It is well known that no one technique is optimal for solving all types of non-linear systems, and the monomial method is no exception. It remains an open question to determine what characteristics a system of equations must have in general for the monomial method to perform well.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. DDM-8957410, and a grant from Apple Computer, Inc.

REFERENCES

1. S. A. Burns, 'Graphical representation of design optimization processes', *Comput. Aided Design*, **21**, 21–24 (1989).
2. S. A. Burns and A. Locascio, 'A monomial-based method for solving systems of nonlinear algebraic equations', *Int. j. numer. methods eng.*, **31**, 1295–1318 (1991).
3. R. J. Duffin, E. L. Peterson and C. M. Zener, *Geometric Programming*, Wiley, New York, 1967, p. 98.
4. R. J. Duffin, 'Linearizing geometric programs', *SIAM Rev.*, **12**, 211–227 (1970).
5. E. Beckenbach and R. Bellman, *Inequalities*, Springer, Berlin, 1961.
6. G. Hardy, J. Littlewood and G. Polya, *Inequalities*, Cambridge University Press, England, 1959.
7. M. Avriel, R. Dembo and U. Passy, 'Solution of generalized geometric programs', *Int. j. numer. methods. eng.*, **9**, 149–168 (1975).
8. M. Avriel and A. Williams, 'Complementary geometric programming', *SIAM J. Appl. Math.*, **19**, 125–141 (1970)
9. L. Pascual and A. Ben-Israel, 'Constrained maximization of posynomials by geometric programming', *J. Opt. Theory Appl.*, **5**, 73–86 (1970).
10. U. Passy, 'Generalized weighted mean programming', *SIAM J. Appl. Math.*, **20**, 763–778 (1971).
11. S. A. Burns, 'Generalized geometric programming with many equality constraints', *Int. j. numer. methods eng.*, **24**, 725–741 (1987).
12. M. J. Rijckaert and X. M. Martens, 'A condensation method for generalized geometric programming', *Math. Prog.*, **11**, 89–93 (1976).
13. R. K. Tulk, 'Transformed Newton's method', *Ph.D. Thesis*, Colorado School of Mines, Golden, CO, 1976.
14. D. W. Baker, 'Application of the geometric inequality to the solution of systems of nonlinear equations', *Ph.D. Thesis*, Colorado School of Mines, Golden, CO, 1980.
15. D. R. Greening, 'A proof of convergence for a condensation approach to the solution of nonlinear equations', *M.S. Thesis*, Colorado School of Mines, Golden, CO, 1982.
16. T. W. Wall, 'A numerical algorithm for the solution of chemical equilibrium problems', *M.S. Thesis*, Colorado School of Mines, Golden, CO, 1984.
17. T. W. Wall, D. Greening and R. E. D. Woolsey, 'Solving complex chemical equilibria using a geometric-programming based technique', *Oper. Res.*, **34**, 345–355 (1986).
18. E. L. Peterson, 'The solution of posynomial equations', *Technical Report #010887-01*, Department of Mathematics and Graduate Program in Operations Research, North Carolina State University, Raleigh, NC, 1987.
19. R. J. Duffin and E. L. Peterson, 'Geometric programming with signomials', *J. Opt. Theory Appl.*, **11**, 3–35 (1973).
20. S. A. Burns and M. A. Raschke, 'Performance characteristics of two fully-stressed design algorithms for structural optimization of frame structures', *Proc. 1993 NSF Design and Manufac. Systems Conf.*, University of North Carolina, Charlotte, NC, 1993.
21. N. Khachaturian and B. Horowitz, 'Properties of optimal structures', *Proc. Symp. on Applications of Computer Methods in Engineering*, University of Southern California, Los Angeles, CA, 1977, pp. 533–542.
22. G. Dahlquist and Å. Björck, *Numerical Methods*, Prentice-Hall, Englewood Cliffs, N.J., 1974, p. 181.
23. U. Passy, 'Condensing generalized polynomials', *J. Opt. Theory Appl.*, **9**, 221–237 (1972).