Polynomial Scaling

Sadegh Ghader Panah

A Thesis
in
the Department
of
Computer Science

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montréal, Quebéc, Canada

May 1988

ISBN 0-315-44850-4

ABSTRACT

Polynomial Scaling

Sadegh Ghader Panah

A wide variation in the magnitude of coefficients of polynomials may be a source of computational problems in root-finding algorithms, as the floating-point arithmetic operations on such coefficients may render floating-point overflow or underflow. This thesis presents a new discrete method for (real) polynomial scaling; specifically, it determines a scale factor which minimizes variation in the magnitude of the coefficients of real polynomials. The method is conceptually simple and is easy to implement. It is based on the phenomenon that the scale factor coincides with the intersection of certain monomials, defined by the respective magnitudes of the nonzero terms of the polynomial to be scaled. A constructive description of this phenomenon is presented. The method compares favorably with an existing general mathematical programming approach. Results on the effect of scaling polynomials on the numerical quality of their approximate roots are presented; these results show that the effect, if any, is insignificant.

ACKNOWLEDGEMENTS

This work owes its inception to the late Professor K. V. Leung, to whom I owe a tremendous debt of gratitude.

I would like to gratefully acknowledge the decisive influence of Professor T. D. Bui on my interest in the field of Numerical Analysis.

I would also like to express my appreciation to Professor Terrill Fancott, whose continual encouragement has been a major source of motivation during the completion of this thesis.

My deepest thanks to Professor Stan L. Klasa, who not only accepted to supervise my thesis but also contributed the presentation of the m-, M- and F-algorithms.

Special thanks are due to Mr. Larry Thiel, whose powerful text formatting utility package, SYMSET, has made the typing of the thesis a transparent task.

As well, I would like to extend my thanks to Dr. Youcef Saad and the refrees of the SIAM Journal of Numerical Analysis, for their valuable feedback.

Finally, I am grateful to my friend, Margaret Long, for her useful suggestions in the writing of this thesis.

Table of Contents

LIST OF TABLES

1. INTRODUCTION

Floating-point arithmetic operations involving values with wide variation in their magnitudes may render results outside the floating-point range provided by the computer in use. Furthermore, floating-point arithmetic operations on such values might yield few significant figures of accuracy, if any. One source where such results are likely to occur is in the application of root-finding methods (particularly those involving calculation of the geratest common divisors of two polynomials) to polynomials whose coefficients vary widely in magnitude. Therefore, in order to suppress floating-point overflow/underflow and round-off error, it is desirable to supplement root-finding methods with efficient algorithms for scaling down the variation of the magnitudes of the coefficients of such polynomials, or simply for scaling of such polynomials, before approximating their roots. One root-finding method of which the initial scaling of the polynomial is an important feature, is a composite method [1] which is based on polynomial factorization by means of Euclid's algorithm. Although it is concluded in [1] that scaling of polynomial coefficients greatly improves the efficiency of the factoring process in the composite method, little attention is paid to the efficiency of the scaling process itself.

The primary object of this thesis is to present a new and efficient discrete method for scaling of real polynomials; specifically, the method determines an optimal scale factor which minimizes variation in the magnitude of the coefficients of real polynomials. The method is conceptually simple, and easy to implement. It is essentially based on the phenomenon that the optimal scale factor coincides with the intersection of certain monomials, which are defined by the respective magnitudes of the nonzero terms of the polynomial to be scaled. This method compares favorably, in terms of the complexity (in the sense of the work involved), storage, operation count, and numerical stability, with the general mathematical programming approach of [1].

The secondary object of this thesis is to investigate numerically the effect of scaling of real polynomials on their approximate roots obtained by general root-finding methods which do not involve calculation of the greatest common divisors of two polynomials. The Laguerre root-finding method is chosen in this investigation. The numerical results indicate an insignificant effect (if any) of scaling on the numerical quality of the approximate roots.

The remainder of this thesis is organized as follows. The problem is formulated in Section 2, and the essence of the scaling method is specified in Section 3. A remark on a pertinent notation is made in Section 4. Based on a constructive proof of the phenomenon mentioned, three algorithms are presented in Sections 5-7, respectively, which together describe a discrete process for determining the optimal scale factor. A comparison between the method of this thesis and that of [1] is presented in Section 8. Also, some results on the effect of scaling real polynomials on the numerical quality of their approximate roots are provided in Section 9. Finally, some concluding remarks are made in Section 10.

2. PROBLEM DEFINITION

Let P(z) be a polynomial of degree n,

(2.1) $$P(z) = a_n z^n + a_{n-1} z^{n-1} + \dots + a_0 \qquad a_n a_0 \neq 0,$$

with real coefficients $\{a_k\}$, and with real or complex roots $z_1,\cdots,z_n$. In addition, let $K=\{k : a_k \neq 0\}$ with $N=|K|$ denoting the cardinality of K. Thus, $2 \leq N \leq n+1$.

Define the variation $\upsilon(P)$ of P(z) by

$$\upsilon(P) = \log\left(\frac{\max\ c_k}{\min\ c_k}\right) \qquad : \quad k \in K$$

where

(2.2) $$c_k = |a_k| \qquad k \in K$$

And finally, let the scaled polynomial Q(s,P) associated with P(z) be given by:

(2.3) $$Q(s,P) = P(sx) = \sum_{k=0}^{n} a_k s^k x^k \qquad s \in (0\ ,\infty)$$

with the parameter s referred to, henceforth, as a scale factor.

The object of this work is to present a discrete method for solving the following minimization problem:

(2.4) $$\text{Minimize: } \upsilon(Q) = \log\left(\frac{\max\ c_k s^k}{\min\ c_k s^k}\right) \qquad \text{subject to: } s \in (0\ ,\infty), \quad k \in K$$

The solution s* of (2.4) shall henceforth be referred to as the optimal scale factor.

3. ESSENCE OF THE METHOD

Consider the monomials

(3.1) $$W_k(s) = c_k s^k \qquad s\in(0,\infty), \quad k\in K$$

and let S_{ij} denote the nonzero intersection of the monomials $W_i(s)$ and $W_j(s)$, which is given by

(3.2) $$S_{ij} = \left(\frac{c_i}{c_j}\right)^{\frac{1}{j-i}} \qquad i<j, \quad i,j\in K$$

and which shall henceforth be referred to as a critical point. In addition, let the minimization problem (2.4) be rewritten in the form:

(3.3) $$\text{Minimize:} \quad \upsilon(Q) = \log \frac{M(s)}{m(s)} \qquad \text{subject to:} \quad s\in(0,\infty)$$

where,

(3.4) $$M(s) = \max\,\{W_k(s)\} \qquad s\in(0,\infty), \quad k\in K$$

and

(3.5) $$m(s) = \min\,\{W_k(s)\} \qquad s\in(0,\infty), \quad k\in K$$

Evidently, solving (3.3) involves considerations on how to handle the presence of the generally defined functions m(s) and M(s). One may consider, for example, to relax the explicit dependence of (3.3) on m(s) and M(s), by converting (3.3) into an equivalent nonlinear mathematical programming problem [1] of the form:

(3.6) $$\text{Minimize:} \quad \log \frac{t}{d} \quad \text{subject to:} \quad t\geq c_k s^k, \quad d\leq c_k s^k, \quad k\in K, \quad s>0, \quad d>0$$

in which the relations $t \geq M(s)$ and $d \leq m(s)$ are implicitly satisfied by virtue of (3.4) and (3.5), respectively. The nonlinear problem (3.6) may readily be transformed into an equivalent linear mathematical programming problem [1] of the form:

(3.7) $\quad$ Minimize $T-D$ $\quad$ subject to: $\quad T-kS \geq C_k, \quad D-kS \leq C_k$ $\quad$ and $\quad k \in K$

where $C_k = \log c_k$, $k \in K$, represent the constants, and where $T = \log t$, $D = \log d$, $S = \log s$, which are evidently unrestricted in sign, represent the independent variables of this problem. By applying a simple transformation to the independent variables, this problem may be solved (for $\log s^*$) by a linear programming method, such as that of the revised simplex method. The purpose of this work is to offer an alternative method for solving (3.3), which is substantially more efficient than the simplex approach in terms of the complexity, storage, operation count and numerical stability.

The essence of the present method is to utilize the phenomenon that the solution s^* of the problem (3.3), or equivalently of the problem:

(3.8) $\quad$ Minimize: $\quad f(s) = \dfrac{M(s)}{m(s)}$ $\quad$ subject to: $\quad s \in (0, \infty)$

coincides with one of the critical points S_{ij}, $i=0,\cdots,N-1$, $j=i+1,\cdots,N$, given by (3.2).

An immediate implication of this phenomenon is a direct but inefficient approach for determining the optimal scale factor s^*, as follows: simply determine the $N(N-1)/2$ critical points S_{ij}, and choose as s^* the critical point at which $v(Q)$ in (2.4) is minimized. Although already more efficient than the application of the simplex process to (3.7), this direct approach may be improved considerably by further exploiting the nature of the phenomenon, to explicitly characterize the functions m(s) and M(s) over $(0, \infty)$, and hence to explicitly minimize f(s) over $(0, \infty)$. The objective function f(s) in (3.8) shall henceforth be referred to as the scale function. In Sections 4-7 below, the improved approach is described in terms of a constructive proof of the phenomenon mentioned; the proof makes use of the following obvious relations:

$$\begin{cases} W_i(s) < W_j(s) & \text{if } s > S_{ij} \\ W_i(s) = W_j(s) & \text{if } s = S_{ij} \\ W_i(s) > W_j(s) & \text{if } s < S_{ij} \end{cases} \qquad i<j, \quad i,j \in K \tag{3.9}$$

4. REMARK

Throughout the remainder of this work, frequent references are made to an N×N upper triangular matrix $S=[S_{ij}]$ whose upper and off-diagonal entries S_{ij}, $i=0,\cdots,N-1$, $j=i+1,\cdots,N$, are defined by the critical points (3.2). It is noted, however, that this is merely for illustrative purposes, and that the construction of S may not necessarily be required in practice.

5. DETERMINATION OF THE FUNCTION m(s)

Define the function m(s) as:

(5.1)
$$m(s) = \begin{cases} W_{\alpha_p}(s) & \text{if} \quad s \in (0, S_{\alpha_{p-1}\alpha_p}] \\ W_{\alpha_l}(s) & \text{if} \quad s \in [S_{\alpha_l\alpha_{l+1}}, S_{\alpha_{l-1}\alpha_l}] \qquad l=1,\cdots,p-1 \\ W_{\alpha_0}(s) & \text{if} \quad s \in [S_{\alpha_0\alpha_1},\infty) \end{cases}$$

where the integers p and $\alpha_i \in K$, $i=0,\cdots,p$ are determined such that

(5.2)
$$0 = \alpha_0 < \alpha_1 < \dots < \alpha_p = n$$

and

(5.3)
$$S_{\alpha_0\alpha_1} > S_{\alpha_1\alpha_2} > \dots > S_{\alpha_{p-1}\alpha_p}$$

Essentially, the α-sequence $\alpha_0, \dots, \alpha_p$, satisfying (5.2) and (5.3), is determined by using (3.9) so that m(s) represents the greatest lower bound for the monomials (3.1). Thus, m(s) may be considered as a piecewise monomial over $(0, \infty)$, which remains constant over $[S_{\alpha_0\alpha_1}, \infty)$ and strictly decreasing over $(0, S_{\alpha_0\alpha_1}]$ as $s \rightarrow 0$; in fact, $\lim_{s\rightarrow 0} m(s) = \lim_{s\rightarrow 0} W_n(s)$.

The m-algorithm defined below determines the α-sequence $\alpha_0, \alpha_1, \dots, \alpha_p$ with respect to (5.2) and (5.3).

1. **Initialization step.**

$\alpha_0 = 0$; $\alpha_{-1} = -1$; $S_{-1,0} = \infty$; $p = 0$.

2. **Induction steps.**

A. Let $l = p$; If α_l and $S_{\alpha_{l-1}\alpha_l}$ are known and $\alpha_l \neq n$, then $S_{\alpha_l\alpha_{l+1}}$ is the largest element in the α_l-th row of S which does not exceed $S_{\alpha_{l-1}\alpha_l}$. The integer α_{l+1} is the largest column index j such that $S_{\alpha_l j} = S_{\alpha_l\alpha_{l+1}}$.

B. Let $p = l+1$; If $\alpha_p = n$ then the m-algorithm is terminated.

6. DETERMINATION OF THE FUNCTION M(s)

Define the function M(s) as:

(6.1) $$M(s) = \begin{cases} W_{\beta_0}(s) & \text{if } s \in (0, S_{\beta_0\beta_1}] \\ W_{\beta_k}(s) & \text{if } s \in [S_{\beta_{k-1}\beta_k}, S_{\beta_k\beta_{k+1}}] \quad k=1,\cdots,q-1 \\ W_{\beta_q}(s) & \text{if } s \in [S_{\beta_{q-1}\beta_q}, \infty). \end{cases}$$

where the integers q and $\beta_j \in K$, $j=0,\cdots,q$ are determined such that

(6.2) $$0 = \beta_0 < \beta_1 < \ldots < \beta_q = n$$

and

(6.3) $$S_{\beta_0\beta_1} < S_{\beta_1\beta_2} < \cdots < S_{\beta_{q-1}\beta_q}$$

Essentially, the β-sequence β_0, ..., β_q, satisfying (6.2) and (6.3), is determined by using (3.9) so that M(s) represents the least upper bound for the monomials (3.1). Thus, M(s) may be considered as a piecewise monomial over $(0, \infty)$, which remains constant over $(0, S_{\beta_0\beta_1}]$ and strictly increasing over $[S_{\beta_0\beta_1}, \infty)$ as $s \to \infty$; in fact, $\lim_{s\to\infty} M(s) = \lim_{s\to\infty} W_n(s)$.

The M-algorithm defined below determines the β-sequence β_0, β_1, ..., β_q with respect to (6.2) and (6.3).

1. **Initialization Step.**

   $\beta_0 = 0$; $\beta_{-1} = -1$; $S_{-1,0} = 0$; $q = 0$.

2. **Induction Steps.**

   A. Let $k = q$; If β_k and $S_{\beta_{k-1}\beta_k}$ are known and $\beta_k \neq n$, then $S_{\beta_k\beta_{k+1}}$ is the smallest element in the β_k-th row of S which does not precede $S_{\beta_{k-1}\beta_k}$. The integer β_{k+1} is the smallest column index i such that $S_{\beta_k i} = S_{\beta_k\beta_{k+1}}$.

   B. Let $q = k+1$; If $\beta_q = n$ then the M-algorithm is terminated.

7. MINIMIZATION OF THE SCALE FUNCTION f(s)

As noted in Sections 5 and 6, the functions m(s) and M(s) are non-decreasing functions over $(0,\infty)$, as $s\to\infty$, and represent respectively, the greatest lower bound and the least upper bound of the monomials (3.1). Thus, the scale function f(s) in (3.8) represents the ratio of the upper bound M(s) to the lower bound m(s), which is to be minimized over $(0,\infty)$. It turns out, however, that the point s* at which this ratio is minimized coincides with one of the α- or β-critical points determined by the m- and M-algorithms, respectively. The following lemmas present a constructive proof of this phenomenon.

Lemma 7.1. No monomial $W_j(s)$, $j\in K-\{0,n\}$, may be involved in the definitions of both functions m(s) and M(s).

Proof. Let $K-\{0,n\}$ be nonempty, and let $\alpha_l=j\in K-\{0,n\}$ for some l, $0<l<p$, so that m(s) is characterized by the monomial $W_{\alpha_l}(s)$ within $[S_{\alpha_l\alpha_{l+1}}, S_{\alpha_{l-1}\alpha_l}]$. Now, for $W_{\alpha_l}(s)$ to also be involved in the definition of M(s), it has to intersect, by virtue of (6.1), with monomials of degrees higher than α_l within $[S_{\alpha_{l-1}\alpha_l},\infty)$. But this is impossible, because by virtue of (5.1), $W_{\alpha_l}(s)$ has already intersected with all monomials of degrees higher than α_l within $(0, S_{\alpha_l\alpha_{l+1}}]$. Q.E.D.

Lemma 7.2. Let the subdomain

$$D = [S_{\beta_0\beta_1}, S_{\alpha_0\alpha_1}] \tag{7.1}$$

be formed by merging the α- and β-critical points with respect to (5.3) and

(6.3), and discarding the β-critical points to the right of $S_{\alpha_0\alpha_1}$ and the α-critical points to left of $S_{\beta_0\beta_1}$. Then, the global minimum s^* of the scale function $f(s)$ over $(0,\infty)$ is located within D.

Proof. It is first noted that $S_{\beta_0\beta_1}$ and $S_{\alpha_0\alpha_1}$ represent respectively, the smallest and largest points of intersection of the constant monomial $W_0(s)$ with $W_j(s)$, $\forall j\in K-\{0\}$; that is, $S_{\beta_0\beta_1}\leq S_{\alpha_0\alpha_1}$, and hence the form of D. To show that $s^*\in D$, it suffices to show that the respective minima of $f(s)$ over the subdomains $D_\beta=(0, S_{\beta_0\beta_1}]$ and $D_\alpha=[S_{\alpha_0\alpha_1},\infty)$ may occur only at the critical points $S_{\alpha_0\alpha_1}$ and $S_{\beta_0\beta_1}$. This readily follows by noting that in view of (5.1), $m(s)=c_0$ and $M(s)$ is strictly increasing within D_α, as $s\to\infty$, and that in view of (6.1), $M(s)=c_0$ and $m(s)$ is strictly decreasing within D_β, as $s\to 0$. Hence, $s^*\in D$. Q.E.D.

Lemma 7.3. If there exists a point $x\in(0,\infty)$ at which $m(x)=M(x)$, then $s^*=x$, and in particular, the subdomain D in (7.1) reduces to $\{x\}$.

Proof. Since $m(s)\leq c_0\leq M(s)$, over $(0,\infty)$, it follows from (3.8) that $f(s)\geq 1$, $\forall s\in(0,\infty)$, and in particular, $f(x)=1$. Hence $s^*=x$, and by Lemma 7.2, $x\in D$. Now, since $m(s)<c_0<M(s)$ within the open intervals $(0, S_{\beta_0\beta_1})$, $(S_{\beta_0\beta_1}, S_{\alpha_0\alpha_1})$ and $(S_{\alpha_0\alpha_1},\infty)$, it follows that either $x=S_{\beta_0\beta_1}$ or $x=S_{\alpha_0\alpha_1}$, or both. But, $S_{\beta_0\beta_1}\neq S_{\alpha_0\alpha_1}$ would imply $m(x)<M(x)$ contrary to the assumption. Hence, $x=S_{\beta_0\beta_1}=S_{\alpha_0\alpha_1}$ and $D=\{x\}$. Q.E.D.

Lemma 7.4. Let the subdomain D in (7.1) be divided into $\alpha\beta$-subdomains of the types:

(7.2) $$D_{\alpha\beta} \equiv [S_{\alpha_l\alpha_{l+1}}, S_{\beta_k\beta_{k+1}}]$$

(7.3) $$D_{\beta\alpha} \equiv [S_{\beta_{k-1}\beta_k}, S_{\alpha_{l-1}\alpha_l}]$$

(7.4) $$D_{\alpha\alpha} \equiv [S_{\alpha_l\alpha_{l+1}}, S_{\alpha_{l-1}\alpha_l}]$$

(7.5) $$D_{\beta\beta} \equiv [S_{\beta_{k-1}\beta_k}, S_{\beta_k\beta_{k+1}}]$$

where the indices l and k are adjusted so that the boundary points of each type represent a pair of consecutive $\alpha\beta$-critical points within D. Also let the indices i and j be chosen such that $S_{\beta_{i-1}\beta_i} \in D$ is the nearest β-critical point to the left of $S_{\alpha_l\alpha_{l+1}} \in D_{\alpha\alpha}$, and that $S_{\alpha_{j-1}\alpha_j} \in D$ is the nearest α-critical point to the right of $S_{\beta_k\beta_{k+1}} \in D_{\beta\beta}$. Then, the respective monomials characterizing the functions m(s) and M(s) within the above $\alpha\beta$-subdomains are defined as in the following Table:

(7.6)

subdomain	$D_{\alpha\beta}$ and $D_{\beta\alpha}$	$D_{\alpha\alpha}$	$D_{\beta\beta}$
M(s):	$W_{\beta_k}(s)$	$W_{\beta_i}(s)$	$W_{\beta_k}(s)$
m(s):	$W_{\alpha_l}(s)$	$W_{\alpha_l}(s)$	$W_{\alpha_j}(s)$

Moreover, if [x,y] is an $\alpha\beta$-subdomain of any of the types (7.2)-(7.5), within which $M(s)=W_{\beta_r}(s)$ and $m(s)=W_{\alpha_t}(s)$, then the minimum s_m of f(s) within [x,y] is located at $s_m = x$ if $\beta_r > \alpha_t$, or at $s_m = y$ if $\beta_r < \alpha_t$.

Proof. Clearly, (7.2)-(7.5) represent the only possible types of $\alpha\beta$-subdomains whose boundary points are defined by pairs of adjacent critical points within D. The characterization of the functions m(s) and M(s) within an $\alpha\beta$-subdomain, is simply based on the type and position of the boundary points of that $\alpha\beta$-subdomain. To be specific, consider an $\alpha\beta$-subdomain [x,y] of one of the types (7.2)-(7.5). Now, if $[x,y]\equiv D_{\alpha\beta}$ or $[x,y]\equiv D_{\beta\alpha}$, then it readily follows from (5.1) and (6.1) that m(s) and M(s) are characterized as given in Table 7.6. However, the function M(s) within $[x,y]\equiv D_{\alpha\alpha}$ and m(s) within $[x,y]\equiv D_{\beta\beta}$ are not directly identified by the respective type of the $\alpha\beta$-subdomain [x,y]. Nonetheless, information about these functions within [x,y] may easily be obtained by considering the nearest $\alpha\beta$-critical point within D$-$[x,y], whose type is the opposite of that of the boundary points x and y of [x,y]. The existence of at least one such nearest point is guaranteed by noting that $D\equiv D_{\beta\alpha}$ and $[x,y]\subseteq D$; specifically, there always exists a β-critical point to the left of x if $[x,y]\equiv D_{\alpha\alpha}$, and there always exists an α-critical point to the right of y if $[x,y]\equiv D_{\beta\beta}$. Hence (7.6).

As regards the minimum s_m of f(s) within the $\alpha\beta$-subdomain [x,y], it suffices to explicitly express and minimize f(s) within [x,y]. Specifically, in view of (3.1) and (3.8), the scale function may explicitly be expressed in the form:

$$f(s) = \frac{M(s)}{m(s)} = \frac{W_{\beta_r}(s)}{W_{\alpha_t}(s)} = \left(\frac{c_{\beta_r}}{c_{\alpha_t}}\right) s^{(\beta_r-\alpha_t)} \qquad s\in[x,y]$$

which is, by Lemma 7.1, strictly increasing or decreasing depending on whether $\beta_r>\alpha_t$ or $\beta_r<\alpha_t$, respectively. Thus, the minimum s_m of f(s) within [x,y] is located at $s_m = x$ if $\beta_r>\alpha_t$, or at $s_m = y$ if $\beta_r<\alpha_t$. Q.E.D.

This completes a constructive proof of the phenomenon that the scale factor is indeed a critical point.

We shall now describe an algorithm for determining the global minimum of f(s) within D. Let the array of all $\alpha\beta$-critical points within D be denoted by x, with $x_0 = S_{\alpha_0\alpha_1}$ and $x_{i+1} \leq x_i$, $i \geq 0$. Thus, the $\alpha\beta$-subdomains of the types (7.2)-(7.5) may be expressed in the form $[x_{i+1}, x_i]$ for appropriate values of $i \geq 0$. The f-algorithm described below, first determines the type of the $\alpha\beta$-subdomain $[x_{i+1}, x_i]$ at hand, and then determines the local minimum of f(s) over the subdomain accordingly. Among the local minima thus found, the scale factor s* is taken to be the one at which f(s) attains its minimum value.

To simplify the description of the f-algorithm, the following construct is defined separately.

```
B:  IF   $\beta_k \geq \alpha_l$   THEN
        $f_i = \left(\frac{c_{\beta_k}}{c_{\alpha_l}}\right) (x_{i+1})^{(\beta_k - \alpha_l)}$ ;  $s_i = x_{i+1}$
    ELSE
        $f_i = \left(\frac{c_{\beta_k}}{c_{\alpha_l}}\right) (x_i)^{(\beta_k - \alpha_l)}$ ;  $s_i = x_i$
```

This construct will be referred to as block B. Evidently, the inputs to this block are the $\alpha\beta$-critical points x_i and x_{i+1}, representing the $\alpha\beta$-subdomain $[x_{i+1}, x_i]$, and the values α_l and β_k characterizing m(s) and M(s) over $[x_{i+1}, x_i]$, respectively. This block returns s_i as the minimizer of f(s) over $[x_{i+1}, x_i]$, and f_i as $f(s_i)$. The f-algorithm is described as follows:

```
1.  Initialize: $i = 0$ ; $x_i = S_{\alpha_0\alpha_1}$ ; $f_0 = c_n x_i^n / c_0$ ; $s_0 = x_i$
2.  IF  $x_i = S_{\beta_0\beta_1}$  GO TO 6
3.  Let $x_{i+1}$ be the largest $\alpha\beta$-critical point smaller than $x_i$;
4.  IF  $x_i = S_{\alpha_{l-1}\alpha_l}$  THEN
        IF  $x_{i+1} = S_{\beta_{k-1}\beta_k}$  THEN B
        ELSE $x_{i+1} = S_{\alpha_l\alpha_{l+1}}$;
            $S_{\beta_k\beta_{k-1}} = x_{i+j}$ is the next immediate $\beta$-critical point with $j>0$;
            B;
    ELSE  $x_i = S_{\beta_k\beta_{k+1}}$
        IF  $x_{i+1} = S_{\alpha_l\alpha_{l+1}}$  THEN B
        ELSE $x_{i+1} = S_{\beta_{k-1}\beta_k}$;
            $S_{\alpha_{l-1}\alpha_l} = x_{i-m}$ is the last immediate $\alpha$-critical point with
            $m>0$;
            B;
5.  $i = i+1$ ; GO TO 2;
6.  Terminate with  $f_j = \min(f_i)$ and $s^* = s_j$
```

7.1. REMARK

In this section we present a lemma which will be used in Section 8 for determining an estimate of the number of arithmetic operations involved in the m- and M-algorithms.

Lemma 7.1.1. There may exist at most N−*1* $\alpha\beta$-subdomains of the tyes (7.2)-(7.5) within D.

Proof. It suffices to show that D may contain at most N critical points. To do this, it is first noted that in the case the functions m(s) and M(s) intersect, say at x, we have by Lemma 7.3, $D_{\alpha\beta} \equiv D_{\beta\alpha} \equiv D-\{x\}$, regardless of the value of N. Now suppose that the functions m(s) and M(s) do not intersect. Then, except for $W_0(s)$ and $W_n(s)$ which are permanently fixed in the definitions of m(s) and M(s), there may exist monomials $W_j(s)$, $j \in K-\{0,n\}$, passing between m(s) and M(s) without being involved in their definitions. That is, if L of the monomials $W_j(s)$, $j \in K-\{0,n\}$, are present in the definition of m(s), there can be at most N−(L+2) of the remaining monomials present in the definition of M(s). Therefore, in this case, there are L+*1* α-critical points and at most N−(L+*1*) β-critical points, giving a maximum of N $\alpha\beta$-critical points over (*0* ,∞), and hence within D. Q.E.D.

8. COMPARISON

In this section, the present discrete minimization method is compared with Dunaway's general mathematical programming approach [1]. Evidently, the problem (3.7) readily suggests for its solution, the use of a linear programming method such as the revised simplex method. The comparison, therefore, requires specification of essential factors involved in the application of the revised simplex process to (3.7). Any realistic comparison must include the four factors: complexity, storage requirement, operation count and numerical stability. In the following, these four factors are considered.

8.1 Complexity

Problem (3.7) consists of 2N inequality constraints and *3* unrestricted (in sign) independent variables T, D and S. In order to begin Phase I of the revised simplex method, this system of inequality constraints must first be transformed into a system of equality constraints. The transformation involves [2, p. 173] replacing each of the unrestricted variables T, D and S by the difference of a pair of nonnegative variables, and thus doubling the number of independent variables to *6*. In addition, it involves the introduction of a total of 2N slack and surplus variables, resulting in a system of 2N equations in 2N+*6* nonnegative variables. Moreover, the coefficients of the system of equations thus obtained may have to be adjusted so that the right hand side trems of the system are all nonnegative; the result is a system of equations with nonnegative right hand sides, containing a total of 2N positive and negative unit vectors. Now, for the Phase I of the simplex process to be applicable, this last system of equations must undergo another transformation to convert all but one of the negative unit vectors (if any) to distinct positive unit vectors; the remaining negative unit vector remains unchanged while an additional artificial variable is introduced.

Fortunately, most computer centers generally support a linear programming system; otherwise, one would have to be proficient in the theory of the linear programming methods, and the associated computational techniques, to embark on the design of such system with an overwhelming amount of coding. However, since the supported linear programming systems are generally coded to be much more than an implementation of the simplex method, they may require a certain level of familiarity on how to use and interact with them. For instance, we have access to the IMSL/Zx2LP [4] linear programming package, which may be invoked by a Fortran call statement involving 13 parameters of which 7 are used as inputs, 4 as outputs, and 2 as auxiliaries. Thus the complexity and/or size of such systems may be factors of concern to users unfamiliar with linear programming methods, and/or to users of computers with limited core storage.

In contrast, the present method is conceptually simple and easy to implement. It may be coded into a small size program (e.g., about 100 statements in Fortran) requiring the coefficients and degree of the polynomial to be scaled as inputs, and returning the scale factor as output.

8.2 Storage Requirement

It follows from the preparatory transformations discussed in Section 8.1 that the major storage requirement of the revised simplex method on problem (3.7) consists of $2N\times(2N+7)$ locations in the presence of an artificial variable, or otherwise, of $2N \times (2N+6)$ locations.

In the present method, the major storage requirement may at first seem to consist of $N(N-1)/2$ locations for storing the entries S_{ij}, $i=0,\cdots,N-1$, $j=i+1,\cdots,N$, of the matrix S, which are computed according to (3.2) and are referenced only by the m- and M-algorithms. It turns out, however, that the major storage requirement of the present method consists of at most N locations. To justify this fact, it suffices to note that except for the first row, no other rows of the matrix S may be scanned by both the m- and M-algorithms during the determination of the α- and β-sequences (5.2) and (6.2), respectively; and that, one scan of the first row of S is adequate for determining both α_1 and β_1. In fact, the maximum number of references to the S_{ij} entries may be at most $N(N-1)/2$, because in view of Lemmas (7.1) and (7.3), some rows of S may never get scanned. In particular, among the S_{ij} entries referenced, only those located within the subdomain D in (7.1) need be stored; these stored S_{ij} entries are utilized by the f-algorithm for the global minimization of the scale function. And it follows from Lemma (7.1.1), that there are at most N $\alpha\beta$-critical points within D. Hence, the requirement of at most N storage locations.

8.3. Operation Count

In this Section, estimates of the number of arithmetic operations involved in each method are determined. The arithmetic operations taken into account are those of multiplications, divisions, and exponentiations; the latter operations are used only by the present method. These operations are usually performed in computer hardware by similar methods using only shifting, addition and subtraction operations; such methods for logarithm and exponentiation are given in [6, p. 26].

As suggested in [3, p. 215], the total number of operations performed at each iteration by the revised simplex method, applied to (3.7), is about $2N(2N+7) + (2N)^2 + 3(2N) = 4(2N^2 + 5N)$. According to [2, p. 65], there are usually between $2N$ and $4N$ iterations involved in the simplex process to find a minimum feasible solution of (3.7). Thus, in the linear programming approach, the total number of operations is between $8(2N^3+5N^2)$ and $16(2N^3+5N^2)$.

In the present method, the bulk of the arithmetic operations takes place in the m- and M-algorithms for computing the $\alpha\beta$-critical points according to (3.2), and in block B of the f-algorithm for computing the local minima. As pointed out in Section 8.2, the m- and M-algorithms may involve at most $N(N-1)/2$ references to (3.2), with each reference requiring 3 operations. And, in view of Lemma 7.1.1, the f-algorithm may involve at most $N-1$ invocations of block B, with each invocation involving 3 operations. Thus, the present method involves a maximum of $N-1$ multiplications, $(N-1)(N+1)$ divisions and $(N-1)(N+2)$ exponentiations, adding up to a maximum of $3(N-1)(N+2)/2$ operations.

8.4 Numerical Stability and Accuracy of finding s*

Inherently, the present method is numerically stable as it involves no possibility of degeneracy, cycling, or propagation of accumulated round-off or numerical errors through the minimization process. This, however, may not be the case for the simplex method, although effective methods exist to handle these possibilities in the simplex process [2]. Numerically, the system of simultaneous equations passed to Phase I of the revised simplex process, represents a perturbed version of the problem (3.7); this is due to the preparatory transformations (described in Section 8.1) applied to (3.7). Moreover, the solution determined by the simplex process has to undergo a final transformation, as it represents an approximation to the logarithm of the scale factor.

Computationally, the cost of preserving maximum numerical accuracy through the simplex process can increase, as it involves periodical inversion of the basis matrix.

In the present method, however, the scale factor is determined to be one of the critical points computed directly by (3.2). The accuracy of the optimal scale factor, therefore, depends solely on the limiting accuracy of the finite precision arithmetic used in computing (3.2).

9. NUMERICAL RESULTS AND REMARKS

To investigate numerically the effect of scaling on the roots of polynomials, we adapted the use of Laguerre's root-finding method. Our choice of Laguerre's method was solely based on its usefulness as a general purpose root-finding method [8]; indeed, the choice of the root-finding method was not particularly important, as the primary concern in this investigation was the numerical quality of the approximated roots before and after scaling, as opposed to how they were approximated.

The numerical quality of the approximated roots was measured in terms of the maximum relative error involved. However, in most of the tests carried out, it turned out that in general the scaling did not significantly (if at all) affect the quality of the approximated roots. Therefore, the cost of approximating the roots with and without scaling was also considered as a factor in measuring the effect of scaling. The approximation cost was measured in terms of the number of Laguerre's iterations involved in approximating the roots before and after scaling. For each root z_i, $i=1,\cdots,n$, this number comprised Laguerre's iterations in two successive groups: the first group consisted of iterations leading to an approximate root $z_i^{(k)}$ satisfying the stopping criteria

(9.1) $$|z_i^{(k+1)}-z_i^{(k)}| \leq \epsilon = 10^{-20} \qquad 0\leq k\leq 100, \quad i=1,\cdots n$$

where the superscripts denote the iteration numbers, and where $z_i^{(0)}=0$; the second group consisted of iterations leading possibly to an improved approximation $z_i^{(k+j)}$ satisfying simultaneously the stopping critera [10, p.463]:

$$\begin{cases} \left|z_i^{(k+j+1)}-z_i^{(k+j)}\right| \geq \left|z_i^{(k+j)}-z_i^{(k+j-1)}\right| \\ \left|z_i^{(k+j)}-z_i^{(k+j-1)}\right| \leq \epsilon \end{cases} \qquad 0\leq j\leq 50, \quad i=1,\cdots,n$$

with k satisfying (9.1). The iterations involved in these two groups shall henceforth be referred to as the primary and secondary iterations, respectively. Once approximated, each root was extracted from the respective polynomial by the polynomial deflation process. It is noted that the choice of $z_i^{(0)}=0$, $i=1,\cdots,n$, always resulted, at least for polynomials with real roots only, in the approximate roots in increasing order; this was a desirable feature as it kept the polynomial deflation process stable [9].

Finally, once all the roots were approximated, each was optionally refined by one last application of Laguerre's iteration to the undeflated polynomial [9, p. 65]. The refinement step was taken optionally, because the computations involved in this last step might possibly render exponent underflow or overflow on polynomials with high variations in the magnitude of their coefficients. In this respect, we also investigated the extent to which the degrees of such polynomials could be raised with and without this last refinement step.

In all test runs, the scaled polynomial was taken to be the normalized version of (2.3) defined by the monic polynomial:

$$R(s,P) = \sum_{k=1}^{n} \left(\frac{a_k\, s^{k-n}}{a_n}\right) x^k \tag{9.2}$$

It is noted that, because of the finite precision arithmetic involved in the computation of the coefficients of (2.3) and (9.2), the numerical representation of both Q(s,P) and R(s,P) may be in error [9, p. 30]; this is, of course, in addition to the error that already may be present in the coefficients of the original polynomial P(z). One way to avoid the introduction of possible errors in the coefficients of the scaled polynomials is to choose, as the scale factor, the integer $s^{**}=2^j$ with j minimizing $|s^{*}-2^j|$. This essentially has the effect of adjusting the exponents of the coefficients without altering their corresponding mantissa. The choice of s^{**} as an alternate optimal scale factor was also investigated.

The algorithms described in this thesis were coded in Fortran using double precision floating-point arithmetic. The program was successfully used on the CYBER 830D machine providing an 11 binary digit exponent and 96 binary digits double precision mantissa for floating-point operations.

The effect of scaling was examined on a large number of test polynomials, some well-conditioned, some ill-conditioned and others randomly generated by a number of techniques [5]. The following represents only a small sample of the polynomials tested; this sample epitomizes polynomials with wide variations in the magnitude of their coefficients.

$$P_1(z) = \prod_{k=1}^{n} (z-10^k) \qquad z_k = 10^k,\ k=1,\cdots,n$$

$$P_2(z) = \prod_{k=1}^{n} (z-10^{-k}) \qquad z_k = 10^{-k},\ k=1,\cdots,n$$

$$P_3(z) = \prod_{k=0(2)}^{n \text{ even}} (z^2+10^{2k}z+10^{2k+1}) \qquad z_k = \bar{z}_{k+1} = \frac{1}{2}10^k(-1+i39^{\frac{1}{2}}),\ k=0(2)n$$

$$P_4(z) = \prod_{k=1}^{n} (z-k) \qquad z_k = k,\ k=1,\cdots n$$

$$P_5(z) = \prod_{k=1}^{n} (z-10-10^{-k}) \qquad z_k=10+10^{-k},\ k=1,\cdots,n$$

$$P_6(z) = \prod_{k=1}^{n} (z-10k)^{\lfloor \frac{1}{2}k+1 \rfloor} \qquad z_k=10k,\ k=1,\cdots,n$$

The collected data on the effect of scaling the above polynomials on their roots are presented by Tables 1-6, respectively. These Tables are organized as follows: the first column (DEG) gives the degrees of the particular polynomial tested; columns 2-5 give the results obtained on the original polynomial P(z); columns 6-10 on the scaled polynomial R(s**,P); columns 11-15 on the scaled polynomial R(s*,P); the last column (RO) indicates whether the final refinement option was enabled (E) or disabled (D). More specifically: columns 6 and 11 show the corresponding optimal scale factors s** and s*, respectively; columns 2, 7 and 12 the variations (VAR); columns 3, 8 and 13 the maximum relative error (MRE) among all the approximated roots; columns 4, 9 and 14 give the total number (NP) of primary iterations involved in approximating all the roots; columns 5, 10 and 15 the total number (NS) of secondary iterations involved in improving all the approximate roots. The presence of blank entries in columns 3-5 indicates that, on our machine, the original polynomial could not be handled computationally for the corresponding degrees specified in column 1. Finally, in all the tests carried out, the refinement option was initially enabled; however, if during a test run this option resulted in computational breakdown due to floating-point underflow or overflow, then the test would be repeated with the refinement option disabled.

The polynomials P_1, P_2, and P_3 are well-conditioned. P_2 and P_3 were taken from [5] and [7], respectively. For higher degrees, overflow is likely to occur in the evaluation of P_1 and P_3, and underflow is likly in the evaluation of P_2. Tables 1-3 show that the scaling has virtually no effect on the accuracy of the approximated roots before and after scaling, and that, for higher degrees, the scaled polynomials involve fewer iterations than does the original polynomial.

Polynomials of the form P_4 represent typical examples of ill-conditioned polynomials; it is the linear distribution of the roots of such polynomials that make them ill-conditioned [9]. Table 4 shows that the results before and after scaling are of comparable qualities, and that for higher degrees, fewer iterations are involved in the approximation of the roots of the scaled polynomials. For $n=20$, the coefficients of P_4 and $R(s^*,P_4)$ are given by Table 7. Comparison shows that our result of the scale factor for P_4 with $n=20$ is slightly better than that obtained in [1]; this is probably because the present scaling process is less subject to accumulation of rounding or numerical errors than the simplex process.

Polynomials of the form P_5 are very ill-conditioned, as each possesses a cluster of nearly equal roots. Table 5 shows the results with and without the final refinement step. In this case too, the results are of comparable qualities; although for higher degrees, the approximated roots of both the original and scaled polynomials are greatly in error. As shown by the results in Table 6, similar remarks apply to the polynomial P_6 with roots z_k of multiplicity $\lfloor \frac{1}{2}k+1 \rfloor$, $k=1,\cdots,n$, where $\lfloor . \rfloor$ denotes the floor function.

10. CONCLUSION

As the primary object of this work, a new and efficient discrete method for scaling of real polynomials has been presented. On the theoretical side, the method is based on the phenomenon that the scale factor coincides with one of the intersections between monomials defined by the respective magnitudes of the nonzero terms of the polynomial to be scaled. A constructive proof of this phenomenon has been presented. On the practical side, the method has been described in terms of three efficient and easy to implement algorithms, which may be coded, for example, into a small Fortran program of about 100 statements. In addition, on both theoretical and practical sides, it has been shown that the method compares favourably with an existing mathematical programming approach, in terms of the complexity, storage requirement, operation count, and numerical stability.

As the secondary object of this work, an extensive numerical investigation has been carried out on the effect of scaling of real polynomials on their approximate roots obtained by a general root-finding method which does not involve the calculation of the greatest common divisor of two polynomials. It has been numerically experienced that scaling of polynomials has an insignificant, if any, effect on the numerical quality of the approximated roots. However, it has been observed that the initial scaling may extend the range of polynomials, with wide variation in the magnititude of their coefficients, to be solved by general root-finding algorithms.

In continuing this research work, the author has undertaken the development of a method for parallel determination of approximate factors of real polynomials using Euclid's algorithm; the approximate factors of interest are those with only one real multiple root or one pair of complex conjugate multiple roots. Moreover, the author is also concerned with the theoritical justification of the numerical insensitivity of the roots of real polynomials to scaling.

REFERENCES

[1] D. K. Dunaway, Calculation of Zeros of a Real Polynomial Through Factorization Using Euclid's Algorithm, SIAM J. Numer. Anal., Vol. 11., No. 6, December 1974, pp. 1087-1104.

[2] S. Gass, Linear Programming: Methods and Applications (fourth edition), McGraw-Hill, New York, 1975.

[3] G. Hadley, Linear Programming, Addison-Wesley, Reading, Mass., 1962.

[4] IMSL Library, Fortran Subroutines for Mathematics and Statistics, Vol. 4, Edition 9.2, IMSL LIB-0009, Houston, Texas, November 1984.

[5] M. A. Jenkins and J. F. Traub, Principles for Testing Polynomial Zerofinding Programs, ACM Transaction on Mathematical Software, Vol. 1, No. 1, March 1975, pp. 26-34.

[6] D. E. Knuth, The Art of Computer Programming, Vol. 1, 2nd edition, Addison-Wesley, Reading, Mass., 1973.

[7] G. Peters and J. H. Wilkinson, Practical Problems Arising in the Solution of Polynomial Equations, J. Inst. Maths Applics, No. 8, 1971, pp. 16-35.

[8] A. Ralston, A First Course in Numerical Analysis, McGraw-Hill, New York, 1965.

[9] J. H. Wilkinson, Rounding errors in Algebraic Processes, Prentice-Hall, Englewood Cliffs, N.J., 1963.

[10] J. H. Wilkinson, The Algebraic Eigenvalue Problem, London, Oxford University Press, 1965.

Table 1. $P_1(z) = \prod_{k=1}^{n} (z-10^k)$

	$P_1(z)$				$R(s^{**},P_1)$					$R(s^*,P_1)$					
DEG	VAR	MRE	NP	NS	s**	VAR	MRE	NP	NS	s*	VAR	MRE	NP	NS	RO
5	.1500E+02	.4136E-28	13	3	.1024E+04	.3081E+01	.4136E-28	13	3	.1000E+04	.3050E+01	.9926E-28	13	7	E
6	.2100E+02	.5170E-28	17	6	.4096E+04	.4888E+01	.5170E-28	17	6	.3162E+04	.4551E+01	.2068E-27	17	11	E
7	.2800E+02	.4136E-28	21	5	.8192E+04	.6397E+01	.4136E-28	21	5	.1000E+05	.6051E+01	.1809E-27	21	6	E
8	.3600E+02	.4235E-28	25	7	.3277E+05	.8112E+01	.4235E-28	25	7	.3162E+05	.8051E+01	.2326E-27	25	11	E
9	.4500E+02	.9694E-28	29	11	.1311E+06	.1064E+02	.9694E-28	29	11	.1000E+06	.1005E+02	.2068E-27	29	12	E
10	.5500E+02	.1355E-27	32	11	.2621E+06	.1296E+02	.1355E-27	32	11	.3162E+06	.1255E+02	.2965E-27	32	12	D
11	.6600E+02	.2033E-27	37	12	.1049E+07	.1517E+02	.2033E-27	36	13	.1000E+07	.1505E+02	.3309E-27	36	15	D
12	.7800E+02	.1735E-27	42	13	.4194E+07	.1879E+02	.1735E-27	40	15	.3162E+07	.1805E+02	.2220E-27	40	14	D
13	.9100E+02				.8389E+07	.2158E+02	.1525E-27	45	18	.1000E+08	.2105E+02	.2647E-27	45	20	E
14	.1050E+03				.3355E+08	.2473E+02	.2382E-27	48	26	.3162E+08	.2455E+02	.4549E-27	48	19	D
15	.1200E+03				.1342E+09	.2907E+02	.2482E-27	52	26	.1000E+09	.2805E+02	.5790E-27	52	23	D
16	.1360E+03				.2684E+09	.3262E+02	.2728E-27	56	21	.3162E+09	.3205E+02	.5912E-27	56	26	D
17	.1530E+03				.1074E+10	.3633E+02	.2547E-27	60	20	.1000E+10	.3605E+02	.3469E-27	60	29	D
18	.1710E+03				.4295E+10	.4175E+02	.2202E-27	64	33	.3162E+10	.4055E+02	.2910E-27	64	31	D
19	.1900E+03				.8590E+10	.4571E+02	.3260E-27	68	24	.1000E+11	.4505E+02	.6519E-27	68	27	D
20	.2100E+03				.3436E+11	.5041E+02	.2812E-27	72	26	.3162E+11	.5005E+02	.3805E-27	72	37	D
21	.2310E+03				.1374E+12	.5657E+02	.2168E-27	76	34	.1000E+12	.5505E+02	.4172E-27	76	28	D
22	.2530E+03				.2749E+12	.6122E+02	.2682E-27	80	32	.3162E+12	.6055E+02	.5722E-27	80	33	D
23	.2760E+03				.1100E+13	.6655E+02	.3049E-27	179	30	.1000E+13	.6605E+02	.4959E-27	179	37	D

DEG	Degree of $P_1(z)$	VAR	Variation
MRE	Maximum relative error	RO	Refinement options; E:enabled, D:disabled
NP	Number of primary Laguerre's steps	NS	Number of secondary Laguerre's steps
s*	Optimal scale factor	s**	2^j with j minimizing $\|s^* - 2^j\|$
$R(s^*,P_1)$	P_1 scaled by s*	$R(s^{**},P_1)$	P_1 scaled by s**

Table 2. $P_2(z) = \prod_{k=1}^{n} (z-10^{-k})$

	$P_2(z)$				$R(s^{**},P_2)$					$R(s^*,P_2)$					
DEG	VAR	MRE	NP	NS	s**	VAR	MRE	NP	NS	s*	VAR	MRE	NP	NS	RO
5	.1500E+02	.1233E-27	13	5	.1953E-02	.3922E+01	.1233E-27	13	5	.1000E-02	.3050E+01	.1849E-27	13	3	E
6	.2100E+02	.1541E-27	17	9	.4883E-03	.5117E+01	.1541E-27	17	9	.3162E-03	.4551E+01	.1578E-27	17	8	E
7	.2800E+02	.7889E-28	21	9	.2441E-03	.7601E+01	.1262E-27	21	9	.1000E-03	.6051E+01	.1479E-27	21	6	E
8	.3600E+02	.9861E-28	25	8	.6104E-04	.9193E+01	.9861E-28	25	8	.3162E-04	.8051E+01	.1685E-27	25	14	E
9	.4500E+02	.1128E-27	29	11	.1526E-04	.1097E+02	.1128E-27	29	11	.1000E-04	.1005E+02	.2633E-27	29	11	E
10	.5500E+02	.1505E-27	33	17	.7629E-05	.1446E+02	.1505E-27	33	17	.3162E-05	.1255E+02	.2504E-27	33	10	E
11	.6600E+02	.1578E-27	36	20	.1907E-05	.1673E+02	.9466E-28	37	19	.1000E-05	.1505E+02	.1893E-27	37	14	E
12	.7800E+02	.1317E-27	39	20	.4768E-06	.1912E+02	.1317E-27	41	18	.3162E-06	.1805E+02	.3205E-27	41	17	E
13	.9100E+02	.1763E-27	57	68	.2384E-06	.2369E+02	.1578E-27	45	18	.1000E-06	.2105E+02	.2761E-27	45	15	E
14	.1050E+03	.2840E-27	72	119	.5960E-07	.2648E+02	.3155E-27	48	19	.3162E-07	.2455E+02	.3471E-27	48	19	D
15	.1200E+03	.6950E-27	86	168	.1490E-07	.2944E+02	.3787E-27	52	19	.1000E-07	.2805E+02	.3820E-27	52	23	D
16	.1360E+03	.1784E-25	97	217	.7451E-08	.3503E+02	.3787E-27	56	21	.3162E-08	.3205E+02	.6626E-27	56	30	D
17	.1530E+03	.1885E-06	106	227	.1863E-08	.3848E+02	.3787E-27	60	22	.1000E-08	.3605E+02	.4102E-27	60	25	D
18	.1710E+03				.4657E-09	.4206E+02	.4102E-27	64	24	.3162E-09	.4055E+02	.4733E-27	64	29	D

DEG	Degree of $P_2(z)$	VAR	Variation
MRE	Maximum relative error	RO	Refinement options; E:enabled, D:disabled
NP	Number of primary Laguerre's steps	NS	Number of secondary Laguerre's steps
s*	Optimal scale factor	s**	2^j with j minimizing $\|s^* - 2^j\|$
$R(s^*,P_2)$	P_2 scaled by s*	$R(s^{**},P_2)$	P_2 scaled by s**

Table 3. $P_3(z) = \prod_{k=0(2)}^{n \text{ even}} (z^2+10^{2k}z+10^{2k+1})$

	$P_3(z)$				$R(s^{**},P_3)$					$R(s^*,P_3)$					
DEG	VAR	MRE	NP	NS	s**	VAR	MRE	NP	NS	s*	VAR	MRE	NP	NS	RO
6	.1500E+02	.1226E-27	21	8	.2560E+03	.4368E+01	.1226E-27	21	8	.3162E+03	.4000E+01	.1226E-27	21	10	E
8	.2800E+02	.1264E-27	31	14	.4096E+04	.8450E+01	.1264E-27	31	14	.3162E+04	.8000E+01	.8564E-28	31	10	E
10	.4500E+02	.9612E-28	41	22	.3277E+05	.1209E+02	.9612E-28	41	22	.3162E+05	.1200E+02	.1707E-27	41	20	E
12	.6600E+02	.1097E-27	50	24	.2621E+06	.1849E+02	.1097E-27	50	24	.3162E+06	.1800E+02	.1842E-27	50	26	D
14	.9100E+02				.4194E+07	.2498E+02	.9612E-28	61	28	.3162E+07	.2400E+02	.1832E-27	61	36	E
16	.1200E+03				.3355E+08	.3221E+02	.1798E-27	70	36	.3162E+08	.3200E+02	.2717E-27	70	38	D
18	.1530E+03				.2684E+09	.4071E+02	.1581E-27	80	44	.3162E+09	.4000E+02	.1729E-27	80	32	D
20	.1900E+03				.4295E+10	.5133E+02	.1628E-27	90	52	.3162E+10	.5000E+02	.2043E-27	90	56	D
22	.2310E+03				.3436E+11	.6043E+02	.2592E-27	100	58	.3162E+11	.6000E+02	.3312E-27	100	44	D
24	.2760E+03				.2749E+12	.7273E+02	.1515E-27	110	68	.3162E+12	.7200E+02	.3024E-27	110	54	D

DEG	Degree of $P_3(z)$	VAR	Variation
MRE	Maximum relative error	RO	Refinement options: E:enabled, D:disabled
NP	Number of primary Laguerre's steps	NS	Number of secondary Laguerre's steps
s*	Optimal scale factor	s**	2^j with j minimizing $\|s^* - 2^j\|$
$R(s^*,P_3)$	P_3 scaled by s*	$R(s^{**},P_3)$	P_3 scaled by s**

Table 4. $P_4(z) = \prod_{k=1}^{n} (z-k)$

	$P_4(z)$				$R(s^{**},P_4)$					$R(s^*,P_4)$					
DEG	VAR	MRE	NP	NS	s**	VAR	MRE	NP	NS	s*	VAR	MRE	NP	NS	RO
5	.2438E+01	.2726E-26	16	3	.2000E+01	.1449E+01	.2726E-26	16	3	.2605E+01	.1105E+01	.2501E-25	16	7	E
6	.3246E+01	.1782E-25	21	10	.4000E+01	.1815E+01	.1782E-25	21	10	.2994E+01	.1438E+01	.2847E-25	21	7	E
7	.4118E+01	.1372E-24	26	12	.4000E+01	.1998E+01	.1372E-24	26	12	.3380E+01	.1715E+01	.5491E-24	26	10	E
8	.5072E+01	.4618E-24	31	8	.4000E+01	.2154E+01	.4618E-24	31	8	.3764E+01	.2048E+01	.1557E-23	31	8	E
9	.6069E+01	.1914E-23	36	13	.4000E+01	.2420E+01	.1914E-23	36	13	.4147E+01	.2342E+01	.5468E-23	36	13	E
10	.7106E+01	.2734E-22	41	22	.4000E+01	.2945E+01	.2734E-22	41	22	.4529E+01	.2675E+01	.3628E-22	41	15	E
11	.8179E+01	.6871E-22	48	16	.4000E+01	.3513E+01	.6871E-22	47	17	.4909E+01	.2979E+01	.6440E-21	47	22	E
12	.9286E+01	.4586E-21	55	19	.4000E+01	.4100E+01	.4586E-21	55	19	.5289E+01	.3313E+01	.1120E-20	55	18	E
13	.1042E+02	.2110E-20	62	25	.8000E+01	.4658E+01	.2110E-20	61	26	.5668E+01	.3624E+01	.8263E-20	62	24	E
14	.1159E+02	.1114E-19	68	21	.8000E+01	.4851E+01	.1114E-19	68	21	.6046E+01	.3958E+01	.1420E-19	68	21	E
15	.1279E+02	.3496E-19	74	26	.8000E+01	.5022E+01	.3496E-19	74	26	.6423E+01	.4275E+01	.1633E-18	74	29	E
16	.1401E+02	.5183E-18	80	25	.8000E+01	.5174E+01	.5183E-18	80	25	.6800E+01	.4609E+01	.1045E-17	80	25	E
17	.1526E+02	.2113E-17	87	27	.8000E+01	.5339E+01	.2113E-17	86	28	.7177E+01	.4931E+01	.4748E-17	86	28	E
18	.1653E+02	.7955E-17	93	31	.8000E+01	.5489E+01	.7955E-17	92	32	.7553E+01	.5264E+01	.4357E-16	92	30	E
19	.1783E+02	.7591E-16	99	34	.8000E+01	.5625E+01	.7591E-16	99	34	.7929E+01	.5590E+01	.3618E-15	99	27	E
20	.1914E+02	.2619E-15	105	35	.8000E+01	.6086E+01	.2619E-15	105	35	.8304E+01	.5923E+01	.1554E-14	105	30	E
21	.2048E+02	.3440E-14	117	40	.8000E+01	.6641E+01	.1934E-14	111	41	.8679E+01	.6252E+01	.5097E-14	111	32	E
22	.2183E+02	.1223E-13	147	35	.8000E+01	.7206E+01	.2171E-13	117	44	.9054E+01	.6585E+01	.1100E-13	117	43	E
23	.2320E+02	.1201E-12	243	33	.8000E+01	.7779E+01	.9782E-13	133	35	.9429E+01	.6916E+01	.3292E-12	129	29	E
24	.2460E+02	.5635E-12	371	34	.8000E+01	.8376E+01	.3596E-12	141	43	.9803E+01	.7249E+01	.5421E-12	145	35	E
25	.2601E+02	.3857E-11	569	38	.8000E+01	.8989E+01	.6098E-11	174	39	.1018E+02	.7583E+01	.8746E-11	171	36	E
26	.2744E+02	.2118E-10	615	35	.8000E+01	.9610E+01	.9583E-11	348	33	.1055E+02	.7916E+01	.2247E-10	244	36	E
27	.2889E+02	.1589E-09	935	39	.8000E+01	.1024E+02	.1474E-09	579	37	.1092E+02	.8251E+01	.1537E-09	408	36	E
28	.3035E+02	.5158E-09	1166	36	.8000E+01	.1088E+02	.5720E-09	760	39	.1130E+02	.8584E+01	.1503E-08	566	43	E
29	.3183E+02	.1609E-08	1221	42	.1600E+02	.1101E+02	.3490E-08	997	49	.1167E+02	.8921E+01	.4875E-08	868	41	E

DEG	Degree of $P_4(z)$	VAR	Variation
MRE	Maximum relative error	RO	Refinement options; E:enabled, D:disabled
NP	Number of primary Laguerre's steps	NS	Number of secondary Laguerre's steps
s*	Optimal scale factor	s**	2^j with j minimizing $\|s^* - 2^j\|$
$R(s^*,P_4)$	P_4 scaled by s*	$R(s^{**},P_4)$	P_4 scaled by s**

Table 5. $P_5(z) = \prod_{k=1}^{n.} (z-10-10^{-k})$

	$P_5(z)$				$R(s^{**},P_5)$					$R(s^{*},P_5)$					
DEG	VAR	MRE	NP	NS	s**	VAR	MRE	NP	NS	s*	VAR	MRE	NP	NS	RO
4	.3995E+01	.1327E-18	9	4	.8000E+01	.9696E+00	.1327E-18	9	4	.9972E+01	.7782E+00	.4131E-18	9	5	E
5	.4995E+01	.1178E-12	53	3	.8000E+01	.1288E+01	.1355E-12	39	6	.9978E+01	.1000E+01	.1551E-13	113	3	E
6	.5995E+01	.8853E-07	75	4	.8000E+01	.1589E+01	.8853E-07	75	4	.9981E+01	.1301E+01	.4128E-06	101	4	E
7	.6995E+01	.3192E-04	126	5	.8000E+01	.1929E+01	.3192E-04	126	5	.9984E+01	.1544E+01	.3840E-04	322	5	E
8	.7995E+01	.2268E-03	356	8	.8000E+01	.2230E+01	.2268E-03	355	9	.9986E+01	.1845E+01	.2486E-03	440	11	E
9	.8995E+01	.9385E-03	527	18	.8000E+01	.2582E+01	.9385E-03	527	18	.9988E+01	.2100E+01	.7938E-03	537	10	E
10	.9996E+01	.5010E-02	529	15	.8000E+01	.2901E+01	.5010E-02	529	15	.9989E+01	.2401E+01	.5052E-02	519	15	E
11	.1104E+02	.4319E-02	667	14	.8000E+01	.3243E+01	.4319E-02	645	14	.9990E+01	.2665E+01	.4497E-02	711	16	E
12	.1207E+02	.6847E-02	783	18	.8000E+01	.3574E+01	.6847E-02	783	18	.9991E+01	.2966E+01	.7018E-02	797	18	E
13	.1311E+02	.1583E-01	822	13	.8000E+01	.3910E+01	.1583E-01	820	15	.9991E+01	.3235E+01	.1212E-01	829	18	E
14	.1414E+02	.1683E-01	1015	18	.8000E+01	.4250E+01	.1678E-01	829	16	.9992E+01	.3536E+01	.1708E-01	1013	24	E
15	.1517E+02	.2261E-01	1038	21	.8000E+01	.4581E+01	.2312E-01	858	19	.9993E+01	.3809E+01	.8559E-02	262	29	E
16	.1620E+02	.2963E-01	1077	28	.8000E+01	.4928E+01	.2975E-01	1029	32	.9993E+01	.4110E+01	.3494E-01	1025	26	E
4	.3995E+01	.1106E-18	8	4	.8000E+01	.9696E+00	.1106E-18	8	4	.9972E+01	.7782E+00	.2885E-18	8	5	D
5	.4995E+01	.4822E-13	52	3	.8000E+01	.1288E+01	.1217E-12	38	6	.9978E+01	.1000E+01	.4765E-14	112	3	D
6	.5995E+01	.3622E-07	74	4	.8000E+01	.1589E+01	.3622E-07	74	4	.9981E+01	.1301E+01	.8455E-07	100	4	D
7	.6995E+01	.3456E-04	125	5	.8000E+01	.1929E+01	.3456E-04	125	5	.9984E+01	.1544E+01	.1145E-04	321	5	D
8	.7995E+01	.1811E-03	355	8	.8000E+01	.2230E+01	.1811E-03	354	9	.9986E+01	.1845E+01	.2550E-03	439	11	D
9	.8995E+01	.9314E-03	526	18	.8000E+01	.2582E+01	.9314E-03	526	18	.9988E+01	.2100E+01	.8451E-03	536	10	D
10	.9996E+01	.1306E-01	528	15	.8000E+01	.2901E+01	.1306E-01	528	15	.9989E+01	.2401E+01	.1332E-01	518	15	D
11	.1104E+02	.3621E-02	666	14	.8000E+01	.3243E+01	.3621E-02	644	14	.9990E+01	.2665E+01	.4765E-02	710	16	D
12	.1207E+02	.9184E-02	782	18	.8000E+01	.3574E+01	.9184E-02	782	18	.9991E+01	.2966E+01	.6289E-02	796	18	D
13	.1311E+02	.1301E-01	821	13	.8000E+01	.3910E+01	.1301E-01	819	15	.9991E+01	.3235E+01	.1895E-01	828	18	D
14	.1414E+02	.1734E-01	1014	18	.8000E+01	.4250E+01	.1734E-01	828	16	.9992E+01	.3536E+01	.1743E-01	1012	24	D
15	.1517E+02	.2239E-01	1037	21	.8000E+01	.4581E+01	.2239E-01	857	19	.9993E+01	.3809E+01	.8540E+00	261	29	D
16	.1620E+02	.2962E-01	1076	28	.8000E+01	.4928E+01	.29[illegible]2E-01	1028	32	.9993E+01	.4110E+01	.2817E-01	1024	26	D

DEG	Degree of $P_5(z)$	VAR	Variation
MRE	Maximum relative error	RO	Refinement options: E:enabled, D:disabled
NP	Number of primary Laguerre s steps	NS	Number of secondary Laguerre s steps
s*	Optimal scale factor	s**	2^j with j minimizing $\|s^* - 2^j\|$
$R(s^*,P_5)$	P_5 scaled by s*	$R(s^{**},P_5)$	P_5 scaled by s**

Table 6. $P_6(z) = \prod_{k=1}^{n} (z-10k)^{\lfloor \frac{1}{2}k+1 \rfloor}$

	P6(z)				R(s**,P6)					R(s*,P6)					
DEG	VAR	MRE	NP	NS	s**	VAR	MRE	NP	NS	s*	VAR	MRE	NP	NS	RO
5	.6556E+01	.2911E-07	140	21	.1600E+02	.1374E+01	.2911E-07	140	21	.2048E+02	.1053E+01	.8793E-13	50	5	E
8	.1136E+02	.1643E-04	418	9	.3200E+02	.2283E+01	.1643E-04	252	9	.2632E+02	.1944E+01	.6080E-08	236	10	E
11	.1646E+02	.2935E-07	615	19	.3200E+02	.2860E+01	.2935E-07	615	19	.3136E+02	.2813E+01	.3282E-07	318	13	E
15	.2357E+02	.7499E-03	811	33	.3200E+02	.4552E+01	.7499E-03	538	30	.3728E+02	.4021E+01	.6358E-05	1111	27	E
19	.3095E+02	.7929E-04	1047	23	.3200E+02	.6520E+01	.6488E-04	1053	31	.4257E+02	.5243E+01	.9505E-04	1350	23	E
24	.4047E+02	.1631E-01	1801	35	.6400E+02	.8306E+01	.1631E-01	1356	42	.4855E+02	.6788E+01	.1930E-01	1846	41	E
29	.5024E+02	.7836E-01	2228	54	.6400E+02	.9434E+01	.7836E-01	2054	58	.5400E+02	.8329E+01	.2557E-01	2100	53	E

DEG	Degree of $P_6(z)$	VAR	Variation
MRE	Maximum relative error	RO	Refinement options; E:enabled, D:disabled
NP	Number of primary Laguerre's steps	NS	Number of secondary Laguerre's steps
s*	Optimal scale factor	s**	2^j with j minimizing $\lvert s^* - 2^j \rvert$
R(s*,P6)	P_6 scaled by s*	R(s**,P6)	P_6 scaled by s**

Table 7.. Coefficients associated with $P_4(z) = \prod_{k=1}^{20} (z-k)$

k	$P_4(z)$	$R(s^{**},P_4)$	$R(s^*,P_4)$
0	.2432902008176640000000000000D+19	.21102061141675676481099799280D+01	.10000000000000150406323700583D+01
1	-.8752948036761600000000000000D+19	-.60735777777350180895155062899D+02	-.29876929629942747583132653210D+02
2	.1380375975364070400000000000D+20	.76626259524429940483969403430D+03	.39127782603723003537813720100D+03
3	-.1287093124515098880000000000D+20	-.57158416866935851885500596840D+04	-.30297308284534932020482928530D+04
4	.8037811822645051776000000000D+19	.28556044009935461417626356710D+05	.15712241330028276473413727840D+05
5	-.3599979517947607200000000000D+19	-.10231757181451278029271634300D+06	-.58439510184274616723673975920D+05
6	.1206647803780373360000000000D+19	.27435994611104737487039528790D+06	.16266469190849034594543996610D+06
7	-.3113336431613906400000000000D+18	-.56631259787789647816680371760D+06	-.34853392149159570557547437820D+06
8	.6303081209929489600000000000D+17	.91721903444405901245772838590D+06	.58597371126498794073730428270D+06
9	-.1014229986551145000000000000D+17	-.11807191029087931383401155470D+07	-.78301107965609731635642660420D+06
10	.1307535010540395000000000000D+16	.12177368724163575097918510440D+07	.83828369288513485802522291160D+06
11	-.1355851828995300000000000000D+15	-.10101883329416066408157348630D+07	-.72186527986021735056635473340D+06
12	.1131027699538100000000000000D+14	.67414504262095689773559570310D+06	.50006147139990548727353608280D+06
13	-.7561111845000000000000000000D+12	-.36054190850257873535156250000D+06	-.27761443637437502601568605880D+06
14	.4017177163000000000000000000D+11	.15324314739227294921875000000D+06	.12248519560137280463585127490D+06
15	-.1672280820000000000000000000D+10	-.51033960571289062500000000000D+05	-.42342651929311285821906705790D+05
16	.5332794600000000000000000000D+08	.13019518066406250000000000000D+05	.11213209419490678984758688340D+05
17	-.1256850000000000000000000000D+07	-.24547851562500000000000000000D+04	-.21946474105433440933216644760D+04
18	.2061500000000000000000000000D+05	.32210937500000000000000000000D+03	.29893094665190139686538056720D+03
19	-.2100000000000000000000000000D+03	-.26250000000000000000000000000D+02	-.25287917378333853631200258000D+02
20	.1000000000000000000000000000D+01	.10000000000000000000000000000D+01	.10000000000000000000000000000D+01
VAR:	.19139997392043937907146755610D+02	.60855534564389301976916613060D+01	.59233910179195561340748099610D+01

s**=.80000000000000000000000000000D+01 s*=.83043612037392808527442150810D+01

s^* Optimal scale factor
$R(s^*,P_4)$ P_4 scaled by s^*
s^{**} 2^j with j minimizing $|s^* - 2^j|$
$R(s^{**},P_4)$ P_4 scaled by s^{**}