ELSEVIER

Journal of Computational and Applied Mathematics 64 (1995) 247–268

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

An improved Šiljak's algorithm for solving polynomial equations converges quadratically to multiple zeros

J.A. Stolan

Chemical Systems Division, United Technologies, San José, CA 95148, USA

Received 1 February 1994; revised 26 April 1994

Abstract

Šiljak's method provides a globally convergent algorithm for inclusion of polynomial zeros. The solution procedure is formulated as a minimization process of a positive definite function involving the real and imaginary parts of the polynomial. The main objective of this paper is to propose an improved version of Šiljak's algorithm, which exploits the minimizing function to ensure a quadratic convergence to multiple zeros and, at the same time, determine their multiplicity. Time comparisons with other standard zero inclusion methods are provided to demonstrate the efficiency of the proposed improvement of the original algorithm.

Keywords: Polynomials; Zeros; Roots; Algorithms; Multiple zeros

1. Introduction

There are a large number of algorithms for inclusion of polynomial zeros. Each algorithm exhibits a numerical superiority over the others depending on the nature of the zeros and their configurations. By rule, however, all the existing algorithms have problems with convergence and accuracy when multiple zeros are present. As Press et al. stated in [11]:

"Multiple roots, or closely spaced roots, produce the most difficulty for numerical algorithms. For example, $P(x) = (x - a)^2$ has a double real root at $x = a$. However, we cannot bracket the root by the usual technique of identifying neighborhoods where the function changes sign, nor will slope-following methods such as Newton–Raphson work well, because both the function and its derivative vanish at a multiple root. Newton–Raphson may work, but slowly, since large roundoff errors can occur. When a root is known in advance to be multiple, then special methods of attack are readily devised. Problems arise when (as is generally the case) we do not know in advance what pathology a root will display."

SSDI 0377-0427(94)00114-6

Our objective is to propose an improved Šiljak's algorithm for solving polynomials, which addresses this problem by converging quadratically to multiple zeros and, at the same time, provides the information about the zero's multiplicity.

In his book on nonlinear systems, Šiljak [14] described an algorithm for computing zeros of polynomials via a minimization process involving a function of the real and imaginary parts of a given polynomial. A digital computer implementation of the algorithm has been devised by Moore [8], and has appeared subsequently in several software packages. The Hewlett–Packard version of the algorithm has been listed by Jamshidi and Malek-Zavarei [3], and has been commercially available from Sierra Digital Research [15]. The most attractive feature of the algorithm is its global convergence to a zero of a given polynomial, regardless of the configuration and nature of the zeros. While the speed of convergence may vary depending on the multiplicity of zeros, the monotonicity of the convergent process is ensured by its gradient character. Multiple zeros, however, may slow down the convergence process considerably, thus rendering the original algorithm unattractive or impractical.

In this paper, we propose to extract the information about the multiplicity of polynomial zeros from the minimizing function and its derivatives during the convergent process. This information is then used to determine a step-size which ensures a quadratic convergence to a zero being approximated by the solution process.

Finally, we present how this method fits into the broad range of polynomial zero inclusion methods. Comparisons are offered with conventional methods such as Laguerre's algorithm as presented in Press et al. [11], the Jenkins–Traub method used in the IMSL library [4], simultaneous methods as presented in [10], and the eigenvalue method used by Matlab [7]. We shall provide time comparisons of the new algorithm with the standard zero inclusion algorithms to show that it is as fast or faster for general polynomial problems, and significantly faster for polynomials with multiple zeros.

2. The optimization problem

Let us consider a polynomial

$$P(z) = \sum_{k=0}^{n} a_k z^k \tag{2.1}$$

with complex coefficients

$$a_k = b_k + \mathrm{i}c_k, \quad a_n \neq 0. \tag{2.2}$$

By expressing the complex variable z as

$$z = \sigma + \mathrm{i}\omega, \tag{2.3}$$

we can compute the powers of z,

$$z^k = X_k + \mathrm{i}Y_k, \tag{2.4}$$

where $X_k = X_k(\sigma, \omega)$ and $Y_k = Y_k(\sigma, \omega)$ are Šiljak's polynomials [14] defined by recursive formulas,

$$\begin{aligned} X_k &= 2\sigma X_{k-1} - (\sigma^2 + \omega^2) X_{k-2}, \\ Y_k &= 2\sigma Y_{k-1} - (\sigma^2 + \omega^2) Y_{k-2}, \end{aligned} \tag{2.5}$$

and $X_0 = 1$, $X_1 = \sigma$, $Y_0 = 0$, $Y_1 = \omega$.

By substituting (2.3) into (2.1) and using (2.4), we can split the polynomial $P(z)$ into its real and imaginary parts as

$$P = R + \mathrm{i}I, \tag{2.6}$$

where

$$\begin{aligned} R(\sigma, \omega) &= \sum_{k=0}^{n} (b_k X_k - c_k Y_k), \\ I(\sigma, \omega) &= \sum_{k=0}^{n} (c_k X_k + b_k Y_k). \end{aligned} \tag{2.7}$$

Inclusion of zeros of $P(z)$ was formulated in [14] as a minimization problem involving the penalty function

$$V(\sigma, \omega) = R^2(\sigma, \omega) + I^2(\sigma, \omega), \tag{2.8}$$

which has the following properties:

(P1) $V(\sigma, \omega)$ is nonnegative everywhere.
(P2) $V(\sigma, \omega)$ is twice differentiable everywhere.
(P3) The zeros of $V(\sigma, \omega)$ are located at the zeros of $P(z)$.
(P4) The zeros of $V(\sigma, \omega)$ are the only minima of $V(\sigma, \omega)$.

These properties make the minimization formulation attractive, because they guarantee global convergence of the corresponding gradient inclusion algorithms.

In order to determine the minima of $V(\sigma, \omega)$ we compute its gradient

$$\operatorname{grad} V(\sigma, \omega) = \left(\frac{\partial V}{\partial \sigma}, \frac{\partial V}{\partial \omega} \right)^{\mathrm{T}} = 2 \left(R \frac{\partial R}{\partial \sigma} + I \frac{\partial I}{\partial \sigma}, R \frac{\partial R}{\partial \omega} + I \frac{\partial I}{\partial \omega} \right)^{\mathrm{T}}, \tag{2.9}$$

where superscript T denotes transpose, and

$$\frac{\partial R}{\partial \sigma} = \frac{\partial I}{\partial \omega}, \qquad \frac{\partial R}{\partial \omega} = -\frac{\partial I}{\partial \sigma}. \tag{2.10}$$

To compute grad V, we use the relations

$$\begin{aligned} \frac{\partial X_k}{\partial \sigma} &= k X_{k-1}, & \frac{\partial X_k}{\partial \omega} &= -k Y_{k-1}, \\ \frac{\partial Y_k}{\partial \sigma} &= k Y_{k-1}, & \frac{\partial Y_k}{\partial \omega} &= k X_{k-1}, \end{aligned} \tag{2.11}$$

which follow from the formulas given in [14],

$$X_k(\sigma,\omega) = \sum_{j=0}^{k} (-1)^j \binom{k}{2j} \sigma^{k-2j}\omega^{2j},$$
$$Y_k(\sigma,\omega) = \sum_{j=1}^{k} (-1)^{j-1} \binom{k}{2j-1} \sigma^{k-2j+1}\omega^{2j-1}. \tag{2.12}$$

Then,

$$\frac{\partial R}{\partial \sigma} = \frac{\partial I}{\partial \omega} = \sum_{k=1}^{n} k(b_k X_{k-1} - c_k Y_{k-1}),$$
$$\frac{\partial R}{\partial \omega} = -\frac{\partial I}{\partial \sigma} = -\sum_{k=1}^{n} k(c_k X_{k-1} + b_k Y_{k-1}), \tag{2.13}$$

which allows for fast computations of grad V via recursive formulas (2.5) for any fixed σ and ω.

Relying on the straightforward computation of V, grad V, and the magnitude of grad V,

$$\| \operatorname{grad} V \| = \sqrt{\left(\frac{\partial V}{\partial \sigma}\right)^2 + \left(\frac{\partial V}{\partial \omega}\right)^2}, \tag{2.14}$$

we can compute zeros of $P(z)$ using the original Šiljak's algorithm:

Algorithm 2.1.

Step 1: Pick a starting point.

Step 2: Compute function V, grad V, and $\| \operatorname{grad} V \|$.

Step 3: Travel a distance $2V/\| \operatorname{grad} V \|$ along the gradient direction.

Step 4: Evaluate the value V at the new point. If the new value is less than the old proceed to Step 5, otherwise reduce the distance along the gradient and try again.

Step 5: If the distance traveled is less than the desired zero tolerance, stop. Otherwise, set the located point as the new starting point and go to Step 2.

Our objective in this paper is to improve this algorithm in an essential way by extracting information from the penalty function regarding multiplicity of the polynomial zeros. Before we address the multiplicity issue, we shall establish convergence properties of the solution process using some well-known facts regarding gradient procedures and algorithms [2].

3. Convergence

Let us form a gradient dynamic system

$$S: \quad \dot{\sigma} = -\frac{\partial V}{\partial \sigma},$$
$$\dot{\omega} = -\frac{\partial V}{\partial \omega}. \tag{3.1}$$

By using (2.9), we can rewrite S in terms of the real and imaginary parts of the polynomial $P(z)$ as

$$S: \quad \dot{\sigma} = -2\left(R\frac{\partial R}{\partial \sigma} + I\frac{\partial I}{\partial \sigma}\right),$$
$$\dot{\omega} = -2\left(R\frac{\partial R}{\partial \omega} + I\frac{\partial I}{\partial \omega}\right), \tag{3.2}$$

which, in turn, can be expressed explicitly in terms of σ and ω via X_k and Y_k polynomials by relying on (2.7) and (2.13). Consequently, we obtain right-hand sides of (3.2) as two-variable polynomials in σ and ω.

Let us rewrite S further in a vector form

$$S: \quad \dot{x} = -f(x), \tag{3.3}$$

where $x = (\sigma, \omega)^{\mathrm{T}} \in \mathbb{R}^2$, and $f: \mathbb{R}^2 \to \mathbb{R}^2$ is defined as $f(x) \equiv \mathrm{grad}^{\mathrm{T}} V(x)$. We denote by $x(t; x_0)$ the solutions of (3.3) starting at $t = 0$ and x_0.

Each zero of $P(z)$ is at an equilibrium x^* of S, which is a stationary point of the minimization process of $V(x)$, and is a solution of

$$f(x) = 0. \tag{3.4}$$

To characterize the minimization process, we consider $V(x)$ as a Liapunov function [13], and take the total time derivative

$$\dot{V}(x)_s = -\mathrm{grad}^{\mathrm{T}} V(x) f(x) \tag{3.5}$$

with respect to (3.3). Then, we have:

Theorem 3.1. $\dot{V}(x)_s \leqslant 0$ *for all* $x \in \mathbb{R}^2$, *and* $\dot{V}(x)_s = 0$ *if and only if* $x = x^*$.

Proof. Combining (3.3) and (3.5), we get

$$\dot{V}(x)_s = -\|f(x)\|^2 \tag{3.6}$$

for all $x \in \mathbb{R}^2$, where $\|\cdot\|$ denotes the Euclidean norm defined in (2.14). □

This theorem simply says that if a minimization process $x(t; x_0)$ starts at a sufficiently large $x_0 = (\sigma_0, \omega_0)^{\mathrm{T}}$ it would converge to a bounded region containing the stationary points of $V(x)$, which are the equilibria of S. Once inside the region, the process $x(t; x_0)$ would always descend toward one of the zeros of $V(x)$, since $V[x(t; x_0)]$ is a decreasing function of x unless $x(t; x_0) = x^*$, where $V(x^*) = 0$. The process $x(t; x_0)$ cannot terminate at an equilibrium x^* of S which is not a zero of $V(x)$, because it is not a minimum of $V(x)$. Furthermore, the zeros of $P(z)$ are the only minima of $V(x)$, as stated in (P4). We have:

Corollary 3.2. *The zeros of* $P(z)$ *are all and only asymptotically stable equilibria of* S.

Proof. Obviously the function $V(x)$ is a Liapunov function corresponding to a zero x^* of $P(z)$, that is, in some neighborhood of x^*, we have $V(x) > 0$, $\dot{V}(x)_s < 0$ provided $x \neq x^*$. □

We can further characterize the descent of $x(t;x_0)$ toward the minima of $V(x)$, that is, the zeros of $P(z)$. Let us determine the 2×2 Jacobian matrix $J(x)=(\partial f_i/\partial x_j)$ at x^* to get

$$J(x^*) = -\left[\frac{\partial^2 V(x^*)}{\partial x_i\,\partial x_j}\right] = -H(x^*), \tag{3.7}$$

which is the Hessian matrix $H(x)$ of $V(x)$ with the reversed sign. An important implication of this fact is the following.

Theorem 3.3. *The Jacobian matrix $J(x)$ of $f(x)$ at any equilibrium x^* of S has two negative real eigenvalues.*

Proof. Matrix $J(x)=-H(x)$ is a real symmetric matrix, and its eigenvalues are real. By Corollary 3.2, each x^* is asymptotically stable. Thus, negativity of eigenvalues follows. □

This theorem implies that solutions $x(t;x_0)$ near minima of $V(x)$ cannot exhibit oscillatory behavior: no equilibrium x^* of the linearized system

$$S_L\colon \dot{x} = J(x^*)x \tag{3.8}$$

can be either focus or center; nor can it be an improper node, because $H(x)$ is diagonalizable (it has two distinct eigenvectors). Therefore, the minimization process $x(t;x_0)$ converges globally and monotonically to a zero of $P(z)$.

4. The improved method

Multiple zeros present problems to the original Šiljak method in much the same way they do to most available zero inclusion methods. Our objective is to show how the minimization surfaces can be used to obtain information about the multiplicity of zeros, and how this information can be exploited to devise quadratic convergence to each zero. For example, three-dimensional surface plots of the minimization function $V(\sigma,\omega)$ have been obtained for the following polynomials.

$$\begin{aligned} &P_1(z) = z-(1+\mathrm{i})\colon \quad z_1 = 1+\mathrm{i},\\ &P_2(z) = z^4-1\colon \quad z_1=1,\ z_2=\mathrm{i},\ z_3=-1,\ z_4=-\mathrm{i},\\ &P_3(z) = z^3-(1+\mathrm{i})z^2\colon \quad z_1=1+\mathrm{i},\ z_2=z_3=0. \end{aligned} \tag{4.1}$$

In order to make the zero locations more apparent, the corresponding minimizing functions have been plotted using the log scaling along the V-axis by raising the function to the 0.2 power prior to plotting.

Fig. 1 shows the result of plotting the surface corresponding to the distinct zero $z_1=1+\mathrm{i}$ of $P_1(z)$. Fig. 2 shows the four distinct zeros of $P_2(z)$. The slopes of the surface close to the distinct zeros are considerably sharper than the surface around the multiple zero $z_2=0$ of $P_3(z)$ shown in Fig. 3. This fact indicates that, in order to retain quadratic convergence when multiple zeros are present, we must modify the original gradient scheme described in the preceding section.

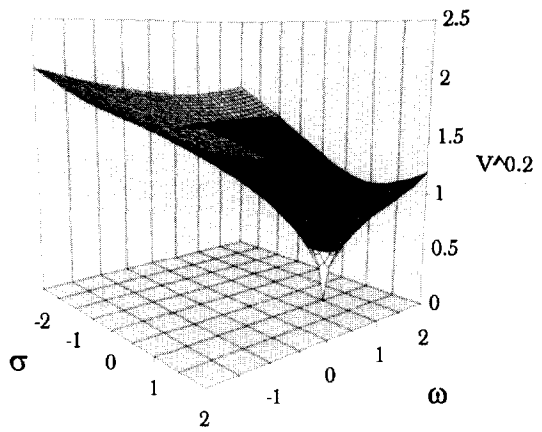

Fig. 1. Šiljak function for a single isolated zero.

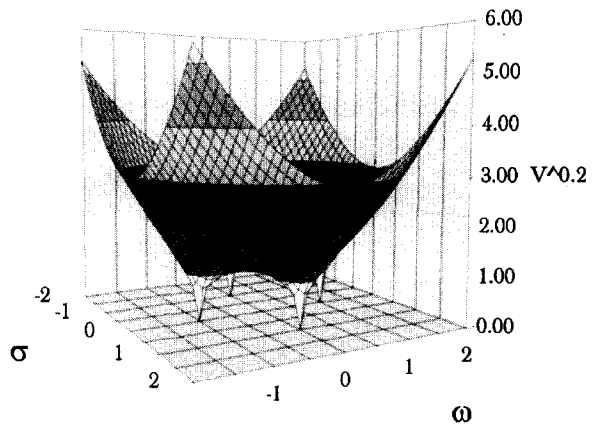

Fig. 2. Šiljak function value for four isolated zeros.

We start with expanding polynomial $P(z)$ about a zero $z_1 = \sigma_1 + i\omega_1$ of multiplicity m,

$$P(z) = a_m(z - z_1)^m + a_{m+1}(z - z_1)^{m+1} + O(z^{m+2}), \tag{4.2}$$

where $a_k = P^{(k)}(z_1)/k!$ The minimizing function can be expressed as

$$V(y) = \|y\|^{2m} \prod_{k=m+1}^{n} \|y - d_k\|^2, \tag{4.3}$$

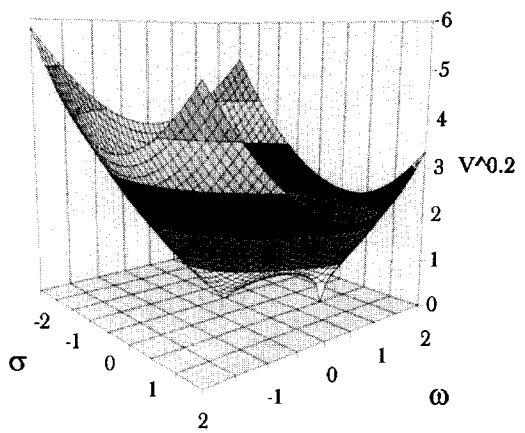

Fig. 3. Šiljak function value for a double zero and an isolated zero.

where $y \in \mathbb{R}^2$ is defined as $y = (\sigma - \sigma_1, \omega - \omega_1)^T$, and $d_k = (\sigma_1 - \sigma_k, \omega_1 - \omega_k)^T$. Using (4.2), we can approximate function $V(y)$ by a positive definite nondecreasing function

$$\bar{V}(y) = \alpha_m \| y \|^{2m} \tag{4.4}$$

throughout a sufficiently small neighborhood of the point (σ_1, ω_1), and $\alpha_m = |a_m|^2$.

The approximation function $\bar{V}(y)$ provides an estimate of the distance $\| y \|$ between the trial point (σ, ω) and the zero location (σ_1, ω_1), as well as the multiplicity of the zero z_1. To see this, let us denote $\bar{g} = \operatorname{grad} \bar{V}(y)$ and introduce the Hessian of $\bar{V}(y)$,

$$\bar{H} = \begin{bmatrix} \dfrac{\partial^2 \bar{V}}{\partial y_1^2} & \dfrac{\partial^2 \bar{V}}{\partial y_1 \partial y_2} \\ \dfrac{\partial^2 \bar{V}}{\partial y_1 \partial y_2} & \dfrac{\partial^2 \bar{V}}{\partial y_2^2} \end{bmatrix}. \tag{4.5}$$

Then, we calculate

$$\begin{aligned} \| \bar{g} \| &= 2m\alpha_m \| y \|^{2m-1}, \\ \frac{\bar{g}^T \bar{H} \bar{g}}{\| \bar{g} \|^2} &= 2m(2m-1)\alpha_m \| y \|^{2m-2}, \end{aligned} \tag{4.6}$$

and by direct computation show that

$$\frac{\bar{V} \| \bar{g} \|}{\| \bar{g} \|^2 - \bar{V}(\bar{g}^T \bar{H} \bar{g} / \| \bar{g} \|^2)} = \frac{2m\alpha_m^2 \| y \|^{4m-1}}{4m^2 \alpha_m^2 \| y \|^{4m-2} - 2m(2m-1)\alpha_m^2 \| y \|^{4m-2}} = \| y \|. \tag{4.7}$$

Similarly, we can estimate the multiplicity of z_1 by the following computation:

$$\frac{\| \bar{g} \|^2}{2(\| \bar{g} \|^2 - \bar{V}(\bar{g}^T \bar{H} \bar{g} / \| \bar{g} \|^2))} = \frac{4m^2 \alpha_m^2 \| y \|^{4m-2}}{2(4m^2 \alpha_m^2 \| y \|^{4m-2} - 2m(2m-1)\alpha_m^2 \| y \|^{4m-2})} = m. \tag{4.8}$$

To derive the improved algorithm and show quadratic convergence, we need to include a higher-order term via a majorizing function

$$\tilde{V}(y) = \alpha_m \| y \|^{2m} + \alpha_{m+1} \| y \|^{2m+1}, \tag{4.9}$$

where $\alpha_{m+1} = 2|a_m \| a_{m+1}|$. Again, this function is positive definite and nondecreasing in a neighborhood of (σ_1, ω_1).

To minimize $\tilde{V}(y)$ we use the following iterative scheme:

$$\| y \|_{t+1} = \| y \|_t - \frac{\tilde{V} \| \tilde{g} \|}{\| \tilde{g} \|^2 - \tilde{V}(\tilde{g}^T \tilde{H} \tilde{g} / \| \tilde{g} \|^2)}, \tag{4.10}$$

which is suggested by (4.7). In (4.10), $\tilde{g} = \operatorname{grad} \tilde{V}$ and $\tilde{H}$ is the Hessian of $\tilde{V}$.

Theorem 4.1. *The iterates* $\| y \|_t$ *of* (4.10) *converge quadratically to zero, that is, there exists a constant* k *such that*

$$\lim_{t \to \infty} \frac{\| y \|_{t+1}}{\| y \|_t^2} = k. \tag{4.11}$$

Proof. By direct computation indicated in (4.10), we obtain first

$$\begin{aligned} \| \tilde{g} \| &= 2m\alpha_m \| y \|^{2m-1} + (2m+1)\alpha_{m+1} \| y \|^{2m}, \\ \frac{\tilde{g}^T \tilde{H} \tilde{g}}{\| \tilde{g} \|} &= 2m(2m-1)\alpha_m \| y \|^{2m-2} + 2m(2m+1)\alpha_{m+1} \| y \|^{2m-1}, \end{aligned} \tag{4.12}$$

and substitute these expressions in (4.10) to get

$$\| y \|_{t+1} = \| y \|_t - \frac{2m\alpha_m^2 \| y \|^{4m-1} + (4m+1)\alpha_m \alpha_{m+1} \| y \|^{4m} + (2m+1)\alpha_{m+1}^2 \| y \|^{4m+1}}{2m\alpha_m^2 \| y \|^{4m-2} + 8m\alpha_m \alpha_{m+1} \| y \|^{4m-1} + (2m+1)\alpha_{m+1}^2 \| y \|^{4m}}. \tag{4.13}$$

Using (4.13), we arrive at (4.11) where

$$k = \frac{(4m-1)\alpha_{m+1}}{2m\alpha_m}. \qquad \square \tag{4.14}$$

Finally, we note that the improved algorithm offers not only the quadratic convergence to a multiple zero (real or complex), but that it provides an estimate of the multiplicity as well.

5. The algorithm and implementation

Before presenting the final algorithm we shall discuss various practical aspects of the proposed method. We recall the expression of the minimizing function $V(y)$ in the neighborhood of a zero z,

having multiplicity m, as

$$V(y) = \|y\|^{2m} \prod_{k=m+1}^{n} \|y - d_k\|^2, \tag{5.1}$$

where, as before, $y = (\sigma - \sigma_1, \omega - \omega_1)^T$ and $d_k = (\sigma_1 - \sigma_k, \omega_1 - \omega_k)$. If the distance $\|y\|$ between the trial point (σ, ω) and the zero location (σ_1, ω_1) is sufficiently smaller than the distance d_k from the zero z_1 to any other zero z_k, $k = m+1, m+2, \ldots, n$, then $V(y)$ can be approximately expressed as

$$V(y) \approx \beta \|y\|^{2m}, \tag{5.2}$$

where β is a positive constant representing the product of squares of the distance between z_1 and z_k's. We can use (5.2) to calculate the "average distance" as

$$\beta_{\text{avg}} = \beta^{1/2(n-m)} = \left(\frac{V(y)}{\|y\|^{2m}} \right)^{1/2(n-m)}, \tag{5.3}$$

which provides several practical insights for implementation of the improved algorithm.

We immediately note that the nearest zero is sufficiently close only if $\beta \ll \beta_{\text{avg}}$. Therefore, if the returned multiplicity m by (4.8) is less than $n-1$ and $2\beta > \beta_{\text{avg}}$, we know we are not in a convergence region of the corresponding zero. Also, in the interest of the rate of convergence, we avoid carrying on the iterations if the value we estimate for m is less than 1 or greater than n, where n is the degree of the polynomial $P(z)$. These facts can be given as a rule of thumb, which indicate that we are not in a suitable region:

$$\begin{aligned} &\text{(i)} && m < 0.9, \\ &\text{(ii)} && m > n + 0.1, \\ &\text{(iii)} && \text{for } m < n-1\text{:} \quad \frac{V(y)}{\|y\|^{2m}} < 2\beta. \end{aligned} \tag{5.4}$$

Another practical matter is the calculation of the Hessian of (4.3). Like the gradient, it can be computed in terms of the polynomials X_k and Y_k. We only derive the formulae

$$\begin{aligned} \frac{\partial^2 V}{\partial \sigma^2} &= 2\left[R\frac{\partial^2 R}{\partial \sigma^2} + \left(\frac{\partial R}{\partial \sigma}\right)^2 \right] + 2\left[I\frac{\partial^2 I}{\partial \sigma^2} + \left(\frac{\partial I}{\partial \sigma}\right)^2 \right], \\ \frac{\partial^2 V}{\partial \sigma\, \partial \omega} &= \frac{\partial^2 V}{\partial \omega\, \partial \sigma} = 2\left[R\frac{\partial^2 R}{\partial \sigma\, \partial \omega} + \left(\frac{\partial R}{\partial \sigma}\right)\left(\frac{\partial R}{\partial \omega}\right) \right] + 2\left[I\frac{\partial^2 I}{\partial \sigma\, \partial \omega} + \left(\frac{\partial I}{\partial \sigma}\right)\left(\frac{\partial I}{\partial \omega}\right) \right], \\ \frac{\partial^2 V}{\partial \omega^2} &= 2\left[R\frac{\partial^2 R}{\partial \omega^2} + \left(\frac{\partial R}{\partial \omega}\right)^2 \right] + 2\left[I\frac{\partial^2 I}{\partial \omega^2} + \left(\frac{\partial I}{\partial \omega}\right)^2 \right] \end{aligned} \tag{5.5}$$

and use (2.5) to compute

$$\frac{\partial^2 R}{\partial \sigma^2} = -\frac{\partial^2 R}{\partial \omega^2} = \frac{\partial^2 I}{\partial \omega\, \partial \sigma} = \frac{\partial^2 I}{\partial \sigma\, \partial \omega} = \sum_{k=2}^{n} k(k-1)(b_k X_{k-2} - c_k Y_{k-2}),$$

$$\frac{\partial^2 I}{\partial \sigma^2} = -\frac{\partial^2 I}{\partial \omega^2} = -\frac{\partial^2 R}{\partial \omega\, \partial \sigma} = -\frac{\partial^2 R}{\partial \sigma\, \partial \omega} = \sum_{k=2}^{n} k(k-1)(b_k Y_{k-2} - c_k X_{k-2}). \tag{5.6}$$

A final issue concerns the handling of the saddle points of function V. While the local minima are all at the zeros of the minimizing function V, the other (unstable) stationary points may exist, where the gradient of V goes to zero. Then, the value of V is used to indicate that we are at or near these points, and to move away from them. A good practical test for the points is that the magnitude of the gradient is always less than the magnitude of the function. The opposite is true for the actual zero locations.

We now have in hand an algorithm which can accurately estimate the distance to single and multiple zeros. Although the algorithm may perform poorly in nonconvergent regions or around saddle points, we also have a test to determine when these situations occur.

Algorithm 5.1.

Step 1: Pick a starting point.

Step 2: Calculate the function value V, the gradient g and the Hessian H.

Step 3: Derive the estimated distance d to the zero cluster from

$$\frac{V\|g\|}{\|g\|^2 - V(g^{\mathrm{T}}Hg/\|g\|^2)}$$

and the estimated order of the zero cluster from

$$\frac{\|g\|^2}{2(\|g\|^2 - V(g^{\mathrm{T}}Hg/\|g\|^2)}.$$

Step 4: If the estimated distance and zero order pass the criteria defined in Eq. (5.4) then travel a distance d in the direction of the gradient. This is the case where a dominant zero or zero cluster generates a good estimate of distance and direction using the improved Šiljak method.

If this condition does not hold, but the magnitude of the gradient is greater than the function value, then we are in a region which is not dominated by a zero cluster, but in which it is safe to perform a Laguerre iteration [9].

If neither of these conditions hold, then we are in a region which is not dominated by a single zero or zero cluster and we are in a region which is close to a saddle point and may not be convergent for the Laguerre method. In such a case we use the gradient to determine the direction of travel, and the function value $V^{1/(2n)}$ where n is the polynomial order, to get the distance. There is a risk that the latter distance will be in error, so we evaluate the function value V at the new point to verify that V has decreased. If not, then the distance is reduced until the new point shows a decrease in V.

Step 5: If the distance travelled is less than the desired zero tolerance, go to Step 6. Otherwise set the new point as the trial point and go to Step 2.

Step 6: Evaluate the multiplicity of the included zero by calculating

$$\frac{\|g\|^2}{2(\|g\|^2 - V(g^{\mathrm{T}}Hg/\|g\|^2))},$$

and rounding to the nearest integer value.

This strategy optimizes the algorithm used. When the estimated zero is in a region which is dominated by a zero or zero cluster then the superior Šiljak algorithm is used. Otherwise the Laguerre method is used, giving better results in regions not dominated by a zero or zero cluster. Convergence is enhanced by a prior check on the gradient and function value, ensuring that the Laguerre algorithm is used only in regions away from saddle points. It is useful to note that the first and second derivatives of the polynomial needed by the Laguerre algorithm are already calculated by the recursive equations used by the Šiljak algorithm.

The strategy in Step 4 will converge to a zero in nearly all cases, but one special case must be checked in order to ensure global convergence. Since there is no verification that the function value V decreases for the Šiljak and Laguerre steps, there are a few cases where the zero estimate may oscillate between two points without converging (in processing hundreds of polynomials, the author has found two). Since this is rare, and since the algorithm usually converges to zeros within three to six iterations, the solution is to begin checking the points for decreasing values of V after a reasonable number of iterations have passed without convergence. The author used ten iterations for algorithm timing tests.

The improved Šiljak algorithm retains the global convergence property of the original algorithm, converges quadratically to zeros of any order, returns the order of each included zero, and converges rapidly for poor initial estimates. In addition, it correctly handles regions where rapid convergence is not guaranteed, greatly enhancing the speed of the overall method. By verifying that the trial point is in a convergence region, it is no longer necessary to check the new point for a reduction in the value of the minimization function, again enhancing the algorithm speed. These enhancements result in an extremely fast algorithm, comparable in speed to Laguerre's method (a method which is not globally convergent). It is significantly faster than other globally convergent algorithms.

In the rest of this section, we will use examples to illustrate the performance of the improved algorithm in four different cases, when the trial point is

(i) close to a distinct zero,
(ii) close to a multiple zero,
(iii) far from all zeros,
(iv) in a potentially nonconvergent region.

The performance will be compared with standard zero inclusion methods.

Example 5.2. Let us consider the polynomial

$$P(z) = z^5 - z^4 - z + 1, \tag{5.7}$$

which has distinct zeros at i, $-$ i and $-$ 1, and a double zero at 1. If we use the Laguerre [9], Halley [1], Newton [12] and improved Šiljak algorithms to isolate the single zero at $-$ 1, we may construct Table 1.

As expected, all four algorithms converge quickly to the isolated zero, improving from two digits of accuracy to better than six digits in three iterations or less.

Example 5.3. If we use the same polynomial $P(z)$ in (5.7) but want to compute the double zero at 1, we obtain Table 2.

In this case we clearly see that the improved Šiljak algorithm is superior to the other methods for the multiple zero case. The improved Šiljak converges to the multiple zero in only three iterations, compared to nine iterations for the second best method. We note that the convergence is indeed quadratic for this multiple zero example.

Example 5.4. One additional feature of the new algorithm is that it is extremely efficient for poor initial estimates. The case of a poor estimate may be viewed as solving a problem with all zeros in

Table 1
Convergence to single zero

	Halley	Laguerre	Newton	Šiljak
Start	(−1.050000, 0.050000)	(−1.050000, 0.050000)	(−1.050000, 0.050000)	(−1.050000, 0.050000)
Iteration 1	(−0.999562, 0.000555)	(−1.000054, −0.00006)	(−1.000907, 0.008896)	(−0.989201, −0.00090)
Iteration 2	(−1.000000, 0.000000)	(−1.000000, 0.000000)	(−0.999844, 0.000035)	(−0.999764, 0.000000)
Iteration 3			(−1.000000, 0.000000)	(−1.000000, 0.000000)

Table 2
Convergence to double zero

	Halley	Laguerre	Newton	Šiljak
Start	(1.050000, 0.050000)	(1.050000, 0.050000)	(1.050000, 0.050000)	(1.050000, 0.050000)
Iteration 1	(1.016650, 0.017521)	(1.013785, 0.013238)	(1.025081, 0.026789)	(0.996145, −0.000163)
Iteration 2	(1.005544, 0.005938)	(1.003793, 0.003607)	(1.012520, 0.013888)	(0.999989, 0.000000)
Iteration 3	(1.001847, 0.001990)	(1.001044, 0.000990)	(1.006248, 0.007073)	(1.000000, 0.000000)
Iteration 4	(1.000616, 0.000665)	(1.000287, 0.000272)	(1.003120, 0.003569)	
Iteration 5	(1.000205, 0.000222)	(1.000079, 0.000075)	(1.001559, 0.001793)	
Iteration 6	(1.000068, 0.000074)	(1.000022, 0.000021)	(1.000779, 0.000899)	
Iteration 7	(1.000023, 0.000025)	(1.000006, 0.000006)	(1.000390, 0.000450)	
Iteration 8	(1.000008, 0.000008)	(1.000001, 0.000002)	(1.000195, 0.000225)	
Iteration 9	(1.000003, 0.000003)	(1.000000, 0.000000)	(1.000097, 0.000113)	
Iteration 10	(1.000001, 0.000001)		(1.000049, 0.000056)	
Iteration 11	(1.000000, 0.000000)		(1.000024, 0.000028)	
Iteration 12			(1.000012, 0.000014)	
Iteration 13			(1.000006, 0.000007)	
Iteration 14			(1.000003, 0.000004)	

a single cluster. In such cases, our initial assumption treats the cluster as a single zero, and trivially that zero is the nearest since there are no other zeros.

Hansen and Patrick [1] have demonstrated that the Laguerre algorithm provides excellent convergence behavior when the initial zero estimate is poor. Below we quote two of their polynomials, confirming their results, then adding the performance of the new algorithm.

For the first polynomial we see that when it is solved starting with a poor estimate, the Laguerre algorithm does much better than Halley. The Laguerre estimate reduces the initial error from almost 1000 to less than 3 in the first step. When the results for the improved Šiljak are added to Hansen and Patrick's result we see that it does even better, reducing the error from 1000 to less than 0.01 in a single step! When the estimate is selected closer to the zeros, there is essentially no difference in performance. Results for the second polynomial are very similar. When starting with a poor initial estimate, the improved Šiljak algorithm is consistently better than the Laguerre algorithm.

The first polynomial is

$$P(z) = z^6 - 6z^5 + 50z^3 - 45z^2 - 108z + 108, \tag{5.8}$$

which has zeros at 1, -2, -2, 3, 3, 3. Table 3 shows the results starting at the initial point at $\sigma = 1000$ and $\omega = 0$, and at the starting point $\sigma = 0$ and $\omega = 0$.

For the second polynomial,

$$P(z) = z^6 - 4z^5 + 190z^3 - 666z^2 + 944z - 600, \tag{5.9}$$

which has zeros at -6, 2, $1 + i$, $1 - i$, $3 + 4i$, $3 - 4i$. Table 4 shows the results starting at the initial point $\sigma = 1000$ and $\omega = 0$, $\sigma = 100$ and $\omega = 100$, and at the starting point $\sigma = 0$ and $\omega = 0$.

Table 4 shows a distinct difference between the rapid convergence for poor initial estimates of Laguerre and Šiljak on the one hand, and Halley on the other. Since Hansen and Patrick list Laguerre as being the best algorithm for poor initial estimates, the performance of the new algorithm is quite impressive.

Example 5.5. Let us once again consider the polynomial

$$P(z) = z^5 - z^4 - z + 1, \tag{5.10}$$

which has single zeros at $-i$, $+i$ and -1, and a double zero at $+1$.

Table 3
Convergence at varying distance from a zero

Starting point	Method	First iteration	Convergence behavior
(1000, 0)	Laguerre	$z_1 = 5.97$	3.000006 in 17 iterations
	Halley	$z_1 = 715$	3.000007 in 36 iterations
	New Šiljak	$z_1 = 1.009$	1.000000 in 3 iterations
(0, 0)	Laguerre	$z_1 = 0.74$	1.000000 in 3 iterations
	Halley	$z_1 = 0.705$	1.000000 in 4 iterations
	New Šiljak	$z_1 = 0.74$	1.000000 in 4 iterations

Table 4
Convergence at varying distance from a zero

Starting point	Method	First iteration	Convergence behavior
(1000, 0)	Laguerre	$z_1 = 4.97$	1.00 + 1.00i in 6 iterations
	Halley	$z_1 = 714$	2.000000 in 20 iterations
	New Šiljak	$z_1 = 0.674$	1.00 − 1.00i in 5 iterations
(100, 100)	Laguerre	$z_1 = 4.54 + 0.54i$	2.00000 in 6 iterations
	Halley	$z_1 = 71.6 + 71.4i$	3.00 + 4.00i in 14 iterations
	New Šiljak	$z_1 = 0.68 - 0.11i$	1.00 − 1.00i in 5 iterations
(0, 0)	Laguerre	$z_1 = 1.3 + 1.8i$	1.00 − 1.00i in 4 iterations
	Halley	$z_1 = 1.15$	2.000000 in 8 iterations
	New Šiljak	$z_1 = 1.3 + 1.8i$	1.00 − 1.00i in 5 iterations

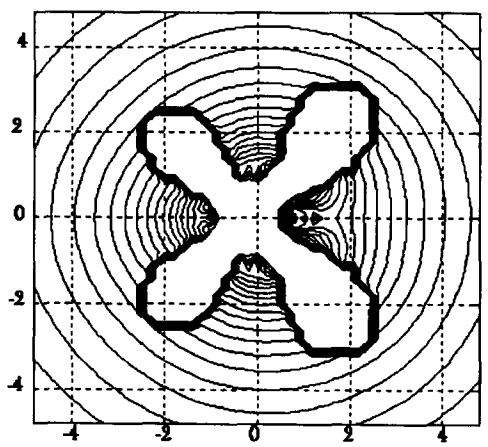

Fig. 4. Identification of region which does not converge on first iteration.

If we apply the rule of thumb given in (5.4) we can evaluate any given point to determine whether it is in a convergent region. If we perform this evaluation over a grid of points we can compute the contour plot shown in Fig. 4. The white X-shaped area in the center is the region which is detected by (5.4) to be nonconvergent. For every point in this area at least one of the test criteria in (5.4) failed. The contour plot for the rest of the graph indicates the amount of improvement in the V estimate for a single Šiljak step starting at the point in question. Every point on the chart was either detected as being potentially nonconvergent, or showed a reduction in the Šiljak function value.

This shows that the rule of thumb provides an important evaluation for improving the speed of the algorithm. While not presented as a rigorous proof, it is apparent that any trial point which passes the convergence criteria given in (5.4) will almost certainly be in a convergence region. As

such, we need not perform Step 4 of Algorithm 2.1. Note that this step has been eliminated from Algorithm 5.1, greatly enhancing the speed of the algorithm once it enters a convergence region.

6. Comparison to modern methods

There is a huge volume of literature relating to polynomial zero inclusion. McNamee [6] has published a bibliography listing over 3000 references to this subject. We would like to clearly show how this method compares to other methods described in this enormous subject area.

The traditional approach to polynomial zero evaluation is to use a selected method to isolate a zero of the polynomial. The polynomial is then "deflated" by performing synthetic division. The polynomial is divided by the zero location. This reduces the order of the polynomial by one, and removes the included zero from the polynomial. This process is repeated until all of the zeros have been found. However, as noted in Press et al. [11]:

"Deflation must, however, be utilized with care. Because each new root is only known with finite accuracy, errors creep into the determination of the coefficients of the successively deflated polynomial. Consequently the roots become more and more inaccurate. It matters a lot whether errors creep in stably (plus or minus a few multiples of the machine precision at each stage) or unstably (erosion of successive significant figures until the results become meaningless). Which behavior occurs depends on just how the root is divided out. Forward deflation, where the new polynomial coefficients are computed in the highest power of x down to the constant term ... turns out to be stable if the root of smallest absolute value is divided out at each stage."

There have been many different methods devised to eliminate this deflation instability. Among the most successful is the Jenkins and Traub [4] method which uses a rigorous procedure which ensures zeros are isolated in the proper order, thus avoiding deflation instability. This method has become one of the standard methods since it is the zero finder used in the IMSL library. A whole class of algorithms have been devised which isolate the zeros in a parallel scheme [10]. These methods are best used as a zero "polishing" step since they are not globally convergent. The most common method is to "polish" the zeros by using the locations found in the original procedure as initial points for a new set of inclusion steps using the original undeflated polynomial.

The method which we propose rests firmly in the traditional method. As each zero of the polynomial is isolated, the location is stored as a tentative zero location, and the polynomial is deflated with the location. Once all of the tentative zero locations have been identified, each zero is polished by using it as a starting point for an inclusion step with the original undeflated polynomial. The overall speed of the algorithm is comparable to the best of the traditional methods, and, as shown in Section 3, it is globally convergent. In addition, the new method has three distinct advantages over previous methods:

(1) As noted in the introduction, nearly all methods have significant problems converging to multiple zeros. As shown in Section 4, the proposed algorithm converges quadratically to zeros of any multiplicity.

(2) The proposed method automatically identifies the multiplicity of each isolated zero. Thus, as multiple zeros are identified, the polynomial may be deflated by several orders at the same time.

(3) One of the most annoying problems with deflation instability occurs when two or more tentative zero locations converge to the same value under polishing. It is not a trivial task to determine if the polishing algorithm has correctly identified a multiple zero, or if deflation instability has caused enough error to place two tentative zeros within the compass of an isolated zero. Since our proposed algorithm returns the multiplicity of each zero, it is easy to determine which of these conditions has occurred.

In order to demonstrate these advantages we have performed time comparisons of the new algorithm with three of the best modern methods. The Jenkins and Traub algorithm (see above for discussion) is one of the standard zero inclusion algorithms. It is the method used by the IMSL library. Another common method is used by Matlab, one of the most popular of the available numerical analysis software packages. The zero inclusion method utilized by this program solves the eigenvalues of a diagonal matrix formed by placing the coefficients of the polynomial into the diagonal matrix. The eigenvalues are the zeros of the polynomial. Finally, Press et al. [11] recommend Laguerre's algorithm used in conjunction with a traditional deflate and polish algorithm. This algorithm has the virtue of being extremely fast, especially when the zero estimate is far from the actual zeros, but it is not globally convergent [12]. (It is interesting to note that Matlab version 6.0 implemented a version of this algorithm as a faster alternative to their eigenvalue method, but removed it, probably because of convergence problems.)

In order to perform the time comparisons, the improved Šiljak algorithm, the Jenkins and Traub algorithm and the Laguerre algorithm were programmed in the "C" language. The program included a timing algorithm accurate to 1 ms. The Matlab program was run directly, entering the polynomial in and then solving using the "roots" function. Timing in this case was limited to 0.05 s resolution. Three classes of polynomials were evaluated:

(1) *Polynomials with isolated zeros*: Each program was timed in its ability to isolate the zeros of the polynomial $z^n - 1 = 0$, with the value of n ranging up to 40. This provided a set of n zeros, all distributed evenly around the unit circle. This ensures that the zeros are isolated from each other by the same amount.

(2) *Polynomial with multiple zeros*: Each program was timed in its ability to isolate the zeros of the polynomial, with the value of n ranging up to 8. This provides a set of n zeros, $n - 1$ of which are all located at the location $1 + 0i$. In addition to timing, each algorithm was evaluated on its ability to accurately isolate the multiple zero.

(3) *A set of difficult polynomials as specified in* [5]: Each algorithm was evaluated on its ability to accurately isolate the zeros of the polynomials.

A good benchmark task for evaluating polynomial zeros is to determine the zero locations for a polynomial which has a number of isolated zeros. Ideally the zeros will not be clustered, and consist of problems with real and imaginary components. A good sample problem is to solve the polynomial $z^n - 1 = 0$, where n is the order of the polynomial. Although this polynomial is very simple, once a complex zero is found, synthetic division will produce an intermediate polynomial of greater complexity and possessing complex coefficients. The success of the algorithms is very easy to judge for this polynomial, since the resultant zeros are spaced evenly around the unit circle.

This problem was solved using each of the algorithms for problems of order 1 to order 40. The results are tabulated in Table 5.

Table 5
Convergence time for isolated zeros

n	Improved Šiljak	Laguerre	Jenkins	Matlab
1	0.004	0.004	0.011	0.05
2	0.013	0.014	0.042	0.05
3	0.038	0.035	0.083	0.11
4	0.046	0.05	0.143	0.11
5	0.073	0.078	0.193	0.16
6	0.108	0.107	0.403	0.22
7	0.134	0.143	0.311	0.22
8	0.195	0.181	0.353	0.28
9	0.216	Fails	0.726	0.33
10	0.275	Fails	0.543	0.44
11	0.324	Fails	1.403	0.60
12	0.471	Fails	0.701	0.72
13	0.566	Fails	0.852	0.80
14	0.531	Fails	0.953	0.87
15	0.602	Fails	1.065	0.91
16	0.572	Fails	1.233	1.030
17	0.717	Fails	1.392	1.06
18	0.816	Fails	1.667	1.09
19	0.970	Fails	1.729	1.130
20	0.950	Fails	2.008	1.35
21	1.012	Fails	2.050	1.31
22	1.223	Fails	4.152	1.48
23	1.105	Fails	5.002	1.52
24	1.252	Fails	4.373	1.620
25	1.442	Fails	4.847	1.74
26	1.579	Fails	5.679	1.77
27	1.464	Fails	5.892	1.93
28	1.783	Fails	5.785	2.21
29	1.932	Fails	8.121	2.30
30	2.316	Fails	8.158	2.49
31	2.299	Fails	8.249	2.71
32	2.638	Fails	8.533	2.87
33	2.428	Fails	10.402	3.00
34	2.716	Fails	9.924	3.32
35	2.613	Fails	13.102	3.45
36	3.192	Fails	11.003	3.79
37	2.772	Fails	14.407	3.98
38	3.251	Fails	18.321	4.10
39	3.416	Fails	17.100	4.18
40	3.258	Fails	15.196	4.19

The Laguerre and improved Šiljak algorithms are the fastest of the algorithms, being very close in calculation time. However the Laguerre algorithm fails badly for this problem unless the starting point is very close to a zero location. The selected starting point for this problem was 2 + i.

Several other observation may be made from the results:

(a) The Laguerre algorithm fails badly for this problem because the search algorithm tends to converge on regions where $P'(X)$ and $P''(X)$ approach 0. This is not typical, and Laguerre will usually converge with times competitive with the improved Šiljak.

(b) While not causing large error, the lack of a polishing step in the Jenkins–Traub algorithm does affect the overall accuracy. For the higher-order problems, Jenkins–Traub typically exhibits 9–10 decimal digits of accuracy compared to 14–15 digits for the other algorithms.

One of the most difficult tasks in evaluating polynomial zeros is the case of multiple zeros. As discussed previously, many of the algorithms exhibit linear convergence for zeros of multiplicity higher than one. If it is known in advance that certain zeros are multiple, then special action can be taken. This is not generally the case.

Two of the algorithms tested do not share this problem. The Jenkins–Traub method uses three stages of polynomial evaluation. The third and final stage exhibits quadratic convergence to multiple zeros. This is possible because special tests are used to ensure that the algorithm will converge prior to entering the third stage. The improved Šiljak algorithm estimates the zero multiplicity at the sampling point as part of its operation. It uses the estimate to ensure quadratic convergence to the zero.

To test performance for the algorithms the polynomial $(z + 1)(z - 1)^n = 0$ was evaluated for orders 1–8. The results are summarized in Table 6.

The results depict two separate effects of the multiple zeros on performance. As previously mentioned, some of the algorithms will not converge well on multiple zeros. In addition, the multiple zero will cause the polynomial value, $P(z)$, to approach the lower bound on numerical accuracy at a significant distance from the zero. For example, a distance 0.001 away from a seventh-order multiple zero will have a polynomial value $P(z) = D\ (0.001)^7$, where D is the complex distance to the other zeros. If $D = 1$ then $P(z)$ will be $1e - 21$, which is less than the accuracy of the double precision arithmetic. This results in round-off error which may cause a considerable increase in calculation time.

Table 6
Time comparison for multiple zeros

Order	Improved Šiljak	Laguerre	Jenkins	Matlab
1	0.010	0.010	0.039	0.05
2	0.024	0.060	0.062	0.28
3	0.040	0.113	0.114	0.33
4	0.078	0.298	0.154	0.44
5	0.084	0.500	0.219	0.60
6	0.102	2.843	0.263	0.86
7	0.119	4.002	0.348	0.90
8	0.113	4.730	0.424	0.94

A summary of the algorithm performance may be deduced from the table:

(1) The improved Šiljak algorithm is the best of all for these problems. The zero estimate process ensures quadratic convergence to multiple zeros. For cases where round-off error may be a problem, the algorithm automatically switches to a method which is globally convergent. Accuracy for the eighth-order problem was better than 3 decimal digits, which was within the computational tolerance of the machine.

(2) The Jenkins–Traub algorithm was a clear second. The third stage process eliminated the linear convergence properties of the other algorithms. Overall accuracy was also 3 decimal digits.

(3) The Laguerre and Matlab algorithm's performance was poor for this problem. Convergence was linear, limiting the accuracy to 1 or 2 decimal digits, and slowing the performance. In addition, the round-off error increased calculation times dramatically for multiplicities greater than five.

We now compare the performance of the five algorithms in resolving zeros of difficult problems. These polynomials have been selected from examples given by Acton, Jenkins, and Traub. For each problem we show the polynomial, its zeros, and give a brief discussion of why it is difficult to resolve. We then present an evaluation of the performance of the various techniques.

Example 6.1.

$$P(z) = (z - 1)(z - 1.01)(z - 0.99)(z - 1 + 0.01\mathrm{i})(z - 1 - 0.01\mathrm{i}).$$

This problem tests an algorithm's ability to isolate five closely spaced zeros in a cluster separated by less than 0.01.

All of the algorithms were able to isolate the zero cluster, except that the original Šiljak failed due to deflation instability at one starting point. The performance times are summarized in Table 7. The results show that the improved Šiljak algorithm was slightly slower than Laguerre's algorithm for this problem.

Example 6.2.

$$P(z) = (z - 1)(z - 1.01)(z - 0.99)(z - 1 + 0.01\mathrm{i})(z - 1 - 0.01\mathrm{i})(z - 5)(z - 6).$$

This problem tests an algorithm's ability to isolate five closely spaced zeros in a cluster separated by less than 0.01 when the polynomial also possesses other zeros away from the zero cluster.

Table 7
Performance for difficult problems

Polynomial	Improved Šiljak	Laguerre	Jenkins	Matlab
Example 6.1	0.108	0.088	0.134	0.05
Example 6.2	0.108	0.145	0.179	0.16
Example 6.3	0.161	0.294	0.283	0.22
Example 6.4	0.037	0.089	0.075	0.05
Example 6.5	0.041	0.056	0.091	0.05

All of the algorithms were able to isolate the zero cluster without error. The performance times are summarized in Table 7. The results show that the improved Šiljak algorithm was the fastest overall.

Example 6.3.

$$P(z) = (z - 1)(z - 1.01)(z - 0.99)(z - 1 + 0.01\mathrm{i})(z - 1 - 0.01\mathrm{i})(z - 5)(z - 5)(z - 6).$$

This problem tests an algorithm's ability to isolate five closely spaced zeros in a cluster separated by less than 0.01 when the polynomial also possesses additional zeros away from the cluster. This problem further complicates the problem by increasing the multiplicity of the additional zeros. The results show that the improved Šiljak algorithm was the fastest overall.

Example 6.4.

$$P(z) = (z - 1)(z - 1)(z - 2)(z - 2).$$

This problem tests an algorithm's ability to isolate two sets of double zeros.

All of the algorithms were able to isolate the double zero. The performance times are summarized in Table 7. The results show that the improved Šiljak algorithm was the fastest overall by a significant margin.

Example 6.5.

$$P(z) = (z - 1)(z - 1)(z - 2 + 0.01\mathrm{i})(z - 2 - 0.01\mathrm{i}).$$

This problem tests an algorithm's ability to isolate a zero cluster and a double zero.

All of the algorithms were able to isolate the zero cluster and multiple zero. The performance times are summarized in Table 7. The results show that the improved Šiljak algorithm was the fastest overall.

7. Conclusions

We have presented an improved technique for using Šiljak's zero finding algorithm. We have shown that this algorithm retains the global convergence property of the original technique while providing quadratic convergence to zeros of any multiplicity. In addition, the algorithm provides a simple method of determining the multiplicity of each zero as it is included. By performing a simple test to determine the convergence region of the trial point, a significant speed increase is achieved, resulting in an algorithm which is much faster than other globally convergent methods. Time trials with standard methods from IMSL, Numerical Recipes and Matlab have been included.

References

[1] E. Hansen and M. Patrick, A family of root finding methods, *Numer. Math.* **27** (1977) 257–269.

[2] M. Hirsch and S. Smale, *Differential Equations, Dynamical Systems and Linear Algebra* (Academic Press, New York, 1974).
[3] M. Jamshidi and M. Malek-Zavarei, *Linear Control Systems: A Computer-Aided Approach* (Pergamon Press, Oxford, 1986).
[4] M. Jenkins and J. Traub, A three-stage variable shift algorithm for polynomial zeros and its relation to generalized Rayleigh iteration, *Numer. Math.* **20** (1970) 252–263.
[5] M. Jenkins and J. Traub, Principles for testing polynomial zerofinding programs, *Proc. Math. Software II*, Purdue Univ. (1974) 84–107.
[6] J. McNamee, A bibliography on roots of polynomials, *J. Comput. Appl. Math.* **47** (1993) 391–394 (including diskette).
[7] Matlab Reference Guide (August 1992) 424–425.
[8] J. Moore, A convergent algorithm for solving polynomial equations, *J. Appl. Math.* **14** (1967) 324–328.
[9] H.J. Orchard, The Laguerre method for finding the zeros of polynomials, *IEEE Trans. Circuits and Systems* **36** (1989) 1377–1381.
[10] M. Petković, *Iterative Methods for Simultaneous Inclusion of Polynomial Zeros*, Lecture Notes in Math. **87** (Springer, New York, 1989).
[11] W. Press, B. Flannery, S. Teukolsky and W. Vetterling, *Numerical Recipes, The Art of Scientific Computing* (Cambridge Univ. Press, Cambridge, 1986) 259–269.
[12] A. Ralston and P. Rabinowitz, *A First Course in Numerical Analysis* (McGraw-Hill, New York, 2nd ed., 1978).
[13] N. Rouche, P. Habets and M. Laloy, *Stability Theory by Lyapunov's Direct Method* (Springer, New York, 1977).
[14] D.D. Šiljak, *Nonlinear Systems* (Wiley, New York, 1969).
[15] Šiljak's polynomial root finder, *Educalc* **42** (1988) 94.