Università di Pisa

Dipartimento di Informatica

Technical Report : TR-96-16

POLYNOMIAL ROOT COMPUTATION BY MEANS OF THE LR ALGORITHM

L. Gemignani

May 16, 1996

ADDR: Corso Italia 40,56125 Pisa,Italy. TEL: +39-50-887111. FAX: +39-50-887226

POLYNOMIAL ROOT COMPUTATION BY MEANS OF THE LR ALGORITHM*

LUCA GEMIGNANI†

Abstract. By representing the LR algorithm of Rutishauser and its variants in a polynomial setting, we derive numerical methods for approximating either all of the roots or a number k of the roots of minimum modulus of a given polynomial $p(t)$ of degree n. These methods share the convergence properties of the LR matrix iteration but, unlike it, they can be arranged to produce parallel and sequential algorithms which are highly efficient expecially in the case where $k << n$.

Key words. Root-finding algorithms, LR matrix iteration.

AMS(MOS) subject classifications. 65H05, 65F15

1. Introduction. The numerical computation of the roots of a given polynomial arises naturally in many important applications. The search for methods which effectively use the power of parallel computers is at the bottom of the increasing interest in the simultaneous computation of polynomial roots.

In this paper, we derive a basic numerical scheme for the computation of the roots of a given polynomial $p(t)$ which is particularly suited to the parallel architectures because of its possibility of extensive vectorization. It is based on the representation in a polynomial setting of the classical LR algorithm of Rutishauser and its variants for Hessenberg matrices [14]. Dekker and Traub [3], [4] explained that certain transformation rules for LR shifts can be identified as iterations involving rational functions whose poles are the eigenvalues of the starting matrix. Jenkins and Traub have developed these ideas into efficient numerical algorithms for computing both a single zero [7] and a quadratic factor [8] of a polynomial. The last mentioned works present effective numerical algorithms which have been implemented in the NAPAK [6] and IMSL [9] libraries. Moreover, since the LR iteration defines a sort of subspace iteration, it is not surprising that our scheme consists of a sort of generalization of certain methods for finding one single zero which are based on the power method. Such methods have been investigated by Sebastião e Silva [15] and Bauer [1]. Stewart [16] gave a systematic theoretical description of these developments. The relation of Jenkins and Traub algorithms to the work of Sebastião e Silva was already observed in a different perspective in the Householder's book [13].

It is a standard approach to compute the roots of a given polynomial $p(t)$ by computing the eigenvalues of a $n \times n$ matrix A having $p(t)$ as characteristic polynomial, for example A is the companion matrix associated with $p(t)$. The LR iteration applied to the starting matrix A produces a sequence of similar matrices $\{A_s\}$ where A_s tends to upper triangular or block triangular form revealing the eigenvalues of A. In the practice

* Supported by 40% and 60% funds of the Progetto Analisi numerica e matematica computazionale of MURST and by G.N.I.M. of C.N.R..

† Dipartimento di Informatica, Università di Pisa, 56125 Pisa, Italy.

of the numerical computation it has been observed that the LR iteration suffers from numerical instabilities arising in the triangular factorization of the matrices A_s. The QR iteration overcomes these numerical difficulties and indeed this is how the MATLAB command *roots* performs its computation.

However these matrix methods are quite expensive; in fact they use order n^2 arithmetical operations or order $n^2/\min\{p,n\}$ parallel steps with p processors per iteration. Further, no reduction both in cost and in storage can be achieved in the case where we want to approximate only a given number $k < n$ of the roots of $p(t)$, typically the k roots of maximum or minimum modulus.

Our scheme implements the LR iteration in a polynomial setting without performing any transition from the polynomial setting to the matrix one. It addresses some of the stability problems of the LR matrix iteration. The proposed approach has also the advantage of decreasing the computational costs. One iteration of our method for finding k roots of $p(t)$ requires order nk arithmetical operations or $\max\{nk/p, 1\}$ parallel steps with order p processors. Moreover, it takes $O(nk)$ memory space.

The convergence properties of our algorithm follow from the convergence theorems for the matrix iterations. These results guarantee a global linear convergence that is governed by the separation ratio, that is the ratio of the smallest of the remaining roots to the considered one . In order to improve the convergence rate, one can consider the technique of shifting. On the other hand, it is clear that shifting corresponds with a rearrangement of the eigenvalues on output and, therefore, it destroys the simultaneous convergence properties. However, the polynomial formulation allows us to use a mixed approach where we accellerate the convergence towards a zero, once it has been isolated from the other ones, without performing a global shift. Similarly, in the case where $p(t)$ is a real polynomial, then complex arithmetic can be avoided by forcing the convergence towards a quadratic factor with complex conjugate zeros.

The paper is organized as follows. Section 2 gives the necessary background on the LR iteration and its polynomial representation. In Section 3 we discuss the properties of the resulting algorithms and we present some numerical experiments.

2. Simultaneous root computation and LR algorithm. Let $p(t)$ be a monic complex polynomial of degree n such that

$$p(t) = \sum_{i=0}^{n-1} p_i t^i + t^n = \prod_{i=1}^{n} (t - t_i), \tag{1}$$

and let us denote as A_1 a $n \times n$ lower Hessenberg matrix with unit superdiagonal entries,

$$A_1 = \begin{pmatrix} a_{1,1}^{(1)} & 1 & & & \\ a_{2,1}^{(1)} & a_{2,2}^{(1)} & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ a_{n-1,1}^{(1)} & a_{n-1,2}^{(1)} & \dots & \dots & 1 \\ a_{n,1}^{(1)} & a_{n,2}^{(1)} & \dots & \dots & a_{n,n}^{(1)} \end{pmatrix} \tag{2}$$

such that

$$p(t) = \det(tI - A_1). \tag{3}$$

The standard LR algorithm with explicit shift defines a sequence of similar matrices by

$$\begin{aligned} A_s - \sigma_s I &= L_s R_s \\ A_{s+1} &= \sigma_s I + R_s L_s, \quad s \geq 1 \end{aligned} \tag{4}$$

where L_s is unit lower triangular, R_s is upper triangular and σ_s is some shift of the origin of the spectrum of A_s.

More recently the following multishift version of (4) has been considered [18]:

$$\begin{aligned} q_s(A_s) &= L_s R_s \\ A_{s+1} &= L_s^{-1} A_s L_s, \quad s \geq 1 \end{aligned} \tag{5}$$

where $q_s(t)$ is some monic polynomial of degree k_s less than n. Writing $q_s(t)$ in factored form, we find that the scheme (5) corresponds with k_s steps of (4) where the shifts are the roots of $q_s(t)$. In this way, a representation of the generalized scheme (5) in a polynomial setting can be obtained by considering a generic step of (4). In the case where $q_s(t)$ is the characteristic polynomial of the trailing $k \times k$ submatrix of A_s with $k \in \{1, 2\}$, the multishift strategy reduces to the classical ones [19].

Under suitable conditions the sequence $\{A_s\}_{s \in N}$, generated by the above LR algorithms, will tend to upper triangular, or at least block triangular, form, yielding information about the eigenvalues. Moreover, since A_1 is a lower Hessenberg matrix with unit superdiagonal entries, then R_1 is an upper bidiagonal matrix with unit superdiagonal entries and, therefore, A_2 is of lower Hessenberg form with unit superdiagonal entries . Hence all A_s are lower Hessenberg matrices with unit superdiagonal entries.

The polynomial vector $(\psi_0^{(s)}(t), \ldots, \psi_{n-1}^{(s)}(t))^T$ such that

$$\psi_0^{(s)}(t) = 1, \quad \psi_i^{(s)}(t) = \det(tI - A_{s,i}), \ 1 \leq i \leq n-1,$$

where $A_{s,i}$ represents the $i \times i$ leading principal submatrix of A_s, satisfies the following equation

$$t \begin{pmatrix} \psi_0^{(s)}(t) \\ \vdots \\ \psi_{n-1}^{(s)}(t) \end{pmatrix} = A_s \begin{pmatrix} \psi_0^{(s)}(t) \\ \vdots \\ \psi_{n-1}^{(s)}(t) \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ p(t) \end{pmatrix}. \tag{6}$$

Similarly, given a polynomial vector $(\psi_0^{(s)}(t), \ldots, \psi_{n-1}^{(s)}(t))^T$ such that $\psi_i^{(s)}(t)$ is a monic polynomial of degree i, there is a unique lower Hessenberg matrix A_s satisfying relation (6) [11].

In this way, since we have

$$t \begin{pmatrix} \psi_0^{(s+1)}(t) \\ \vdots \\ \psi_{n-1}^{(s+1)}(t) \end{pmatrix} = L_s^{-1} A_s L_s \begin{pmatrix} \psi_0^{(s+1)}(t) \\ \vdots \\ \psi_{n-1}^{(s+1)}(t) \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ p(t) \end{pmatrix}, \tag{7}$$

we find that

$$\begin{pmatrix} \psi_0^{(s+1)}(t) \\ \vdots \\ \psi_{n-1}^{(s+1)}(t) \end{pmatrix} = L_s^{-1} \begin{pmatrix} \psi_0^{(s)}(t) \\ \vdots \\ \psi_{n-1}^{(s)}(t) \end{pmatrix}.$$

By replacing A_{s+1} in (7) with $\sigma_s I + R_s L_s$, we obtain that

$$(t-\sigma_s) \begin{pmatrix} \psi_0^{(s+1)}(t) \\ \vdots \\ \psi_{n-1}^{(s+1)}(t) \end{pmatrix} = R_s \begin{pmatrix} \psi_0^{(s)}(t) \\ \vdots \\ \psi_{n-1}^{(s)}(t) \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ p(t) \end{pmatrix}.$$

Since R_s is an upper bidiagonal matrix with unit superdiagonal entries,

$$R_s = \begin{pmatrix} r_1^{(s)} & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & r_n^{(s)} \end{pmatrix},$$

we find that the s-th step of (4) defines the following scheme

$$\begin{cases} (t-\sigma_s)\psi_0^{(s+1)}(t) = r_1^{(s)}\psi_0^{(s)}(t) + \psi_1^{(s)}(t); \\ \cdots\cdots\cdots; \\ \cdots\cdots\cdots; \\ (t-\sigma_s)\psi_{n-1}^{(s+1)}(t) = r_n^{(s)}\psi_{n-1}^{(s)}(t) + p(t). \end{cases} \tag{8}$$

By evaluating (8) at $t = \sigma_s$ there follows that

$$r_i^{(s)} = -\frac{\psi_i^{(s)}(\sigma_s)}{\psi_{i-1}^{(s)}(\sigma_s))}, \quad 1 \le i \le n, \tag{9}$$

where we set $\psi_n^{(s)}(t) = p(t)$. By replacing (9) in (8) we arrive at the following equalities

$$\begin{cases} (t-\sigma_s)\psi_0^{(s+1)}(t) = -(\psi_1^{(s)}(\sigma_s)/\psi_0^{(s)}(\sigma_s))\psi_0^{(s)}(t) + \psi_1^{(s)}(t), \\ \cdots\cdots\cdots\cdots; \\ \cdots\cdots\cdots\cdots; \\ (t-\sigma_s)\psi_{n-1}^{(s+1)}(t) = -(p(\sigma_s)/\psi_{n-1}^{(s)}(\sigma_s))\psi_{n-1}^{(s)}(t) + p(t). \end{cases} \tag{10}$$

In the case where $q_s(t)$ is a quadratic polynomial, say $q_s(t) = (t-\alpha)(t-\beta)$, then by performing 2 steps of (10) with shifts α and β we obtain the following equations:

$$\begin{cases} q_s(t)\psi_0^{(s+1)}(t) = b_{0,0}^{(s)}\psi_0^{(s)}(t) + b_{0,1}^{(s)}\psi_1^{(s)}(t) + \psi_2^{(s)}(t); \\ \cdots\cdots\cdots; \\ \cdots\cdots\cdots; \\ q_s(t)\psi_{n-2}^{(s+1)}(t) = b_{n-2,n-2}^{(s)}\psi_{n-2}^{(s)}(t) + b_{n-2,n-1}^{(s)}\psi_{n-1}^{(s)}(t) + p(t); \\ q_s(t)\psi_{n-1}^{(s+1)}(t) = b_{n-1,n-1}^{(s)}\psi_{n-1}^{(s)}(t) + g^{(s)}(t)p(t); \end{cases} \tag{11}$$

where $b_{i,j}^{(s)}$ are suitable scalars and $g^{(s)}(t) = t + \gamma_s$ is a monic polynomial of degree 1. By evaluating each equality at the zeros of $q_s(t)$, that is, at the shift parameters, we are able to find both the scalars $b_{i,j}^{(s)}$ and γ_s as solutions of 2×2 linear systems. Specifically, for $0 \le j \le n-2$, we have that

$$\begin{cases} b_{j,j}^{(s)} \psi_j^{(s)}(\alpha) + b_{j,j+1}^{(s)} \psi_{j+1}^{(s)}(\alpha) = -\psi_{j+2}^{(s)}(\alpha) \\ b_{j,j}^{(s)} \psi_j^{(s)}(\beta) + b_{j,j+1}^{(s)} \psi_{j+1}^{(s)}(\beta) = -\psi_{j+2}^{(s)}(\beta) \end{cases} \tag{12}$$

which gives both $b_{j,j}^{(s)}$ and $b_{j,j+1}^{(s)}$ in terms of the quotients

$$\frac{\psi_{j+h}^{(s)}(\alpha)}{\psi_{j+h-1}^{(s)}(\alpha)}, \quad \frac{\psi_{j+h}^{(s)}(\beta)}{\psi_{j+h-1}^{(s)}(\beta)}, \quad h = 1, 2.$$

Analogously, for $j = n-1$ we obtain that

$$\begin{cases} b_{n-1,n-1}^{(s)} \psi_{n-1}^{(s)}(\alpha) + \gamma_s p(\alpha) = -\alpha p(\alpha) \\ b_{n-1,n-1}^{(s)} \psi_{n-1}^{(s)}(\beta) + \gamma_s p(\beta) = -\beta p(\beta). \end{cases} \tag{13}$$

The ordering of the shifts, which affects the accuracy of the LR algorithm, is irrelevant to the above iterative schemes.

The iterative schemes (10) and (11) provide for a generalization of the Jenkins and Traub methods. In fact the last equalities of (10) and (11) coincide with the iterations proposed in [7] and [8], respectively. In the basic versions of the iterative schemes (10) and (11), we choose a starting shift $q_1(t)$ of degree k_1, $k_1 \in \{1, 2\}$, and we set $q_s(t) = q_1(t)$, $\forall s \ge 1$; we will refer to this case as to the stationary case. Since the coefficients of $\psi_j^{(s)}(t)$ are rational functions of the coefficients of $\psi_{j+h}^{(0)}(t)$ for $1 \le h \le j-n$, then we observe that for almost any starting polynomial vector $(\psi_0^{(0)}(t), \ldots, \psi_{n-1}^{(0)}(t))^T$ the polynomial sequences $\{\psi_i^{(s+1)}(t)\}$ can be constructed for any $s \in \mathbf{N}$.

A convergence theory for the multishift LR matrix iteration (5) can be found in [18]. This leads to a convergence result for the iterative schemes (10) and (11) in the stationary case whenever $q_s(A_1) = q_1(A_1)$ has eigenvalues of distinct modulus. In [5] the author proved a more general result about the convergence of the polynomial schemes which follow from the multishift LR iteration in the stationary case. The convergence of (10) and (11) has been proved without any assumption about the multiplicity of the zeros of $p(t)$.

By combining these results, we arrive at the next theorem on the convergence of the stationary version of the schemes (10) and (11).

THEOREM 2.1. *Let $q_1(t)$ be a polynomial of degree k_1, $k_1 \in \{1, 2\}$, which satisfies*

$$|q_1(t_1)| \ge |q_1(t_2)| \ge \ldots \ge |q_1(t_h)| > |q_1(t_{h+1})| \ge \ldots |q_1(t_n)|$$

where $t_1, \ldots, t_n$ denote the zeros of $p(t)$. For almost any starting polynomial vector $(\psi_0^{(0)}(t), \ldots, \psi_{n-1}^{(0)}(t))^T$ the schemes (10) and (11) with $q_s(t) = q_1(t)$ do not break down for any $s \in \mathbf{N}$. If we set

$$\rho = \frac{|q_1(t_{h+1})|}{|q_1(t_h)|},$$

then $\psi_h^{(s)}(t)$ *tends to the polynomial* $\prod_{i=1}^{h}(t-t_i)$ *in the following sense: for every* $\tilde{\rho}$ *such that* $\rho < \tilde{\rho} < 1$ *there exists a constant* C *such that*

$$||\psi_h^{(s)}(t) - \prod_{i=1}^{h}(t-t_i)||_\infty \leq C\tilde{\rho}^s,$$

holds for all s.

This theorem says that the iterative schemes (10) and (11) can be used for the simultaneous computation of the roots of a polynomial. To be specific, assume that we are looking for the k roots of $p(t)$ of minimum modulus. Set $q_s(t) = t$ at any iteration s so that the matrix iteration (5) reduces to the standard LR iteration without shifting. In the view of Theorem 2.1, the polynomial $\psi_{n-j}^{(s)}(t)$ tends to $\prod_{i=1}^{n-j}(t-t_i)$ where the zeros of $p(t)$ are numbered according to

$$|t_1| \geq |t_2| \geq \ldots \geq |t_{n-j}| > |t_{n-j+1}| \geq \ldots |t_n|.$$

The computation of last k equalities of the iterative schemes (10) and (11) can be performed at the cost of order nk arithmetical operations or $\max\{nk/p, 1\}$ parallel steps with order p processors by using $O(nk)$ storage. Hence, our approach is computationally interesting in the typical case where we look for a small number k of the roots of a polynomial of high degree n. Further, it overcomes some of the numerical difficulties which affect the accuracy of the standard LR method. In fact, although the occurrence of breakdowns or near-breakdowns in the LR iteration (4) can most severely impact the accuracy of the computed approximations $\psi_{n-i}^{(s+1)}(t)$, the updating procedure employs the original polynomial $p(t)$ and, therefore, poor results produced at a certain step might be corrected in practice in the successive iterations at the cost of increasing the number of iterations.

As usual in the matrix iteration theory, the linear convergence rate can be improved by considering suitable shift strategies. The shifts most commonly used are provided by the eigenvalues of the $l \times l$, $l \in \{1, 2\}$, trailing principal submatrix of the iterated matrix A_s. This strategy can be easily implemented in our polynomial context (see the next section) and the local convergence of the resulting algorithm is quadratic when it converges. However, a global shifting strategy produces a rearrangement of the zeros on output and, therefore, it should be avoided in order to retain the simultaneous convergence properties. In the polynomial context, this important question could be faced by means of the following hybrid approach. At any iteration s we use a fixed global shift $q_1(t)$ as in the stationary case. At the same time, if we assume that $\psi_j^{(s+1)}(t)$ is the actually computed polynomial of highest degree, then we accellerate the convergence of $\psi_j^{(s+1)}(t)$ by performing a single step with a dynamically chosen shift.

In the next section we will present a more detailed description of a numerical algorithm based on the schemes (10) and (11) which implements the above described strategy.

3. Computational Results. We have produced a Fortran implementation of the numerical schemes developed here for computing k roots of a given monic complex

polynomial $p(t)$ of degree n, $n \geq k$. It implements the last k equalities of the basic schemes (10) and (11).

The evaluation of the quotients $\psi_{j+1}^{(s)}(\sigma_s)/\psi_j^{(s)}(\sigma_s)$ is performed as in the Bini's implementation of the Aberth's method [2]. If $|\sigma_s| \leq 1$ then we apply the customary Ruffini-Horner rule for determining both $\psi_{j+1}^{(s)}(\sigma_s)$ and $\psi_j^{(s)}(\sigma_s)$. If, otherwise, $|\sigma_s| > 1$ then we evaluate the reverse polynomials $\hat{\psi}_{j+1}^{(s)}(\sigma_s)$ and $\hat{\psi}_j^{(s)}(\sigma_s)$ of $\psi_{j+1}^{(s)}(\sigma_s)$ and $\psi_j^{(s)}(\sigma_s)$, respectively, at the point $1/\sigma_s$. Then we use the following formula, that can be easily proved by direct inspection,

$$\psi_{j+1}^{(s)}(\sigma_s)/\psi_j^{(s)}(\sigma_s) = \sigma_s \hat{\psi}_{j+1}^{(s)}(1/\sigma_s)/\hat{\psi}_j^{(s)}(1/\sigma_s).$$

In this way we retain the numerical stability of the Ruffini-Horner rule and we avoid possible overflow problems.

There are many important details concerning the shift strategies. As already observed in the previous section, the last equality of (10) defines the well-known Jenkins and Traub method for finding a single zero of $p(t)$. Now, if we assume that σ_s is an acceptable approximation to a root of $p(t)$ then the zeros of the polynomial $p(t) - (p(\sigma_s)/\psi_{n-1}^{(s)}(\sigma_s))\psi_{n-1}^{(s)}(t)$ are obtained by perturbing the zeros of $p(t)$ and, therefore, the computation of $\psi_{n-1}^{(s+1)}(t)$ amounts to a classical deflation process. In the view of the Wilkinson's analysis about the sensitivity of the polynomial roots, we know that the deflation algorithm can be unstable and that, for best results, zeros should be calculated in increasing order of magnitude. Jenkins and Traub confirmed these observations by means of an extensive testing. This provides the motivation for the algorithm we have developed. It suggests that one might find the k roots of minimum modulus of $p(t)$ by setting $q_s(t) = t$ at any iteration s so that the matrix iteration (5) reduces to the standard LR iteration without shifting. In this case Theorem 2.1 says that $\psi_{n-j}^{(s)}(t)$ tends to the polynomial $\prod_{i=1}^{n-j}(t - t_i)$ where the zeros of $p(t)$ are numbered according to

$$|t_1| \geq |t_2| \geq \ldots \geq |t_{n-j}| > |t_{n-j+1}| \geq \ldots |t_n|.$$

The linear convergence is governed by the separation ratio between the considered root and the smallest remaining one. An improvement can be achieved in the following way. To be specific, let us assume that $\psi_{n-k}^{(s)}(t), \ldots, \psi_{n-j}^{(s)}(t)$, $1 \leq j \leq k$, denote the approximations actually computed. Then we check for the convergence of $\psi_{n-j-1}^{(s)}(t)$ and of $\psi_{n-j}^{(s)}(t)$ by determining the 2×2 trailing principal submatrix of the current A_s. This is formed according to the following rules (compare (6)):

$$a_{n-j+1,n-j+1}^{(s)} = \mathrm{lc}(t\psi_{n-j}^{(s)}(t) - \psi_{n-j+1}^{(s)}(t)); \tag{14}$$

$$a_{n-j+1,n-j}^{(s)} = \mathrm{lc}(t\psi_{n-j}^{(s)}(t) - \psi_{n-j+1}^{(s)}(t) - a_{n-j+1,n-j+1}^{(s)}\psi_{n-j}^{(s)}(t)); \tag{15}$$

$$a_{n-j,n-j}^{(s)} = \mathrm{lc}(t\psi_{n-j-1}^{(s)}(t) - \psi_{n-j}^{(s)}(t)), \tag{16}$$

where $\mathrm{lc}(p(t))$ denotes the leading coefficient of $p(t)$. If the condition

$$|a_{n-j+1,n-j+1}^{(s)} - a_{n-j+1,n-j+1}^{(s-1)}| \leq \eta \tag{17}$$

is satisfied, we compute the coefficients of the monic polynomial $g^{(s)}(t)$ such that

$$(t - a^{(s)}_{n-j+1,n-j+1})g^{(s)}(t) = -\frac{(\psi^{(s)}_{n+j+1}(a^{(s)}_{n-j+1,n-j+1})}{\psi^{(s)}_{n+j}(a^{(s)}_{n-j+1,n-j+1}))}\psi^{(s)}_{n+j}(t) + \psi^{(s)}_{n+j+1}(t).$$

Then we chech if

$$|b^{(s)} - a^{(s)}_{n-j+1,n-j+1}| \leq \eta$$

where

$$b^{(s)} = \mathrm{lc}(tg^{(s)}(t) - \psi^{(s)}_{n-j+1}(t)),$$

and, in this case, we finally replace $\psi^{(s)}_{n-j}(t)$ with $g^{(s)}(t)$. If, otherwise, (17) does not hold then we calculate the eigenvalues $\lambda^{(s)}_1$ and $\lambda^{(s)}_2$, $|\lambda^{(s)}_1| \leq |\lambda^{(s)}_2|$, of the 2×2 trailing principal submatrix of the current A_s and we check if

$$|\lambda^{(s)}_1 - \lambda^{(s-1)}_1| \leq \eta$$

and

$$|\lambda^{(s)}_2 - \lambda^{(s-1)}_2| \leq \eta.$$

If so, then we find the coefficients of the polynomials $g^{(s)}_1(t)$ and $g^{(s)}_2(t)$ such that

$$\begin{cases} q_s(t)g^{(s)}_2(t) = b^{(s)}_{n-j-1,n-j-1}\psi^{(s)}_{n-j-1}(t) + b^{(s)}_{n-j-1,n-j}\psi^{(s)}_{n-j}(t) + \psi^{(s)}_{n-j+1}(t); \\ q_s(t)g^{(s)}_1(t) = b^{(s)}_{n-j,n-j}\psi^{(s)}_{n-j}(t) + g^{(s)}(t)\psi^{(s)}_{n-j+1}(t) \end{cases}$$

where $q_s(t) = (t - \lambda^{(s)}_1)(t - \lambda^{(s)}_2)$ and both the scalars $b^{(s)}_{i,j}$ and the polynomial $g^{(s)}(t)$ are determined according to (12) and (13). Now, let us denote as $a^{(s)}$ and $b^{(s)}$, $|a^{(s)}| \leq |b^{(s)}|$, the eigenvalues of the 2×2 matrix determined as in (14), (15), and (16) from the coefficients of $\psi^{(s)}_{n-j+1}(t)$, $g^{(s)}_1(t)$ and $g^{(s)}_2(t)$. In the case where

$$|a^{(s)} - \lambda^{(s)}_1| \leq \eta$$

and

$$|b^{(s)} - \lambda^{(s)}_2| \leq \eta.$$

we replace $\psi^{(s)}_{n-j}(t)$ and $\psi^{(s)}_{n-j-1}(t)$ with $g^{(s)}_1(t)$ and $g^{(s)}_2(t)$, respectively.

There are two possible ways of terminating the iteration for a zero. First, if $a^{(s)}_{n-j+1,n-j}$ becomes negligible then $a^{(s)}_{n-j+1,n-j+1}$ may be regarded as a zero and, therefore, we set $\psi^{(l)}_{n-j+1}(t) = \psi^{(s)}_{n-j+1}(t))$ for any $l \geq s$. Alternatively, if $a^{(s)}_{n-j,n-j-1}$ becomes negligible then the eigenvalues of the 2×2 trailing principal submatrix of the current A_s may be taken to be zeros. In this case we set $\psi^{(l)}_{n-j+1}(t) = \psi^{(s)}_{n-j+1}(t)$ and $\psi^{(l)}_{n-j}(t) = \psi^{(s)}_{n-j}(t)$ for any $l \geq s$. It is clearly difficult to devise an entirely satisfactory

criterion for determining when a sub-diagonal element is negligible. If we assume that the starting polynomials $\psi_j^{(0)}(t)$ are choosen so as to define a balanced matrix, then the numerical experiments with the matrix iterations indicate that a satisfactory criterion is that the sub-diagonal element is small compared with the two neighbouring diagonal entries. For economy of computation we consider the conditions

$$|a^{(s)}_{n-j+1,n-j}| \leq n\epsilon(|a^{(s)}_{n-j+1,n-j+1}| + |a^{(s)}_{n-j,n-j}|), \tag{18}$$

and

$$|a^{(s)}_{n-j,n-j-1}| \leq n\epsilon(|a^{(s)}_{n-j+1,n-j+1}| + |a^{(s)}_{n-j,n-j}|), \tag{19}$$

where ϵ denotes here the machine precision. However, our numerical experiments indicate that the conditions (18) and (19) are too strong for ill-conditioned zeros. Hence our program stops the iteration for a zero even if the correction of the approximation is relatively smaller than $n\epsilon$. The procedure reports failure if any zero takes more than $itmax$ iterations.

Our Fortran code has been tested by performing numerical experiments on a computer using the equivalent of 15 decimal digit floating point arithmetic ($\epsilon = 2^{-53}$). For each computed approximation $\hat{\alpha}_i$, $1 \leq i \leq k$, we calculate the value $\delta_i = np(\hat{\alpha})/p'(\hat{\alpha})$ where $p'(t)$ denotes the first derivative of $p(t)$. A classical result [12] states that there is a zero α_i of $p(t)$ such that

$$|\alpha_i - \hat{\alpha}_i| \leq \delta_i,$$

and, therefore, we define the approximation error as $\Delta = \max\{\delta_i \ : \ 1 \leq i \leq k\}$. If we have $\Delta \geq 10^{-1}$ then we repeat the computation by using different starting polynomials $\psi_j^{(0)}(t)$ and and by replacing η with η^2. If again $\Delta \geq 10^{-1}$ holds then the program reports failure.

We considered both polynomials having random complex coefficients and specific polynomials with ill-conditioned zeros. For each test we reported the number it of iterations per zero and the approximation error Δ. For randomly generated polynomials we reported also the average number it_{av} of iterations per zero and the average approximation error Δ_{av}. We have compared the results of our numerical experiments with those obtained by using the Jenkins and Traub method (JT- algorithm) as implemented in the NAPAK library. The accuracy of the computed approximations is often of the same order but sometimes differencies can be observed. This happens because JT-algorithm incorporates a refining technique based on the Newton method and, moreover, it performs shifts of the origin in order to speed the convergence.

Concerning random polynomials, we considered the following two test sets.

1. The first set consists of polynomials $p(t)$ with coefficients of this form

$$a_1 \times 10^{e_1} + i\ a_2 \times 10^{e_2},$$

where $i^2 = -1$ and a_i and e_i are drawn from the uniform distributions in the interval $[-1, 1]$ and $[-5, 5]$, respectively. The idea of considering such polynomials for testing zerofinding algorithms was proposed by Jenkins and Traub [9].

We generated 100 random polynomials for each of the following values of the degree n, $n \in \{30, 60\}$, and we calculated half of the zeros of $p(t)$. We chose $\eta = 0.5$ and $itmax = 100\ n$. For $n = 30$ our algorithm reports success in 96 cases. In the remaining 4 cases the algorithm reports failure since the number of iterations exceeds the value $itmax$. By determining the zeros of these hard polynomials by means of the $Mathematica^{TM}$ function $NRoots$, we find that in each case the maximum of the moduli of the separation ratios is between 0.999 and 0.9999. For $n = 60$ our algorithm reports success in 98 cases. Again the computation of the roots of the remaining 2 polynomials reveals that some of them have almost the same modulus. A future implementation of our algorithm should include a starting shift of the origin as already used in the Jenkins and Traub algorithms [7], [8] in order to overcome these difficulties. Table 1 below reports the results of our experiments.

Table 1		
n	it_{av}	Δ_{av}
30	53.1	$1.7\text{E}-009$
60	42.4	$8.3\text{E}-010$

2. The second set consists of polynomials of high degree. We generated 100 polynomials with random coefficients having real and imaginary part between -1 and 1 for each of the following values of the degree n, $n \in \{100, 200, 400\}$. For each polynomial we calculated the 10 zeros of minimum modulus. Usually in these cases $p(t)$ has all the zeros clustered in a small circle around the origin and, therefore, a large number of iterations is needed to separate one zero from the others. On the other hand, the zeros are well-conditioned and in fact the conditions (18) and (19) are generally satisfied. Table 2 below reports the results of our experiments. In the cases $n = 100$ and $n = 200$ the computation has been always carried out successfully whereas for $n = 400$ the program reports failure in one case. We choose $\eta = 0.5 \times 10^{-[n/200]}$ and $itmax = 100n$.

Table 2		
n	it_{av}	Δ_{av}
100	52.6	$1.5\text{E}-011$
200	199.5	$1.1\text{E}-011$
400	253.3	$3.2\text{E}-011$

Concerning specific polynomials we considered the following three test sets.

1. The first set consists of the Legendre polynomials $p(t)$ of degree n, $n \in \{10, 20, 30\}$, suitably normalized in order to have leading coefficient equal to 1. Such polynomials have been already used as test polynomials in [2] since they have the condition number of the zeros which grow exponentially with n. The following table summarizes the results obtained in the computation of half of the zeros of

$p(t)$. It reports also the highest condition number *cond* of the computed zeros

$$cond = \max_i \frac{1}{|t_i p'(t_i)|} \sum_{j=0}^{n} |a_j t_i^j|.$$

Table 3			
n	it	Δ	$cond$
10	12.6	1.6E − 014	21.6
20	11.9	6.5E − 010	3.3E + 003
30	15.6	7.5E − 009	3.7E + 006

Observe that the approximation error is about the product of the machine precision and the condition number.

2. The second set consists of monic polynomials $p(t)$ having n zeros equally spaced in the interval $[-2.1, 1.9]$. We considered the cases $n = 10$, $n = 20$ and $n = 30$. These polynomials are taken from [17]. Again the following table reports the results obtained in the computation of half of the zeros of $p(t)$.

Table 4		
n	it	Δ
10	6.2	4.4E − 015
20	12.6	6.3E − 010
30	6.2	3.7E − 011

3. The third set consists of monic polynomials $p(t)$ of degree 20 having the form

$$p(t) = (t - 2^{-9}\ a) \prod_{i=1}^{19} (t - 2^{-i+10}).$$

In these experiments the simultaneous convergence is fully observed. The following table reports the number of iterations and the approximation error of the first 5 zeros of $p(t)$ in the case where $a = 1.001$.

Table 5		
Zero#	it	δ
1	19	5.1E − 044
2	3	4.2E − 013
3	3	1.2E − 016
4	3	6.4E − 016
5	2	2.2E − 012

4. Conclusion. A representation of the LR matrix iteration in a polynomial setting has been established. This provides the means for obtaining a globally convergent scheme for the approximation of a certain number k of the zeros of a polynomial $p(t)$ of degree n. The complexity of our algorithm is linear with respect to n; moreover, the

storage needed is also linear with respect to n. On the contrary, the methods based on the reduction to the eigenvalue computation use $O(n^2)$ storage and, therefore, they are unsuitable for polynomials of high degree. Furthermore, our algorithm has nice computational features in a parallel model of computation because of its possibility of extensive vectorization. The computational results indicate that the algorithm seems to be robust and numerically stable when the zeros are computed in increasing order of magnitude. However, in this way the convergence is governed by the smallest separation ratio and, therefore, it suffers from the presence of clusters. Future research on this topic might include the use of the algorithm proposed in [5] for computing a prime factor as a tool for obtaining an automatic deflation by a factor of degree greater than 2. Investigations of the use of our algorithm in a multiprecision floating point arithmetic would also be interesting.

REFERENCES

[1] F. L. Bauer, *Das Verfahren der Treppeniteration und verwandte Verfahren zur Lösung algebraische Eigenwertprobleme*, ZAMP VIII (1957), pp. 163-203.

[2] D. Bini, *Numerical Computation of Polynomial Zeros by Means of Aberth's Method*, manuscript, 1994.

[3] T. J. Dekker, J. F. Traub, *An Analysis of the Shifted LR Algorithm*, Numer. Math., 17 (1971), pp. 179-188.

[4] T. J. Dekker, J. F. Traub, *The Shifted QR Algorithm for Hermitian Matrices*, Linear Algebra Appl., 4 (1971), pp. 137-154.

[5] L. Gemignani, *Computing a factor of a polynomial by means of multishift LR algorithms*, submitted for publication.

[6] W. W. Hager, *Applied Numerical Linear Algebra*, Prentice-Hall International Editions, Englewood Cliffs, NJ, 1988.

[7] M. A. Jenkins, J. F. Traub, *A Three-Stage Variable Shift Iteration for Polynomial Zeros and its Relation to Generalized Rayleigh Iteration*, Numer. Math., 14 (1970), pp. 256-263.

[8] M. A. Jenkins, J. F. Traub, *A Three-Stage Algorithm for Real Polynomials Using Quadratic Iteration*, SIAM J. Numer. Anal., 7 (1970), pp. 545-566.

[9] M. A. Jenkins, J. F. Traub, *Principles for testing polynomial zerofinding programs*, Tech. Report, Department of Computer Science, Carnegie-Mellon University.

[10] M. A. Jenkins, J. F. Traub, *Program ZRPOLY,* International Mathematics and Statistics Library, Houston, Texas.

[11] J. Kautsky, G. H. Golub, *On the Calculation of Jacobi Matrices*, Linear Algebra Appl., 52/53 (1983), pp. 439-455.

[12] P. Henrici, *Applied and Computational Analysis*, vol. 1, Wiley, New York, 1977.

[13] A. S. Householder, *The Numerical Treatment of a Single Nonlinear Equation*, McGraw-Hill, 1970.

[14] H. Rutishauser, Lectures on Numerical Mathematics, Birkhäuser, Boston, 1990.

[15] J. Sebastião e Silva, *Sur une méthode d'approximation semblable a celle de Graeffe*, Portugal. Math., 2 (1941), pp. 271-279.

[16] G. W. Stewart, *On the companion operator for analytic functions*, Numer. Math., 18 (1971), pp. 26-43.

[17] K. C. Toh, L. N. Trefethen, *Pseudozeros of polynomials and pseudospectra of companion matrices*, Numer. Math., 68 (1994), pp. 403-425.

[18] D. S. Watkins, L. Elsner, *Convergence of Algorithms of Decomposition Type for the Eigenvalue Problem*, Linear Algebra Appl., 143 (1991), pp. 19-47.

[19] J. H. Wilkinson, The Algebraic Eigenvalue Problem, Oxford U. P., 1965.