
Article:
Winkler, J.R. (2014) Structured matrix methods for the computation of multiple roots of a polynomial. Journal of Computational and Applied Mathematics, 272. 449 - 467. ISSN 0377-0427

https://doi.org/10.1016/j.cam.2013.08.032

Structured matrix methods for the computation of multiple roots of a polynomial

Joab R. Winkler [a]

[a] *Department of Computer Science, The University of Sheffield, Regent Court, 211 Portobello, Sheffield S1 4DP, United Kingdom*

j.winkler@dcs.shef.ac.uk

Abstract

This paper considers the application of structured matrix methods for the computation of multiple roots of a polynomial. In particular, the given polynomial $f(y)$ is formed by the addition of noise to the coefficients of its exact form $\hat{f}(y)$, and the noise causes multiple roots of $\hat{f}(y)$ to break up into simple roots. It is shown that structured matrix methods enable the simple roots of $f(y)$ that originate from the same multiple root of $\hat{f}(y)$ to be 'sewn' together, which therefore allows the multiple roots of $\hat{f}(y)$ to be computed. The algorithm that achieves these results involves several greatest common divisor computations and polynomial deconvolutions, and special care is required for the implementation of these operations because they are ill-posed. Computational examples that demonstrate the theory are included, and the results are compared with the results from MULTROOT, which is a suite of MATLAB programs for the computation of multiple roots of a polynomial.

Key words: Roots of polynomials, structured matrix methods

1 Introduction

The computation of the roots of a polynomial is a classical problem in mathematics and many methods have been developed, including the methods of bisection [1], Laguerre [8,11], Bairstow, Græffe and Müller [9], Horner [12], Jenkins and Traub [14], and Newton [16]. These methods have advantages and disadvantages with respect to speed, accuracy and convergence. For example, the methods of Müller and Newton may diverge, and the methods of Horner and Bairstow have good convergence properties, but all methods yield incorrect results when multiple roots are computed because these roots break up into simple roots. This has motivated the development of methods that are explicitly designed for the computation of multiple roots of a polynomial,

that is, methods that do not cause the multiple roots of the polynomial to break up into simple roots [3,4,24–26]. The methods described in these papers require greatest common divisor (GCD) computations and polynomial deconvolutions, and the difficulty of computing multiple roots of a polynomial follows from the ill-posed nature of these operations. In particular, if two polynomials $\hat{f}(y)$ and $\hat{g}(y)$ have a non-constant GCD, their perturbed forms $f(y)$ and $g(y)$, respectively, are coprime with probability one (under very mild assumptions). Similarly, even if $\hat{g}(y)$ is an exact divisor of $\hat{f}(y)$, the deconvolution $f(y)/g(y)$ is a rational function and not a polynomial. It therefore follows that special care is required for the computational implementation of these operations, and it will be shown that structured matrix methods allow numerically robust solutions to be obtained. This numerical difficulty must be compared with the symbolic implementation of these operations on $\hat{f}(y)$ and $\hat{g}(y)$ using exact arithmetic, which does not present difficulties.

There also exist algorithms for the determination of real roots of a polynomial, for example, Sturm sequences, Descartes' rule of signs and continued fractions techniques, but these algorithms solve a problem that differs from the problem considered in this paper. Also, they fail to terminate when multiple roots are considered, unless additional information is provided.

The method whose implementation is described in this paper is similar to the method considered in [3,4,24], and it is based on the method described in [17], pages 65-68. The examples in [3,4,24] do not consider the effect of added noise, and thus an exact polynomial is specified and the errors in the computed results are due to roundoff error only. By contrast, the structured matrix methods used in this work allow the effects of noise to be considered, such that multiple roots of the exact polynomial $\hat{f}(y)$ are computed, even when an inexact (noisy) form $f(y)$ of $\hat{f}(y)$,

$$f(y) = \hat{f}(y) + \delta f(y), \tag{1}$$

is specified, where $\delta f(y)$ is the noise added to $\hat{f}(y)$.

The polynomial root solver MultRoot is a suite of Matlab programs for the computation of multiple roots of a polynomial in the presence of added noise [25,26], but it differs from the work described in this paper because the magnitude of the noise added to the coefficients of $\hat{f}(y)$ is required, such that a threshold on the singular values of the Sylvester matrix of $f(y)$ and its derivative $f^{(1)}(y)$ can be specified. By contrast, knowledge of this noise level is not required for the work described in this paper, which has practical advantages because the noise level may not be known, or it may only be known approximately, or the noise level may vary between the coefficients, such that a threshold cannot be specified *a priori*. In particular, computational experiments showed that if the signal-to-noise ratio is higher than a specified

value, which is a function of the values and multiplicities of the roots, then the errors in the computed roots are small and knowledge of the noise level is not required. The dependence of the computed roots on these parameters has been observed by computational experiments, and not considered analytically, and thus only a qualitative statement, and not a quantitative statement, can be made. Also, computational experiments showed that the errors in the computed roots decrease as the noise level decreases, which is expected. If, however, the noise level is known, it can be used to verify the acceptability, or otherwise, of the computed roots.

The method whose computational implementation is considered in this paper is described in Section 2. This algorithm is explicitly designed for the computation of multiple roots, and its application to a polynomial, all of whose roots are simple, does not yield any advantages with respect to the methods discussed above. It is shown in Section 3 that the method described in this paper has a geometric interpretation based on the pejorative manifolds of a polynomial that has multiple roots, and that this allows an easier understanding of its numerical implementation [15]. Structured and unstructured condition numbers of a multiple root of a polynomial are discussed in Section 4, and it is shown they may differ by several orders of magnitude. This difference in magnitude provides the motivation for the method that implements the polynomial root solver described in this paper, and it forms a clear distinction between this root solver and the root solvers reviewed above.

The method whose implementation is considered in this paper requires that polynomial deconvolutions of the form $h_i(y) = p_{i-1}(y)/p_i(y)$ be considered. It is clear that the ith and $(i+1)$th deconvolutions are coupled, and it is therefore advantageous to consider these deconvolutions simultaneously, such that the coupling is explicitly included in the formulation of the problem, rather than consider them as decoupled (independent) deconvolutions. It is shown in Section 5 that structured matrix methods can be used for the computation of these coupled deconvolutions. Section 6 contains examples of the polynomial root solver described in this paper, and the results are compared with the results obtained from the polynomial root solver MultRoot. Section 7 contains a summary of the paper.

This paper is a continuation of the work in [23], and it is shown in this paper that:

- Structured matrix methods allow multiple roots of an exact polynomial $\hat{f}(y)$ to be computed when an inexact form $f(y)$ of $\hat{f}(y)$ is given.
- The polynomial root solver used to compute multiple roots of $\hat{f}(y)$ when $f(y)$ is specified can be interpreted in terms of the pejorative manifolds of $\hat{f}(y)$. The equations that define the pejorative manifolds of a third order polynomial are derived.

- Structured and unstructured condition numbers of a multiple root of a polynomial are considered. It is shown that they may differ by several orders of magnitude, and that this difference increases as the multiplicity of the root increases.
- A structured matrix method can be used to perform polynomial deconvolution, and it is shown that this method is better than the method of least squares. Specifically, the structured matrix method returns the coefficients of an exact polynomial, but the method of least squares returns the coefficients of a polynomial that is an approximation of a rational function.
- The results of the polynomial root solver described in this paper are compared with the results of the polynomial root solver MULTROOT when the coefficients of the polynomial are perturbed, such that the signal-to-noise ratio is a random variable that spans two orders of magnitude. This property makes the specification of a threshold difficult, and the examples in this paper differ, therefore, from the examples in [23], in which this ratio is constant.

2 A method for the computation of multiple roots of a polynomial

This section describes the method in [17], pages 65-68, whose computational implementation is considered in this paper. Consider the exact polynomial

$$\hat{f}(y) = \sum_{i=0}^{m} \hat{a}_i y^{m-i} = s_1(y) s_2^2(y) s_3^3(y) \cdots s_{r_{\max}}^{r_{\max}}(y), \tag{2}$$

where $s_1(y)$ is the product of all the linear factors of $\hat{f}(y)$, $s_2^2(y)$ is the product of all the quadratic factors of $\hat{f}(y)$, and in general, $s_i^i(y)$ is the product of all the factors of degree i of $\hat{f}(y)$. If $\hat{f}(y)$ does not contain a factor of degree k, then $s_k(y)$ is equal to a constant, which can be assumed to be unity. The method used for the computation of multiple roots, designated as Method MR, of $f(y)$ is shown below.

Method MR: The computation of multiple roots of a polynomial

Input The polynomial $\hat{f}(y)$ that is defined in (2).

Output The roots of $\hat{f}(y)$.

Begin

(1) Define $q_0(y) = \hat{f}(y)$ and perform the GCD computations,

$$\begin{aligned}
q_1(y) &= \mathrm{GCD}\left(q_0(y), q_0^{(1)}(y)\right) = s_2(y)s_3^2(y)s_4^3(y)\cdots s_{r_{\max}}^{r_{\max}-1}(y)\\
q_2(y) &= \mathrm{GCD}\left(q_1(y), q_1^{(1)}(y)\right) = s_3(y)s_4^2(y)s_5^3(y)\cdots s_{r_{\max}}^{r_{\max}-2}(y)\\
q_3(y) &= \mathrm{GCD}\left(q_2(y), q_2^{(1)}(y)\right) = s_4(y)s_5^2(y)s_6^3(y)\cdots s_{r_{\max}}^{r_{\max}-3}(y)\\
&\vdots
\end{aligned}$$

which terminate at $q_{r_{\max}}(y)$, which is a constant.

(2) Compute the polynomials $h_i(y), i = 1, \ldots, r_{\max}$, from the deconvolutions,

$$\begin{aligned}
h_1(y) &= \tfrac{q_0(y)}{q_1(y)} &&= s_1(y)s_2(y)s_3(y)\cdots\\
h_2(y) &= \tfrac{q_1(y)}{q_2(y)} &&= s_2(y)s_3(y)\cdots\\
h_3(y) &= \tfrac{q_2(y)}{q_3(y)} &&= s_3(y)\cdots\\
&\vdots\\
h_{r_{\max}}(y) &= \tfrac{q_{r_{\max}-1}(y)}{q_{r_{\max}}(y)} &&= s_{r_{\max}}(y).
\end{aligned} \tag{3}$$

(3) Compute the polynomials $s_i(y), i = 1, \ldots, r_{\max}$, from the deconvolutions,

$$s_1(y) = \frac{h_1(y)}{h_2(y)}, \qquad s_2(y) = \frac{h_2(y)}{h_3(y)}, \qquad \cdots$$

up to

$$s_{r_{\max}-1}(y) = \frac{h_{r_{\max}-1}(y)}{h_{r_{\max}}(y)}, \qquad s_{r_{\max}}(y) = h_{r_{\max}}(y). \tag{4}$$

(4) Solve the equations

$$s_1(y) = 0, \quad s_2(y) = 0, \quad \cdots \quad , \quad s_{r_{\max}}(y) = 0,$$

in order to determine the roots of $\hat{f}(y)$. In particular, if α_0 is a root of $s_k(y)$, then it is a root of multiplicity k of $\hat{f}(y)$.

Method MR shows that the computation of the multiple roots of $\hat{f}(y)$ can be broken down into the computation of the roots of several polynomials $s_i(y)$ of lower degree, all of whose roots are simple, and the roots of $s_k(y)$ are roots of multiplicity k of $\hat{f}(y)$. An example of this method for the computation of multiple roots of a polynomial is given in [23], which also includes a discussion of the problems that arise when the method is implemented in a floating point environment and the given polynomial is an inexact form $f(y)$ of $\hat{f}(y)$, where

$f(y)$ is defined in (1). It is shown that these problems arise because the two fundamental operations (GCD computations and polynomial deconvolutions) in Method MR are ill-posed, and that the GCD of two exact polynomials must be replaced by an approximate greatest common divisor (AGCD) of two inexact polynomials.

Examination of Method MR shows that the computation of the roots of $\hat{f}(y)$ is broken down into two stages:

Stage 1: Compute the multiplicities m_i of the distinct roots α_i of $\hat{f}(y)$.
Stage 2: Compute the value of each distinct root α_i of $\hat{f}(y)$.

These two stages reveal an important difference between Method MR and other methods that are used to compute the roots of a polynomial. In particular, these other methods usually start with an estimate of a root and then employ an iterative procedure for its refinement. The same procedure is applied to all roots, independent of their multiplicities, and it usually involves the Newton-Raphson iteration, or a variant of it, and thus problems occur at a multiple root because $f^{(1)}(y) = 0$ for these values of y.

The computation of the multiplicities m_i of the distinct roots of $\hat{f}(y)$ before the computation of the values of the roots provides a geometric interpretation of Method MR. This geometric interpretation is considered in Section 3, and it leads to a discussion of structured and unstructured condition numbers of a multiple root of a polynomial.

3 Geometric interpretation

This section provides a geometric interpretation of Method MR, and it is shown that a distinction must be made between its application to an exact polynomial $\hat{f}(y)$ that has multiple roots, and an inexact form $f(y)$ of $\hat{f}(y)$ because it is assumed that the roots of $f(y)$ are simple.

Consideration is given to the determination of the roots of $\hat{f}(y)$, and it is therefore adequate to consider the monic form of $\hat{f}(y)$. If the distinct roots of $\hat{f}(y)$ are $\alpha_i, i = 1, \ldots, p$, and α_i has multiplicity $m_i \geq 1$,

$$\hat{f}(y) = \sum_{i=0}^{m} \hat{a}_i y^{m-i} = \prod_{i=1}^{p} (y - \alpha_i)^{m_i}, \qquad \hat{a}_0 = 1, \qquad \sum_{i=1}^{p} m_i = m, \tag{5}$$

then the coefficients of $\hat{f}(y)$ have

$$m - \sum_{i=1}^{p} (m_i - 1) = p,$$

degrees of freedom. It is assumed the coefficients $\hat{a}_i$ are real, and thus the roots of $\hat{f}(y)$ are real or they form complex conjugate pairs.

It follows that all polynomials of degree m that have p distinct roots with multiplicities $m_i, i = 1, \ldots, p$, lie on a manifold $\mathcal{M}$ of dimension p in $\mathbb{R}^m$. The manifold $\mathcal{M}$ is an example of a pejorative manifold [15], the formal definition of which is now given. In particular, it follows from (5) that a monic polynomial $\hat{t}(y)$ can be written as

$$\hat{t}(y) = \sum_{i=0}^{m} \hat{t}_i y^{m-i} = \prod_{i=1}^{p} (y - \beta_i)^{m_i}, \qquad \hat{t}_0 = 1, \qquad \sum_{i=1}^{p} m_i = m,$$

and thus there exist non-linear functions $w_i(\beta_1, \ldots, \beta_p)$ such that

$$\hat{t}_i = w_i(\beta_1, \ldots, \beta_p), \qquad i = 1, \ldots, m. \tag{6}$$

Definition 3.1 (Pejorative manifold) *Let $\mu = \{m_1, \ldots, m_p\}$ be the set of multiplicities of the polynomial $\hat{t}(y)$. The pejorative manifold $\mathcal{M}_{(\mu)} \subset \mathbb{R}^m$ consists of the set of real coefficients $(\hat{t}_1, \ldots, \hat{t}_m)$ such that $\hat{t}(y)$ has p distinct roots whose multiplicities are given by μ.*

The pejorative manifold, with specified multiplicities $m_i, i = 1, \ldots, p$, can also be defined as the locus of points $\left(\hat{t}_1, \ldots, \hat{t}_m\right) \in \mathbb{R}^m$ that satisfy (6) for all distinct values of $\beta_i, i = 1, \ldots, p$. Definition 3.1 of a pejorative manifold is restricted to a set of points in $\mathbb{R}^m$, but it can be easily extended to polynomials with complex coefficients.

The importance of a pejorative manifold for the computation of multiple roots of a polynomial can be seen by considering the two forms of $\hat{f}(y)$ in (5). In particular, if the inexact monic polynomial $f(y)$ is defined by perturbing each coefficient $\hat{a}_i$ of $\hat{f}(y)$ by a random variable δa_i, $i = 1, \ldots, m$,

$$f(y) = \sum_{i=0}^{m} (\hat{a}_i + \delta a_i)\, y^{m-i}, \qquad \hat{a}_0 = 1, \qquad \delta a_0 = 0, \tag{7}$$

then the multiple roots of $\hat{f}(y)$ break up into simple roots and thus the point with coordinates $\hat{a}_i + \delta a_i, i = 1, \ldots, m$, does not lie on $\mathcal{M}$. It is shown in Section 4 that in this circumstance, the multiple roots of $\hat{f}(y)$ are ill-conditioned and thus their condition numbers are large, that is, a small change in the coefficients of $\hat{f}(y)$ yields a large change in its roots. If, however, the perturbed form of $\hat{f}(y)$ is defined as

$$f(y) = \prod_{i=1}^{p} \Big(y - (\alpha_i + \delta\alpha_i) \Big)^{m_i},$$

then the multiplicities of the roots α_i are preserved, but their values are changed to $\alpha_i + \delta\alpha_i$, and the perturbations $\delta\alpha_i$ define a structured perturbation of $\hat{f}(y)$. Furthermore, the structured condition number associated with these perturbations is small, and the roots α_i are therefore stable, that is, a small structured perturbation to the coefficients of $\hat{f}(y)$ causes a small change in the values of the roots, and $f(y)$ and $\hat{f}(y)$ are represented by nearby points on $\mathcal{M}$.

It follows from this discussion that a distinction must be made between a random perturbation of $\hat{f}(y)$ that causes its multiple roots to break up into simple roots, and the situation in which a structured perturbation of $\hat{f}(y)$ causes the multiple roots of $\hat{f}(y)$ to retain their multiplicities. The large ratio in the magnitudes of the condition numbers of the roots of $\hat{f}(y)$ associated with these two perturbations shows that the multiplicities of the roots of $\hat{f}(y)$ should be determined before their values are computed, that is, the pejorative manifold $\mathcal{M}$ must be identified. The computation of the roots of $\hat{f}(y)$ is therefore restricted to $\mathcal{M}$, which shows the importance of the computation of the integers $m_i, i = 1, \ldots, p$. These integers are calculated by the GCD computations and polynomial deconvolutions in Method MR.

Example 3.1 Consider a cubic polynomial $\hat{f}(y)$ with real roots α_1, α_2 and α_3,

$$\hat{f}(y) = y^3 - (\alpha_1 + \alpha_2 + \alpha_3)y^2 + (\alpha_1\alpha_2 + \alpha_1\alpha_3 + \alpha_2\alpha_3)y - \alpha_1\alpha_2\alpha_3.$$

- If $\hat{f}(y)$ has a cubic root, then $\alpha_1 = \alpha_2 = \alpha_3$, and thus

  $$\hat{f}(y) = y^3 - 3\alpha_1 y^2 + 3\alpha_1^2 y - \alpha_1^3.$$

  The pejorative manifold $\mathcal{M}_{(3)}$ of cubic polynomials that have one real cubic root is the twisted cubic curve

  $$\mathcal{C} : \begin{pmatrix} -3\alpha_1 & 3\alpha_1^2 & -\alpha_1^3 \end{pmatrix}.$$

- If $\hat{f}(y)$ has one double root and one simple root, then $\alpha_1 = \alpha_2 \neq \alpha_3$, and thus

  $$\hat{f}(y) = y^3 - (2\alpha_1 + \alpha_3)y^2 + (\alpha_1^2 + 2\alpha_1\alpha_3)y - \alpha_1^2\alpha_3.$$

  The pejorative manifold $\mathcal{M}_{(2,1)}$ of cubic polynomials that have one real double root and one real simple root is the surface

$$\mathcal{S}: \left(-(2\alpha_1 + \alpha_3) \;\; (\alpha_1^2 + 2\alpha_1\alpha_3) \;\; -\alpha_1^2\alpha_3 \right), \qquad \alpha_1 \neq \alpha_3.$$

The parametric curve $\mathcal{C}$ and parametric surface $\mathcal{S}$ are, respectively,

$$\mathcal{C}: (\, X \; Y \; Z \,) = (\, X(t) \; Y(t) \; Z(t) \,) = (\, -3t \;\; 3t^2 \;\; -t^3 \,), \qquad t \in \mathbb{R}, \tag{8}$$

and

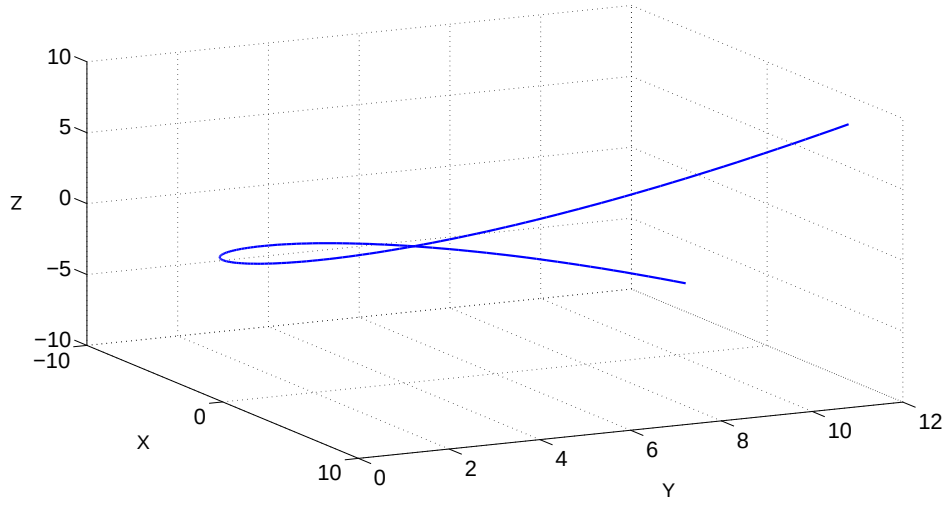
$$\begin{aligned} \mathcal{S}: (\, X \; Y \; Z \,) &= (\, X(s,t) \; Y(s,t) \; Z(s,t) \,) \\ &= (\, -(2s+t) \;\; s^2 + 2st \;\; -s^2 t \,), \quad s \neq t, \quad s, t \in \mathbb{R}, \end{aligned} \tag{9}$$

and they are shown in Figure 1. The closure of the pejorative manifold $\mathcal{M}_{(2,1)}$ is the union of $\mathcal{M}_{(2,1)}$ and $\mathcal{M}_{(3)}$ because this closure is obtained by colliding the roots of $\mathcal{M}_{(2,1)}$. □

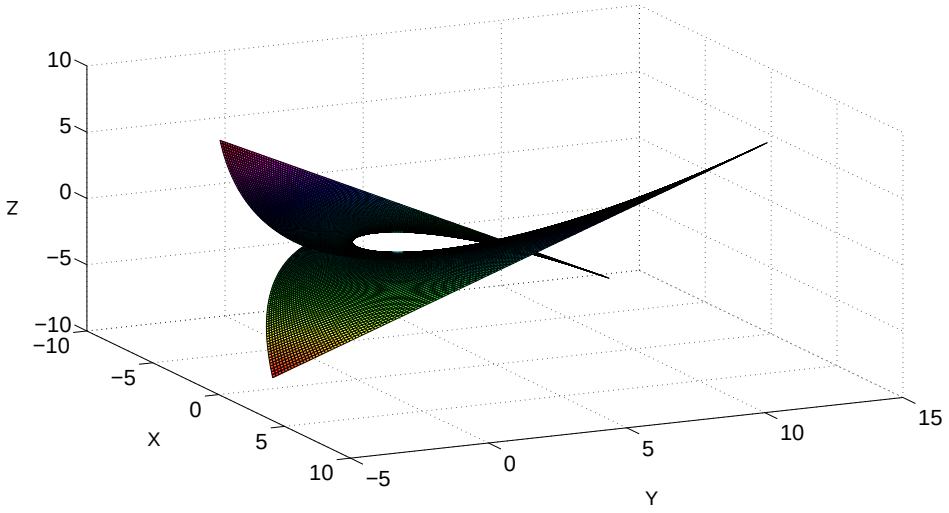
The geometry and interactions between the pejorative manifolds has an interesting structure, some features of which are now considered. Let $\mu = \{m_1, \ldots, m_p\}$ and consider the pejorative manifold $\mathcal{M}_{(\mu)}$. The set of these pejorative manifolds partitions $\mathbb{R}^m$ because each polynomial has a unique root structure. In addition, their closures interact in a natural way because $\overline{\mathcal{M}_{(\mu)}}$ consists of the union of $\mathcal{M}_{(\mu)}$ and all pejorative manifolds that can be obtained by combining the roots of $\mathcal{M}_{(\mu)}$. The closure of a pejorative manifold is an algebraic variety and its structure can be considered using algebraic methods. It contains complicated singularities, but it will not be necessary to understand this geometry for the study of the embeddings of pejorative manifolds.

The feasibility of method MR requires that the embeddings of the pejorative manifolds in $\mathbb{R}^m$ be considered. In particular, the embeddings of different pejorative manifolds should be investigated in order to determine if the root structure of $\hat{f}(y)$ can be recovered from the perturbed polynomial $f(y)$. It is therefore assumed in this section that $f(y)$ and p, that is, the perturbed polynomial and the dimension of the pejorative manifold to which $\hat{f}(y)$ belongs, are known. This section considers the possible determination of the root structure of $\hat{f}(y)$ from $f(y)$ and p. If this root structure can be determined, then it is equal to the multiplicities m_i of the distinct roots of $\hat{f}(y)$.

If two pejorative manifolds of the same dimension are near $f(y)$, then it is plausible that $f(y)$ is derived from a perturbation of a polynomial on either pejorative manifold, but only one of these pejorative manifolds has the same root structure as $\hat{f}(y)$. If, however, only one pejorative manifold is near $f(y)$, then it is likely that $f(y)$ is derived from a perturbation of a polynomial that is represented by a point on this pejorative manifold. Specifically, the closest point on each pejorative manifold of dimension p is a candidate polynomial. If



(a)



(b)

Fig. 1. (a) The curve $\mathcal{C}$, and (b) the surface $\mathcal{S}$, which are defined in Example 3.1. The variables X, Y and Z are defined in (8) and (9) for (a) and (b) respectively.

the polynomial represented by one of these points is significantly nearer than the other polynomials, then it is reasonable to believe that this polynomial has the same root structure as $\hat{f}(y)$. It is therefore desirable to study the geometry and embeddings of pejorative manifolds.

It is reasonable to expect that if p is sufficiently smaller than m, the embeddings of pejorative manifolds are separated in space, because this case represents a very low dimensional space embedded in a much higher dimensional space, that is, $m \gg p$. In this over-determined situation, the algorithm presented in this paper finds a structured perturbation of $f(y)$ that trans-

forms $f(y)$ onto the (typically unique) pejorative manifold $\mathcal{M}_{(\mu)}$ near $f(y)$, from which the values and multiplicities of the roots of $\hat{f}(y)$ can be calculated.

4 Structured and unstructured condition numbers of a polynomial

Section 3 discussed the pejorative manifolds of a polynomial that has one or more multiple roots, and it was stated that a root of a polynomial is stable with respect to perturbations of its coefficients if the multiplicities of its roots are preserved. This situation defines a structured condition number of a root of a polynomial, and it must be compared with the situation that occurs when random perturbations are applied to the coefficients of a polynomial. These perturbations are unstructured and cause multiple roots of a polynomial to break up into simple roots, in which case the perturbed polynomial is not identified with any point on a pejorative manifold. These two situations - a random (unstructured) perturbation and a structured perturbation - are considered in Theorems 4.1 and 4.2 respectively.

Let $g(y, \tilde{\alpha})$ be a polynomial of degree m that is written as the product of two polynomials $g_1(y, \tilde{\alpha})$ and $h(y)$, where the multiplicities of the roots $\tilde{\alpha} = \{\alpha_i\}_{i=1}^{p}$, $\alpha_j \neq \alpha_k$, of $g_1(y, \tilde{\alpha})$ are to be preserved when $g(y, \tilde{\alpha})$ is perturbed, and $h(y)$ contains the other roots of $g(y, \tilde{\alpha})$,

$$g(y, \tilde{\alpha}) = g_1(y, \tilde{\alpha})h(y) = \sum_{i=0}^{m} b_i y^{m-i} = \left(\prod_{i=1}^{p} (y - \alpha_i)^{m_i} \right) h(y).$$

The root α_i has multiplicity m_i, and $h(y)$, which is of degree $n = m - \sum_{i=1}^{p} m_i$, satisfies $h(\alpha_i) \neq 0$. Two forms of $g(y, \tilde{\alpha})$ are needed because the first form (expressed in terms of the coefficients b_i) is used in Theorem 4.1, and the second form (expressed in terms of the roots α_i and the polynomial $h(y)$) is used in Theorem 4.2.

The error model that defines a random perturbation δb_i applied to the coefficient b_i of $g(y, \tilde{\alpha})$ is

$$|\delta b_i| \leq \varepsilon \, |b_i| \,, \qquad i = 0, \ldots, m, \tag{10}$$

where ε^{-1} is the upper bound of the componentwise signal-to-noise ratio. An expression for the condition number of a root of $g(y, \tilde{\alpha})$ when the componentwise perturbations (10) are applied to the coefficients b_i of $g(y, \tilde{\alpha})$ is stated in Theorem 4.1 [18].

Theorem 4.1 *Let the coefficients b_i of $g(y, \tilde{\alpha})$ be perturbed to $b_i + \delta b_i$, where*

(10) is satisfied. The root α_k of $g(y,\tilde{\alpha})$ has multiplicity m_k, and let one of these m_k roots be perturbed to $\alpha_k+\delta\alpha_k$ due to the perturbations in the coefficients. The componentwise condition number of α_k is

$$\begin{aligned}\kappa\left(\alpha_k\right) &= \max_{|\delta b_i|\leq\varepsilon|b_i|}\frac{|\delta\alpha_k|}{|\alpha_k|}\frac{1}{\varepsilon}\\ &= \frac{1}{\varepsilon^{1-\frac{1}{m_k}}}\frac{1}{|\alpha_k|}\left(\frac{m_k!}{\left|g^{(m_k)}\left(\alpha_k,\tilde{\alpha}\right)\right|}\sum_{i=0}^{m}\left|b_i\alpha_k^{m-i}\right|\right)^{\frac{1}{m_k}}. \end{aligned} \tag{11}$$

The next theorem extends Theorem 4.1 to the situation in which the multiplicities of the roots of $g(y,\tilde{\alpha})$ are preserved. This condition requires that a structured perturbation be applied to the coefficients of $g(y,\tilde{\alpha})$ [13].

Theorem 4.2 *The condition number of a root α_k of $g(y,\tilde{\alpha})$, such that the multiplicity m_k of α_k is preserved is*

$$\begin{aligned}\rho(\alpha_k) &= \frac{\Delta\alpha_k}{\Delta g(y,\tilde{\alpha})}\\ &= \frac{\left\|\left(\prod_{i=1}^{p}(y-\alpha_i)^{m_i}\right)h(y)\right\|}{m_k\left|\alpha_k h(\alpha_k)\prod_{j=1,j\neq k}^{p}(\alpha_k-\alpha_j)\right|}\times\\ &\quad\begin{pmatrix}\max\ degree\\ \delta p(y,\tilde{\alpha})\leq(n+p-1)\end{pmatrix}\frac{|\delta p(\alpha_k,\tilde{\alpha})|}{\left\|\left(\prod_{i=1}^{p}\left(y-\alpha_i\right)^{m_i-1}\right)\delta p\left(y,\tilde{\alpha}\right)\right\|}, \end{aligned} \tag{12}$$

where $\delta p(y,\tilde{\alpha})$, whose maximum degree is $(n+p-1)$, is given by

$$\delta p(y,\tilde{\alpha}) = \delta h(y)\prod_{i=1}^{p}(y-\alpha_i) - h(y)\sum_{i=1}^{p}\left((m_i\delta\alpha_i)\prod_{j=1,j\neq i}^{p}(y-\alpha_j)\right), \tag{13}$$

$$\Delta\alpha_k = \frac{|\delta\alpha_k|}{|\alpha_k|} \qquad \textit{and} \qquad \Delta g(y,\tilde{\alpha}) = \frac{\|\delta g(y,\tilde{\alpha})\|}{\|g(y,\tilde{\alpha})\|}.$$

Proof Consider the perturbed polynomial

$$g(y,\tilde{\alpha})+\delta g(y,\tilde{\alpha}) = \left(\prod_{i=1}^{p}\left(y-(\alpha_i+\delta\alpha_i)\right)^{m_i}\right)\Big(h(y)+\delta h(y)\Big),$$

from which it follows that, on retaining only the lowest order terms,

$$
\begin{aligned}
\delta g(y,\tilde{\alpha}) &= \left(\prod_{i=1}^{p}\Big(\,(y-\alpha_i)-\delta\alpha_i\Big)^{m_i}\right)\Big(h(y)+\delta h(y)\Big)\\
&\quad-\left(\prod_{i=1}^{p}(y-\alpha_i)^{m_i}\right)h(y)\\
&= \left(\prod_{i=1}^{p}\Big(\,(y-\alpha_i)^{m_i}-m_i(y-\alpha_i)^{(m_i-1)}\delta\alpha_i\Big)\right)\Big(h(y)+\delta h(y)\Big)\\
&\quad-\left(\prod_{i=1}^{p}(y-\alpha_i)^{m_i}\right)h(y).
\end{aligned}
$$

Thus

$$
\begin{aligned}
\delta g(y,\tilde{\alpha}) &= \delta h(y)\left(\prod_{i=1}^{p}(y-\alpha_i)^{m_i}\right)\\
&\quad -h(y)\sum_{i=1}^{p}\left(\prod_{j=1,j\neq i}^{p}(y-\alpha_j)^{m_j}\right)m_i(y-\alpha_i)^{m_i-1}\delta\alpha_i\\
&= \delta h(y)\left(\prod_{i=1}^{p}(y-\alpha_i)^{m_i-1}\prod_{i=1}^{p}(y-\alpha_i)\right)\\
&\quad -h(y)\sum_{i=1}^{p}\left(\prod_{j=1,j\neq i}^{p}(y-\alpha_j)^{m_j}\right)m_i(y-\alpha_i)^{m_i-1}\delta\alpha_i\\
&= \delta h(y)\left(\prod_{i=1}^{p}(y-\alpha_i)^{m_i-1}\prod_{i=1}^{p}(y-\alpha_i)\right)-h(y)\sum_{i=1}^{p}(m_i\delta\alpha_i)\times\\
&\quad \left(\prod_{j=1,j\neq i}^{p}(y-\alpha_j)^{m_j-1}\prod_{j=1,j\neq i}^{p}(y-\alpha_j)\right)(y-\alpha_i)^{m_i-1}\\
&= \delta h(y)\left(\prod_{i=1}^{p}(y-\alpha_i)^{m_i-1}\prod_{i=1}^{p}(y-\alpha_i)\right)-h(y)\sum_{i=1}^{p}(m_i\delta\alpha_i)\times\\
&\quad \left(\prod_{j=1}^{p}(y-\alpha_j)^{m_j-1}\prod_{j=1,j\neq i}^{p}(y-\alpha_j)\right).
\end{aligned}
$$

The first order perturbation $\delta g(y,\tilde{\alpha})$ is therefore equal to

$$
\delta g(y,\tilde{\alpha}) = \left(\prod_{i=1}^{p}(y-\alpha_i)^{m_i-1}\right)\delta p\,(y,\tilde{\alpha})\,, \tag{14}
$$

where $\delta p\,(y,\tilde{\alpha})$ is defined in (13), and thus

$$
\delta h(y) = \frac{\delta p(y,\tilde{\alpha})+h(y)\sum_{i=1}^{p}\left(m_i\delta\alpha_i\prod_{j=1,j\neq i}^{p}(y-\alpha_j)\right)}{\prod_{i=1}^{p}(y-\alpha_i)}. \tag{15}
$$

The degrees of the numerator and denominator polynomials in this expression for $\delta h(y)$ are $(n+p-1)$ and p respectively. Since $\delta h(y)$ is a polynomial, p of the roots of the numerator polynomial must be $\alpha_k, k = 1, \ldots, p$, and thus the perturbations $\delta\alpha_k$ are given by

$$\delta\alpha_k = -\frac{\delta p(\alpha_k, \tilde{\alpha})}{m_k h(\alpha_k) \prod_{j=1, j\neq k}^{p} (\alpha_k - \alpha_j)}, \qquad k = 1, \ldots, p. \tag{16}$$

The substitution of these expressions for $\delta\alpha_k$ into (15) enables $\delta h(y)$ to be defined, and thus $\delta\alpha_k$ and $\delta h(y)$ are specified for every function $\delta p(y, \tilde{\alpha})$. The condition number (12) follows from (14) and (16). □

The last term on the right hand side of (12) can be cast in terms of matrices and vectors. In particular, if $\delta p_i(\tilde{\alpha})$ are the coefficients of the polynomial $\delta p(y, \tilde{\alpha})$,

$$\delta p(y, \tilde{\alpha}) = \sum_{i=0}^{n+p-1} \delta p_i(\tilde{\alpha}) y^{n+p-1-i},$$

then

$$\delta p(\alpha_k, \tilde{\alpha}) = \sum_{i=0}^{n+p-1} \delta p_i(\tilde{\alpha}) \alpha_k^{n+p-1-i} = \phi^{(n+p)}(\alpha_k)^T \delta p^{(n+p)}(\tilde{\alpha}),$$

where $\delta p_i(\tilde{\alpha})$ are the entries of $\delta p^{(n+p)}(\tilde{\alpha}) \in \mathbb{C}^{n+p}$, and

$$\phi^{(n+p)}(\alpha_k) = \left[\alpha_k^{n+p-1} \;\; \alpha_k^{n+p-2} \;\; \cdots \;\; \alpha_k \;\; 1 \right]^T \in \mathbb{C}^{n+p}.$$

Consider now the denominator of the second term on the right hand side of (12). In particular, the polynomial

$$\left(\prod_{i=1}^{p} (y - \alpha_i)^{m_i - 1} \right) \delta p\,(y, \tilde{\alpha})\,,$$

is of degree $m-1$, and thus

$$\left\| \left(\prod_{i=1}^{p} (y - \alpha_i)^{m_i - 1} \right) \delta p\,(y, \tilde{\alpha}) \right\| = \left\| P \delta p^{(n+p)}(\tilde{\alpha}) \right\|,$$

where $P \in \mathbb{C}^{m \times (n+p)}$ is a Tœplitz matrix whose entries are the coefficients of the polynomial $\prod_{i=1}^{p} (y - \alpha_i)^{m_i - 1}$. The second term on the right hand side of (12) can therefore be written as

$$\frac{|\delta p(\alpha_k,\tilde{\alpha})|}{\left\|\left(\prod_{i=1}^{p}(y-\alpha_i)^{m_i-1}\right)\delta p\,(y,\tilde{\alpha})\right\|}=\frac{\left|\phi^{(n+p)}(\alpha_k)^T\delta p^{(n+p)}(\tilde{\alpha})\right|}{\|P\delta p^{(n+p)}(\tilde{\alpha})\|}$$
$$=\frac{\left|\phi^{(n+p)}(\alpha_k)^T\delta p^{(n+p)}(\tilde{\alpha})\right|}{\left\|(P^TP)^{\frac{1}{2}}\,\delta p^{(n+p)}(\tilde{\alpha})\right\|},$$

which is bounded by

$$\frac{\left|\phi^{(n+p)}(\alpha_k)^T\left(P^TP\right)^{-\frac{1}{2}}\left(P^TP\right)^{\frac{1}{2}}\delta p^{(n+p)}(\tilde{\alpha})\right|}{\left\|(P^TP)^{\frac{1}{2}}\,\delta p^{(n+p)}(\tilde{\alpha})\right\|}\leq\left\|\phi^{(n+p)}(\alpha_k)^T\left(P^TP\right)^{-\frac{1}{2}}\right\|,$$

and thus the structured condition number (12) can also be written as

$$\rho(\alpha_k)\leq\frac{\left\|\left(\prod_{i=1}^{p}(y-\alpha_i)^{m_i}\right)h(y)\right\|}{m_k\left|\alpha_k h(\alpha_k)\prod_{j=1,j\neq k}^{p}(\alpha_k-\alpha_j)\right|}\left\|\phi^{(n+p)}(\alpha_k)^T\left(P^TP\right)^{-\frac{1}{2}}\right\|.$$

It is noted that $\kappa(\alpha_k)$ is a function of the upper bound of the componentwise signal-to-noise ratio ε^{-1}, and that $\rho(\alpha_k)$ is independent of ε^{-1}. Example 4.1 considers the implications of this difference.

Example 4.1 Consider the condition numbers (11) and (12) for the situation $h(y)=1$ and $p=1$, in which case $g(y,\alpha)=(y-\alpha)^m$. If $m\gg 1$ and

$$\left(\frac{m!}{\left|g^{(m)}(\alpha,\alpha)\right|}\sum_{i=0}^{m}\left|b_i\alpha^{m-i}\right|\right)^{\frac{1}{m}}=\left(\sum_{i=0}^{m}\left|b_i\alpha^{m-i}\right|\right)^{\frac{1}{m}}\approx 1,$$

then (11) can be approximated by

$$\kappa\,(\alpha)\approx\frac{1}{\varepsilon}\frac{1}{|\alpha|}. \tag{17}$$

This approximation shows that an increase in ε^{-1} causes an increase in the unstructured condition number of a root of high multiplicity, and the instability of this root therefore increases as ε^{-1} increases.

Consider now the structured condition number $\rho(\alpha)$ for $g(y,\alpha)$. In particular, it follows from (12) and (13) that

$$\rho(\alpha)\leq\frac{\|(y-\alpha)^m\|}{m\,|\alpha|\,\|(y-\alpha)^{m-1}\|}=\frac{\|(y-\alpha)^{m-1}(y-\alpha)\|}{m\,|\alpha|\,\|(y-\alpha)^{m-1}\|}\leq\frac{\|y-\alpha\|}{m\,|\alpha|}\approx\frac{1}{m},$$

if $|\alpha| \gg 1$, which must be compared with (17). It is seen that $\rho(\alpha)$ is independent of ε^{-1} and that it decreases as the multiplicity of α increases. This theoretical result is confirmed by the computational results in [5], where it is shown that the efficiency of the computation of a root increases as its multiplicity increases. □

The structured condition number (12) allows the following theorem to be established [13].

Theorem 4.3 *Consider the polynomial $g(y, \tilde{\alpha})$ when $p = 1$,*

$$g(y, \alpha) = (y - \alpha)^m \, h(y), \qquad h(\alpha) \neq 0,$$

and let $\bar{g}(y, \alpha)$ be the neighbouring polynomial,

$$\bar{g}(y, \alpha) = (y - \alpha)^m \left(h(y) - h(\alpha) \right),$$

which has a root α of multiplicity at least $m+1$. The relative difference between $g(y, \alpha)$ and $\bar{g}(y, \alpha)$ is inversely proportional to the condition number $\rho(\alpha)$ of the root $y = \alpha$ of $g(y, \alpha)$.

Proof Since $p = 1$, it follows from (13) that

$$\delta p(y, \alpha) = \delta h(y) \, (y - \alpha) - m h(y) \delta \alpha,$$

and thus from (12),

$$\rho(\alpha) \leq \frac{\| (y - \alpha)^m h(y) \| \, \sigma}{m \, |\alpha h(\alpha)|},$$

where

$$\sigma = \left(\begin{array}{c} \text{max degree} \\ \delta p(y, \alpha) \leq n \end{array} \right) \frac{|\delta p(\alpha, \alpha)|}{\left\| (y - \alpha)^{m-1} \, \delta p \, (y, \alpha) \right\|}.$$

The magnitude of the difference between the polynomials $g(y, \alpha)$ and $\bar{g}(y, \alpha)$ is

$$\| \delta g(y, \alpha) \| = \| g(y, \alpha) - \bar{g}(y, \alpha) \| = \| (y - \alpha)^m \| \, |h(\alpha)| \, ,$$

and thus the relative difference between $g(y, \alpha)$ and $\bar{g}(y, \alpha)$ is inversely proportional to $\rho(\alpha)$,

$$\frac{\|\delta g(y,\alpha)\|}{\|g(y,\alpha)\|} \leq \frac{\sigma}{m\,|\alpha|}\frac{\|(y-\alpha)^m\|}{\rho(\alpha)}. \tag{18}$$

It follows that if $\rho(\alpha)$ is large, there is a neighbouring polynomial that has a root of multiplicity at least $m+1$. □

Equation (18) shows that if $\rho(\alpha)$ is large, then $g(y,\alpha)$ is near another pejorative manifold, and if this polynomial is also ill-conditioned, then it is near a lower dimensional pejorative manifold of this manifold. This procedure of locating manifolds that are defined by higher order multiplicities is continued until the roots of the computed polynomial are sufficiently well-conditioned and close to the original polynomial. In this circumstance, the original polynomial may be considered to be a small perturbation of the computed polynomial, all of whose roots are well-conditioned. The computed polynomial is acceptable if it is sufficiently near the original polynomial, and it is reasonable to hypothesize that the original polynomial has a constraint that favours multiple roots.

5 Structured matrix methods

The polynomial root solver implemented by Method MR contains two operations, GCD computations and polynomial deconvolutions. Previous work has shown that the GCD computations can be implemented by structured matrix methods applied to the Sylvester resultant matrix of two polynomials [19–22]. This section considers, therefore, the application of structured matrix methods for the polynomial deconvolutions in Method MR. The simplest method of performing polynomial deconvolution is least squares, but it is unsatisfactory because it returns the coefficients of a polynomial that is an approximation of a rational function. By contrast, it will be seen that structured matrix methods guarantee that the solution vector contains the coefficients of an exact polynomial, and not the coefficients of a polynomial that is an approximation of a rational function.

A template for polynomial division that returns the quotient and remainder, and is simple and efficient, is described in [6], but it does not consider the effects of errors in the polynomial coefficients. The method for polynomial deconvolution, which is a restricted form of polynomial division when the remainder is identically equal to zero, that is described below enables the effect of perturbations in the polynomial coefficients to be considered. Also, it enables r coupled polynomial deconvolutions to be considered simultaneously, rather than as r individual polynomial deconvolutions.

The method MR contains two sets of polynomial deconvolutions, where the first set of deconvolutions, which is defined in (3), contains $r_{\max}$ deconvolu-

tions, and the second set of deconvolutions, which is defined in (4), contains $r_{\max} - 1$ deconvolutions. It is therefore appropriate to consider the r deconvolutions

$$h_i(y) = \frac{p_{i-1}(y)}{p_i(y)}, \qquad i = 1, \ldots, r, \tag{19}$$

where the polynomial $p_k(y)$ appears in the kth and $(k+1)$th deconvolutions. The degrees of the polynomials are [1]

$$\begin{aligned} \deg p_i(y) &= m_i, \quad i = 0, \ldots, r, \\ \deg h_i(y) &= n_i, \quad i = 1, \ldots, r, \end{aligned}$$

and the integers M, M_1 and N are defined in terms of these degrees as

$$M = \sum_{i=0}^{r-1}(m_i + 1), \qquad M_1 = \sum_{i=0}^{r}(m_i + 1) = M + (m_r + 1),$$

and

$$N = \sum_{i=1}^{r}(n_i + 1), \qquad n_i = m_{i-1} - m_i, \qquad i = 1, \ldots, r.$$

The polynomials $p_i(y), i = 0, \ldots, r$, must be processed before the deconvolutions are performed because computations on polynomials whose coefficients vary widely in magnitude are unreliable [7,10]. These preprocessing operations, which are now considered, minimise the ratio of the coefficient of maximum magnitude, to the coefficient of minimum magnitude, of the polynomials $p_i(y), i = 0, \ldots, r$.

Let the polynomials $p_i(y), i = 0, \ldots, r$, be given by

$$p_i(y) = \sum_{j=0}^{m_i} a_{i,j} y^{m_i - j}, \qquad i = 0, \ldots, r, \tag{20}$$

and consider the substitution

$$y = \theta w,$$

[1] These degrees should not be confused with the multiplicities of the roots of $\hat{f}(y)$.

where θ is a parameter to be determined and w is the new independent variable. The polynomials (20) become, therefore,

$$q_i(w) = \sum_{j=0}^{m_i} \left(a_{i,j} \theta^{m_i - j} \right) w^{m_i - j}, \qquad i = 0, \ldots, r, \tag{21}$$

and the coefficient of maximum magnitude in these $r+1$ polynomials is

$$\max \left\{ \max_{j=0,\ldots,m_0} \left| a_{0,j} \theta^{m_0 - j} \right|, \max_{j=0,\ldots,m_1} \left| a_{1,j} \theta^{m_1 - j} \right|, \ldots, \max_{j=0,\ldots,m_r} \left| a_{r,j} \theta^{m_r - j} \right| \right\}.$$

The expression for the coefficient of minimum magnitude is similar, but with max replaced by min, and thus θ_0, the optimal value of θ, is given by

$$\arg\min_{\theta} \left\{ \frac{\max \left\{ \max_{j=0,\ldots,m_0} \left| a_{0,j} \theta^{m_0 - j} \right|, \ldots, \max_{j=0,\ldots,m_r} \left| a_{r,j} \theta^{m_r - j} \right| \right\}}{\min \left\{ \min_{j=0,\ldots,m_0} \left| a_{0,j} \theta^{m_0 - j} \right|, \ldots, \min_{j=0,\ldots,m_r} \left| a_{r,j} \theta^{m_r - j} \right| \right\}} \right\}.$$

This minimisation problem can be written as:

minimise $\frac{t}{s}$

subject to

$$\begin{aligned}
t &\geq |a_{0,j}| \, \theta^{m_0 - j}, \quad j = 0, \ldots, m_0 \\
t &\geq |a_{1,j}| \, \theta^{m_1 - j}, \quad j = 0, \ldots, m_1 \\
&\vdots \\
t &\geq |a_{r,j}| \, \theta^{m_r - j}, \quad j = 0, \ldots, m_r \\
s &\leq |a_{0,j}| \, \theta^{m_0 - j}, \quad j = 0, \ldots, m_0 \\
s &\leq |a_{1,j}| \, \theta^{m_1 - j}, \quad j = 0, \ldots, m_1 \\
&\vdots \\
s &\leq |a_{r,j}| \, \theta^{m_r - j}, \quad j = 0, \ldots, m_r \\
s &> 0 \\
\theta &> 0.
\end{aligned}$$

The transformations

$$T = \log t, \qquad S = \log s, \qquad \phi = \log \theta,$$

and

$$\alpha_{i,j} = \log |a_{i,j}|, \qquad i = 0, \ldots, r, \qquad j = 0, \ldots, m_r,$$

where $\log \equiv \log_{10}$, enable this constrained minimisation to be written as:

minimise $T - S$

subject to

$$\begin{aligned}
T \quad & -(m_0 - j)\phi \geq \quad \alpha_{0,j}, \quad j = 0, \ldots, m_0 \\
T \quad & -(m_1 - j)\phi \geq \quad \alpha_{1,j}, \quad j = 0, \ldots, m_1 \\
& \quad \vdots \\
T \quad & -(m_r - j)\phi \geq \quad \alpha_{r,j}, \quad j = 0, \ldots, m_r \\
-S \; & +(m_0 - j)\phi \geq -\alpha_{0,j}, \quad j = 0, \ldots, m_0 \\
-S \; & +(m_1 - j)\phi \geq -\alpha_{1,j}, \quad j = 0, \ldots, m_1 \\
& \quad \vdots \\
-S \; & +(m_r - j)\phi \geq -\alpha_{r,j}, \quad j = 0, \ldots, m_r.
\end{aligned}$$

This problem can be written in matrix form as:

$$\text{minimise} \quad \begin{bmatrix} 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} T \\ S \\ \phi \end{bmatrix}$$

subject to

$$\begin{bmatrix} A_0 \\ A_1 \\ \vdots \\ A_r \\ B_0 \\ B_1 \\ \vdots \\ B_r \end{bmatrix} \begin{bmatrix} T \\ S \\ \phi \end{bmatrix} \geq \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_r \\ -\lambda_0 \\ -\lambda_1 \\ \vdots \\ -\lambda_r \end{bmatrix}, \tag{22}$$

where

$$A_i = \begin{bmatrix} 1 & 0 & -m_i \\ 1 & 0 & -(m_i - 1) \\ & \vdots & \\ 1 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix} \in \mathbb{R}^{(m_i+1)\times 3}, \qquad \lambda_i = \begin{bmatrix} \alpha_{i,0} \\ \alpha_{i,1} \\ \vdots \\ \alpha_{i,m_{i-1}} \\ \alpha_{i,m_i} \end{bmatrix} \in \mathbb{R}^{m_i+1},$$

and

$$B_i = \begin{bmatrix} 0 & -1 & m_i \\ 0 & -1 & (m_i - 1) \\ & \vdots & \\ 0 & -1 & 1 \\ 0 & -1 & 0 \end{bmatrix} \in \mathbb{R}^{(m_i+1)\times 3},$$

for $i = 0, \ldots, r$. The coefficient matrix and vector on the right hand side of (22) are of orders $2M_1 \times 3$ and $2M_1$ respectively. The constrained minimisation (22) is a standard problem in linear programming, and it allows the optimal value θ_0 of θ to be calculated.

It follows that the polynomials (21) are given by

$$q_i(w) = \sum_{j=0}^{m_i} \left(a_{i,j} \theta_0^{m_i - j} \right) w^{m_i - j}, \qquad i = 0, \ldots, r,$$

and thus if $\mathbf{q}_i$ denotes the vector of the coefficients of $q_i(w)$,

$$\mathbf{q}_i = \begin{bmatrix} a_{i,0}\theta_0^{m_i} & a_{i,1}\theta_0^{m_i-1} & \cdots & a_{i,m_i} \end{bmatrix}^T \in \mathbb{R}^{m_i+1}, \qquad i = 0, \ldots, r,$$

the r deconvolutions (19) can be written as r decoupled equations,

$$C_i(\mathbf{q}_i)\mathbf{h}_i = \mathbf{q}_{i-1}, \qquad i = 1, \ldots, r, \tag{23}$$

where each matrix $C_i(\mathbf{q}_i) \in \mathbb{R}^{(m_{i-1}+1)\times(n_i+1)}$ is Tœplitz, and the vector $\mathbf{h}_i \in \mathbb{R}^{n_i+1}$ contains the coefficients of the polynomial $h_i(y)$ after the substitution $y = \theta_0 w$ is made. The coefficient matrix $C(\mathbf{q}_1, \ldots, \mathbf{q}_r)$ of the r equations in (23) is

$$C(\mathbf{q}_1, \ldots, \mathbf{q}_r) = \operatorname{diag}\begin{bmatrix} C_1(\mathbf{q}_1) & \cdots & C_{r-1}(\mathbf{q}_{r-1}) & C_r(\mathbf{q}_r) \end{bmatrix} \in \mathbb{R}^{M\times N}.$$

It is noted that the polynomials in (19) are expressed in the independent variable y, and the polynomials in (23) are expressed in the independent variable w.

It is assumed that the coefficients of the polynomials are inexact, and thus (23) does not possess a solution. It is therefore necessary to add a structured matrix to $C(\mathbf{q}_1, \ldots, \mathbf{q}_r)$, and a structured vector to the right hand side, such that the coefficient matrix on the left hand side of the r equations (23) retains its diagonal block Tœplitz form, and the modified form of this equation therefore represents r polynomial deconvolutions. In particular, let $\mathbf{z}_i \in \mathbb{R}^{m_i+1}$ be the vector of perturbations added to the vector $\mathbf{q}_i$ of the coefficients of the polynomial $q_i(w)$, $i = 0, \ldots, r$, and let

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_0 & \mathbf{z}_1 & \cdots & \mathbf{z}_r \end{bmatrix}^T \in \mathbb{R}^{M_1},$$

where

$$\begin{aligned}
\mathbf{z}_0 &= \begin{bmatrix} z_0 & \cdots & z_{m_0} \end{bmatrix}^T \in \mathbb{R}^{m_0+1}, \\
\mathbf{z}_1 &= \begin{bmatrix} z_{m_0+1} & \cdots & z_{m_0+m_1+1} \end{bmatrix}^T \in \mathbb{R}^{m_1+1}, \\
\mathbf{z}_2 &= \begin{bmatrix} z_{m_0+m_1+2} & \cdots & z_{m_0+m_1+m_2+2} \end{bmatrix}^T \in \mathbb{R}^{m_2+1}, \\
&\vdots \\
\mathbf{z}_r &= \begin{bmatrix} z_M & \cdots & z_{M_1-1} \end{bmatrix}^T \in \mathbb{R}^{m_r+1}.
\end{aligned}$$

A matrix of structured perturbations is added to each matrix $C_i(\mathbf{q}_i), i = 1, \ldots, r$, and thus the coefficient matrix in (23) is replaced by

$$\begin{aligned} B(\mathbf{z}_1, \ldots, \mathbf{z}_r) &= C(\mathbf{q}_1, \ldots, \mathbf{q}_r) + E(\mathbf{z}_1, \ldots, \mathbf{z}_r) \\ &= \text{diag}\begin{bmatrix} C_1(\mathbf{q}_1) & C_2(\mathbf{q}_2) & \cdots & C_{r-1}(\mathbf{q}_{r-1}) & C_r(\mathbf{q}_r) \end{bmatrix} + \\ &\quad \text{diag}\begin{bmatrix} E_1(\mathbf{z}_1) & E_2(\mathbf{z}_2) & \cdots & E_{r-1}(\mathbf{z}_{r-1}) & E_r(\mathbf{z}_r) \end{bmatrix}, \end{aligned}$$

where $B(\mathbf{z}_1, \ldots, \mathbf{z}_r) \in \mathbb{R}^{M \times N}$ and $E_i(\mathbf{z}_i) \in \mathbb{R}^{(m_{i-1}+1)\times(n_i+1)}, i = 1, \ldots, r$, are Tœplitz matrices.

Consider now the vector on the right hand side of (23), the perturbed form of which is

$$\begin{bmatrix} \mathbf{q}_0 + \mathbf{z}_0 \\ \mathbf{q}_1 + \mathbf{z}_1 \\ \vdots \\ \mathbf{q}_{r-2} + \mathbf{z}_{r-2} \\ \mathbf{q}_{r-1} + \mathbf{z}_{r-1} \end{bmatrix} = \begin{bmatrix} \mathbf{q}_0 \\ \mathbf{q}_1 \\ \vdots \\ \mathbf{q}_{r-2} \\ \mathbf{q}_{r-1} \end{bmatrix} + \begin{bmatrix} I_M \vdots 0 \end{bmatrix} \begin{bmatrix} \mathbf{z}_0 \\ \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_{r-1} \\ \cdots \\ \mathbf{z}_r \end{bmatrix} = \begin{bmatrix} \mathbf{q}_0 \\ \mathbf{q}_1 \\ \vdots \\ \mathbf{q}_{r-2} \\ \mathbf{q}_{r-1} \end{bmatrix} + P\mathbf{z},$$

where

$$P = \begin{bmatrix} I_M \vdots 0 \end{bmatrix} \in \mathbb{R}^{M \times M_1}.$$

Equation (23) shows that the ith and $(i+1)$th deconvolutions, $i = 1, \ldots, r$, are coupled, and the matrix P guarantees that this coupling is preserved in the vectors $\mathbf{z}_i, i = 1, \ldots, r$. It also follows that the corrected form of (23) is

$$\Big(C(\mathbf{q}_1, \ldots, \mathbf{q}_r) + E(\mathbf{z}_1, \ldots, \mathbf{z}_r)\Big)\mathbf{h} = \mathbf{q} + P\mathbf{z}, \tag{24}$$

where $\mathbf{h} \in \mathbb{R}^N$ and $\mathbf{q} \in \mathbb{R}^M$ are given by, respectively,

$$\mathbf{h} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \cdots & \mathbf{h}_{r-1} & \mathbf{h}_r \end{bmatrix}^T \qquad \text{and} \qquad \mathbf{q} = \begin{bmatrix} \mathbf{q}_0 & \mathbf{q}_1 & \cdots & \mathbf{q}_{r-2} & \mathbf{q}_{r-1} \end{bmatrix}^T.$$

The residual due to an approximate solution of (24) is

$$\tilde{r} = \tilde{r}(\mathbf{z}) = (\mathbf{q} + P\mathbf{z}) - \Big(C(\mathbf{q}_1, \ldots, \mathbf{q}_r) + E(\mathbf{z}_1, \ldots, \mathbf{z}_r)\Big)\mathbf{h}, \tag{25}$$

and thus a first order Taylor expansion of $\tilde{r}(\mathbf{z})$ yields

$$\begin{aligned}
\tilde{r}(\mathbf{z} + \delta\mathbf{z}) = &\Big(\mathbf{q} + P\left(\mathbf{z} + \delta\mathbf{z}\right)\Big) \\
&-\Big(C(\mathbf{q}_1, \ldots, \mathbf{q}_r) + E(\mathbf{z}_1 + \delta\mathbf{z}_1, \ldots, \mathbf{z}_r + \delta\mathbf{z}_r)\Big)\left(\mathbf{h} + \delta\mathbf{h}\right) \\
= &\tilde{r}(\mathbf{z}) + P\delta\mathbf{z} - \Big(C(\mathbf{q}_1, \ldots, \mathbf{q}_r) + E(\mathbf{z}_1, \ldots, \mathbf{z}_r)\Big)\delta\mathbf{h} \\
&-\delta E(\mathbf{z}_1, \ldots, \mathbf{z}_r)\mathbf{h},
\end{aligned} \tag{26}$$

where

$$\delta E(\mathbf{z}_1, \ldots, \mathbf{z}_r) = \operatorname{diag}\Big[\, \delta E_1(\mathbf{z}_1) \;\; \cdots \;\; \delta E_{r-1}(\mathbf{z}_{r-1}) \;\; \delta E_r(\mathbf{z}_r) \,\Big].$$

There exist matrices $Y_i(\mathbf{h}_i) \in \mathbb{R}^{(m_{i-1}+1)\times(m_i+1)}$, $i = 1, \ldots, r$, such that

$$E_i(\mathbf{z}_i)\mathbf{h}_i = Y_i(\mathbf{h}_i)\mathbf{z}_i, \qquad i = 1, \ldots, r,$$

and thus

$$\delta E_i(\mathbf{z}_i)\mathbf{h}_i = Y_i(\mathbf{h}_i)\delta\mathbf{z}_i, \qquad i = 1, \ldots, r,$$

from which it follows that

$$\begin{aligned}
\delta E(\mathbf{z}_1, \ldots, \mathbf{z}_r)\mathbf{h} &= \operatorname{diag}\Big[\, Y_1(\mathbf{h}_1) \;\; Y_2(\mathbf{h}_2) \;\; \cdots \;\; Y_{r-1}(\mathbf{h}_{r-1}) \;\; Y_r(\mathbf{h}_r) \,\Big] \begin{bmatrix} \delta\mathbf{z}_1 \\ \delta\mathbf{z}_2 \\ \vdots \\ \delta\mathbf{z}_{r-1} \\ \delta\mathbf{z}_r \end{bmatrix} \\
&= \begin{bmatrix} 0 & Y_1(\mathbf{h}_1) & & & & \\ 0 & & Y_2(\mathbf{h}_2) & & & \\ \vdots & & & \ddots & & \\ 0 & & & & Y_{r-1}(\mathbf{h}_{r-1}) & \\ 0 & & & & & Y_r(\mathbf{h}_r) \end{bmatrix} \delta\mathbf{z}, \\
&= Y(\mathbf{h}_1, \ldots, \mathbf{h}_r)\delta\mathbf{z},
\end{aligned} \tag{27}$$

where $Y = Y(\mathbf{h}_1, \ldots, \mathbf{h}_r) \in \mathbb{R}^{M \times M_1}$ is given by

$$Y(\mathbf{h}_1, \ldots, \mathbf{h}_r) = \begin{bmatrix} 0 & Y_1(\mathbf{h}_1) & & & \\ 0 & & Y_2(\mathbf{h}_2) & & \\ \vdots & & & \ddots & \\ 0 & & & & Y_{r-1}(\mathbf{h}_{r-1}) \\ 0 & & & & & Y_r(\mathbf{h}_r) \end{bmatrix}.$$

The substitution of (27) into (26) yields the non-linear equation

$$\tilde{r}(\mathbf{z} + \delta\mathbf{z}) = \tilde{r}(\mathbf{z}) - (C + E)\delta\mathbf{h} - (Y - P)\delta\mathbf{z},$$

and thus the Newton-Raphson method requires the iterative solution of

$$\begin{bmatrix} (C+E) & (Y-P) \end{bmatrix} \begin{bmatrix} \delta\mathbf{h} \\ \delta\mathbf{z} \end{bmatrix} = \tilde{r},$$

which is an under-determined equation, where $\tilde{r} = \tilde{r}(\mathbf{z})$ and

$$\begin{bmatrix} (C+E) & (Y-P) \end{bmatrix} \in \mathbb{R}^{M \times (N+M_1)}.$$

If $\mathbf{h}^{(0)}$ and $\mathbf{z}^{(0)} = 0$ are the initial values of $\mathbf{h}$ and $\mathbf{z}$, respectively, in the Newton-Raphson method, then the $(j+1)$th iteration requires the minimisation of

$$\begin{aligned} \left\| \begin{matrix} \mathbf{h}^{(j+1)} - \mathbf{h}^{(0)} \\ \mathbf{z}^{(j+1)} \end{matrix} \right\|^2 &= \left\| \begin{matrix} \mathbf{h}^{(j)} + \delta\mathbf{h}^{(j)} - \mathbf{h}^{(0)} \\ \mathbf{z}^{(j)} + \delta\mathbf{z}^{(j)} \end{matrix} \right\|^2 \\ &= \left\| \begin{bmatrix} \delta\mathbf{h}^{(j)} \\ \delta\mathbf{z}^{(j)} \end{bmatrix} - \begin{bmatrix} -\left(\mathbf{h}^{(j)} - \mathbf{h}^{(0)}\right) \\ -\mathbf{z}^{(j)} \end{bmatrix} \right\|^2, \end{aligned}$$

subject to

$$\begin{bmatrix} (C+E) & (Y-P) \end{bmatrix}^{(j)} \begin{bmatrix} \delta\mathbf{h}^{(j)} \\ \delta\mathbf{z}^{(j)} \end{bmatrix} = \tilde{r}^{(j)},$$

where the initial value of $\mathbf{h}$ is calculated from (23),

$$\mathbf{h}^{(0)} = \begin{bmatrix} C_1(\mathbf{q}_1) & & & & \\ & C_2(\mathbf{q}_2) & & & \\ & & \ddots & & \\ & & & C_{r-1}(\mathbf{q}_{r-1}) & \\ & & & & C_r(\mathbf{q}_r) \end{bmatrix}^{\dagger} \begin{bmatrix} \mathbf{q}_0 \\ \mathbf{q}_1 \\ \vdots \\ \mathbf{q}_{r-2} \\ \mathbf{q}_{r-1} \end{bmatrix}, \tag{28}$$

and the superscript † denotes pseudo-inverse. This is a least squares minimisation with an equality constraint (the LSE problem),

$$\min_y \|Fy - s\| \qquad \text{subject to} \qquad Gy = t, \tag{29}$$

where

$$F = I_{N+M_1}, \qquad G = \begin{bmatrix} (C+E) & (Y-P) \end{bmatrix}^{(j)} \in \mathbb{R}^{M\times(N+M_1)},$$

$$y = \begin{bmatrix} \delta\mathbf{h}^{(j)} \\ \delta\mathbf{z}^{(j)} \end{bmatrix} \in \mathbb{R}^{N+M_1}, \qquad s = \begin{bmatrix} -\left(\mathbf{h}^{(j)} - \mathbf{h}^{(0)}\right) \\ -\mathbf{z}^{(j)} \end{bmatrix} \in \mathbb{R}^{N+M_1},$$

and $t = \tilde{r}^{(j)} \in \mathbb{R}^M$.

Algorithm 5.1 shows that the QR decomposition can be used to solve the LSE problem (29).

Algorithm 5.1: Deconvolution using the QR decomposition

Input The $r+1$ polynomials $p_i(y), i = 0, \ldots, r$.

Output The r polynomials $h_i(y), i = 1, \ldots, r$.

Begin

(1) Solve the linear programming problem (22) and compute the vectors $\mathbf{q}_i$ that store the coefficients of the polynomials $q_i(w), i = 0, \ldots, r$.
(2) Set $\mathbf{z}^{(0)} = 0$ and calculate $\mathbf{h}^{(0)}$ from (28).

(3) **Repeat**

(a) Compute the QR decomposition of G^T,

$$G^T = QR = Q \begin{bmatrix} R_1 \\ 0 \end{bmatrix}.$$

(b) Set $w_1 = R_1^{-T} t$.
(c) Partition FQ as

$$FQ = \begin{bmatrix} F_1 & F_2 \end{bmatrix},$$

where $F_1 \in \mathbb{R}^{(N+M_1)\times M}$ and $F_2 \in \mathbb{R}^{(N+M_1)\times(N+M_1-M)}$.
(d) Compute

$$w_2 = F_2^{\dagger} \left(s - F_1 w_1 \right).$$

(e) Compute the solution

$$y = Q \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}.$$

(f) Set $\mathbf{h} := \mathbf{h} + \delta\mathbf{h}$ and $\mathbf{z} := \mathbf{z} + \delta\mathbf{z}$.
(g) Update G, s and t, and compute the residual $\tilde{r}(\mathbf{z})$ from (25).

Until $\frac{\|\tilde{r}(\mathbf{z})\|}{\|\mathbf{q}+P\mathbf{z}\|} \leq 10^{-12}$

(4) Transform the polynomials whose coefficients are stored in $\mathbf{h}$ from the variable w to the variable y by the substitution $w = y/\theta_0$.

End

---

6 Examples

This section contains three examples in which the polynomial root solver described in this paper is compared with MULTROOT, which is another polynomial root solver explicitly designed for the computation of multiple roots of a polynomial [25,26].

Random noise is added to the exact coefficients of the polynomials specified

in the examples, but the form of the noise differs between Examples 6.1 and 6.2, and Example 6.3:

- The componentwise signal-to-noise ratio is a random variable, and not constant, in Examples 6.1 and 6.2. In particular, it follows from (5) and (7) that since ε_i^{-1} is the upper bound of the signal-to-noise ratio of the coefficient $\hat{a}_i$ of the exact polynomial $\hat{f}(y)$, the perturbations δa_i are given by

$$\delta a_i = \hat{a}_i r_i \varepsilon_i, \qquad i = 0, \ldots, m, \tag{30}$$

  where r_i and ε_i are uniformly distributed random variables in the intervals $[-1, 1]$ and $[10^{-9}, 10^{-7}]$, respectively. The random nature of ε_i makes the specification of a threshold for the small singular values of a matrix, and therefore the determination of its numerical rank, difficult.

  The roots of the polynomials in Example 6.1 are of the same order of magnitude, and similarly, the roots of the polynomials in Example 6.2 are of the same order of magnitude. These examples consider, therefore, the effect of a variation of two orders of magnitude in the values of ε_i.
- In Example 6.3, the upper bound of the componentwise signal-to-noise ratio ε^{-1} is constant and equal to 10^8, and the roots vary widely in magnitude. In particular, if α_i is a root of $\hat{f}(y)$, then

$$\frac{\max_i |\alpha_i|}{\min_i |\alpha_i|} \approx 10^5,$$

  in this example.

The polynomial root solver MultRoot is called with the function

```
multroot(poly,threshold)
```

where `poly` is the vector of the coefficients of the polynomial whose roots are to be computed, and `threshold` is the value of the threshold, below which the singular values of a matrix are assumed to be zero. If this argument is omitted, it is assigned a default value of 10^{-10}. There are two other input arguments, a threshold for the singular values that are defined to be zero and a parameter that controls the growth of the residuals in the GCD computations, but these two arguments are optional.

Bini and Fiorentino developed the multiprecision polynomial root solver MPSolve [2]. The algorithm that is implemented in this root solver is based on a sequence of nested sets that contain the roots of the polynomial. The algorithm is particularly suitable for sparse polynomials or polynomials that are generated by straight line programs. [2] The polynomial $p(y)$ whose roots are

[2] A straight line program is a sequence of arithmetic assignments to new variables, where the operands are constants, previously assigned variables, indeterminates or

to be computed is assumed to be exact, and provision is not made for errors in the coefficients of $p(y)$. The polynomial root solver described in this paper is explicitly designed to compute multiple roots in the presence of noise, and thus a comparison of the results obtained by MPSolve and the polynomial root solver described in this paper is not valid.

The three examples in this section compare MultRoot and the polynomial root solver described in this paper, and noise is added to the coefficients of the polynomials, as described above. The examples show that differences exist between these two root solvers in this circumstance. It is noted, however, that they return identical results in the absence of added noise.

Example 6.1 Noise with componentwise signal-to-noise ratio ε_i^{-1}, where ε_i is a uniformly distributed random variable in the interval $[10^{-9}, 10^{-7}]$, was added to the coefficients of the polynomial

$$\hat{f}(y) = (y + 9.7177)^2(y + 5.7885)^2(y + 4.5993)^3(y + 6.8623)^4 \times (y - 1.9438)^4(y - 5.6878)^5,$$

as shown in (30), thereby yielding the polynomial $f(y)$. The computed roots of $f(y)$ are shown in Table 1, where the first two columns show the exact roots and their multiplicities, and the last three columns show the computed roots, the computed multiplicities and the relative errors in the computed roots. It is seen that the multiplicities of the roots of $\hat{f}(y)$ are preserved in the roots of $f(y)$, and that the largest relative errors occur for the roots $\{-4.5993, -5.7885, -6.8623, -9.7177\}$, which is expected because these roots are closely spaced. Likewise, the smallest relative errors occur for the roots $\{1.9438, 5.6878\}$ because these roots are well separated from the other roots.

exact root	exact mult.	computed root	computed mult.	relative error
-9.7177	2	-9.71864125	2	9.68588764e−05
-5.7885	2	-5.78993616	2	2.48105234e−04
-4.5993	3	-4.59916523	3	2.93012847e−05
-6.8623	4	-6.86127489	4	1.49383132e−04
1.9438	4	1.94380036	4	1.84824181e−07
5.6878	5	5.68779415	5	1.02806795e−06

Table 1: The results of Example 6.1.

constants, and the operator is addition, multiplication, subtraction or division.

The roots of $f(y)$ were then computed using MULTROOT for the values

$$\texttt{threshold} = \left\{10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}\right\}, \tag{31}$$

and the computed roots were almost independent of the value of `threshold`. The values of `threshold` stated in (31) were selected because they span the range of values of ε_i. A typical set of computed roots is shown in Figure 2, and it is seen that simple roots were returned, that is, the multiple nature of the roots of $\hat{f}(y)$ is not preserved in the roots of $f(y)$. □

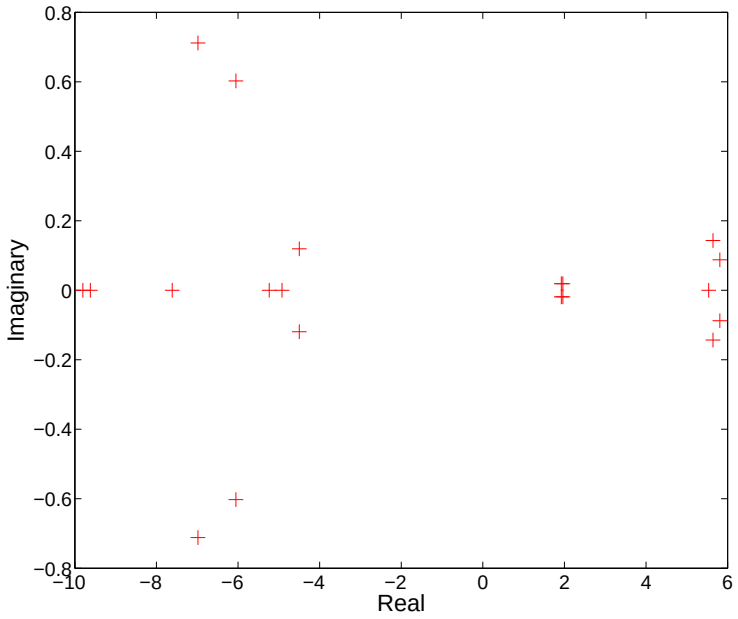


Fig. 2. The roots of $f(y)$ computed by MULTROOT for Example 6.1.

Example 6.2 The procedure described in Example 6.1 was repeated for the polynomial,

$$\hat{f}(y) = (y + 5.8308)^3(y + 4.5941)^5(y - 7.060)^6(y - 7.4785)^7,$$

and the results from the polynomial root solver considered in this paper are shown in Table 2, where the columns show the exact roots, their multiplicities, the computed roots, their multiplicities, and the relative errors in the computed roots. It is clear that the multiple nature of the exact roots of $\hat{f}(y)$ has been retained in the computed roots of $f(y)$. The results in the table are similar to the results in Table 1 because the relative errors of the closely spaced roots, $\{7.0600, 7.4785\}$, are $O(10^{-6})$, but the relative errors of the well separated roots, $\{-4.5941, -5.8308\}$, are smaller. It is also noted that the multiplicities of the closely spaced roots are larger than the multiplicities of

the well separated roots, and this difference also contributes to the differences in the relative errors of the computed roots.

exact root	exact mult.	computed root	computed mult.	relative error
-5.8308	3	-5.83079995	3	9.10861928e−09
-4.5941	5	-4.59410012	5	2.70268815e−08
7.0600	6	7.06000635	6	9.00016803e−07
7.4785	7	7.47849411	7	7.87925358e−07

Table 2: The results of Example 6.2.

The roots of $f(y)$ were then computed by MULTROOT, using the values of the parameter `threshold` in (31). In all cases, MULTROOT returned simple roots, and the computed roots showed very little variation with the value of `threshold`. A typical set of computed roots is shown in Figure 3, and it is seen that cluster analysis would suggest there are three distinct roots because the roots $\{7.0600, 7.4785\}$ have merged. □

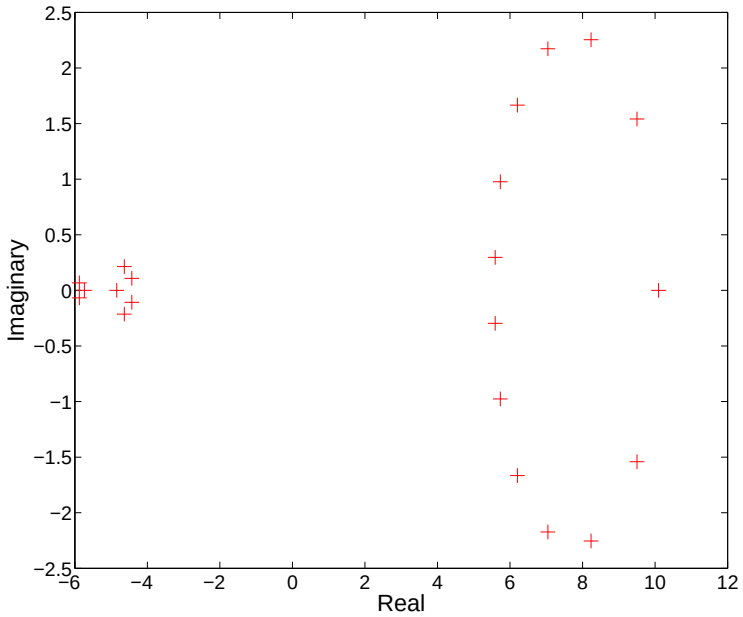


Fig. 3. The roots of $f(y)$ computed by MULTROOT for Example 6.2.

Example 6.3 Noise with componentwise signal-to-noise ratio $\varepsilon^{-1} = 10^8$ was added to the coefficients of the polynomial

$$\hat{f}(y) = (y - 2.6911 \times 10^{-3})^3 (y + 77.785)^4 (y - 2.1469 \times 10^{-2})^3 \times$$
$$(y - 7.7952)^2 (y + 1.8629 \times 10^2)^2 (y + 3.7298 \times 10^{-2})^2,$$

thereby yielding the polynomial $f(y)$. The roots of $\hat{f}(y)$ span about five orders of magnitude, and their computed values are shown in Table 3. The multiple nature of the roots of $\hat{f}(y)$ is preserved in the computed roots, and the relative errors in the roots are approximately equal to, or less than, ε, which shows that the computed roots are acceptable.

The roots of $f(y)$ were computed by MULTROOT for the values of `threshold` defined in (31). The program returned simple, and therefore incorrect, roots, for `threshold` $< \varepsilon$, but the correct roots and multiplicities were returned for `threshold` $\geq \varepsilon$. This must be compared with the results shown in Table 3, which did not require the specification of a threshold. □

exact root	exact mult.	computed root	computed mult.	relative error
-1.8629e+02	2	-1.86290015e+02	2	8.20301293e−08
-3.7298e−02	2	-3.72980000e−02	2	3.34291232e−10
7.7952e+00	2	7.79519991e+00	2	1.10662241e−08
2.6911e−03	3	2.69110007e−03	3	2.60706889e−08
2.1469e−02	3	2.14689999e−02	3	3.58568239e−09
-7.7785e+01	4	-7.77849973e+01	4	3.41004828e−08

Table 3: The results of Example 6.3.

The results of Example 6.3 are typical of many other results, and they therefore illustrate an important difference between MULTROOT and the polynomial root solver described in this paper.

7 Summary

This paper has described the implementation of a polynomial root solver explicitly designed for the computation of multiple roots of a polynomial in the presence of noise. In this polynomial root solver, the multiplicities of the roots are computed initially, and they are then used as constraints for the computation of the values of the roots. This polynomial root solver contains GCD computations and polynomial deconvolutions, both of which are ill-posed operations, and it was shown that structured matrix methods allow computa-

tionally reliable solutions to these operations to be obtained. It was shown that the method used in the polynomial root solver has a geometric interpretation in terms of the pejorative manifolds of a polynomial that has multiple roots. Structured and unstructured condition numbers of a multiple root of a polynomial were considered, and it was shown that the differences between them explain the good results obtained from the polynomial root solver described in this paper, and the poor results obtained from polynomial root solvers that do not make provision for multiple roots.

The MATLAB software that was used to obtain the results in this paper can be obtained by contacting the author.

References

[1] G. Arfken. *Mathematical Methods for Physicists*. Academic Press, 1985.

[2] D. A. Bini and G. Fiorentino. Design, analysis and implementation of a multiprecision polynomial rootfnder. *Numerical Algorithms*, 23:127–173, 2000.

[3] F. C. Chang. Factoring a polynomial with multiple roots. *Int. J. Comp. Math. Sciences*, 2(4):173–176, 2008.

[4] F. C. Chang. Solving multiple-root polynomials. *IEEE Antennas and Propagation Magazine*, 51(6):151–155, 2009.

[5] F. C. Chang. Polynomial GCD derived through monic polynomial subtractions. *International Scholarly Research Network, ISRN Applied Mathematics*, Article ID 714102, 7 pages, 2011.

[6] F. C. Chang. Polynomial division template. *IOSR Journal of Engineering*, 2(10):13–16, 2012.

[7] D. K. Dunaway. *A Composite Algorithm for Finding Zeros of Real Polynomials*. PhD thesis, Southern Methodist University, Texas, 1972.

[8] L. Foster. Generalizations of Laguerre's method. *SIAM J. Numer. Anal.*, 18:1004–1018, 1981.

[9] C. F. Gerald and P. O. Wheatley. *Applied Numerical Analysis*. Addison-Wesley, USA, 1994.

[10] S. Ghaderpanah and S. Klasa. Polynomial scaling. *SIAM J. Numer. Anal.*, 27(1):117–135, 1990.

[11] E. Hansen, M. Patrick and J. Rusnack. Some modificiations of Laguerre's method. *BIT*, 17:409–417, 1977.

[12] W. G. Horner. A new method of solving numerical equations of all orders by continuous approximation. *Philos. Trans. Roy. Soc. London*, 109:308–335, 1819.

[13] D. G. Hough. *Explaining and Ameliorating the Ill Condition of Zeros of Polynomials*. PhD thesis, Department of Computer Science, University of California, Berkeley, USA, 1977.

[14] M. A. Jenkins and J. F. Traub. A three-stage variable-shift iteration for polynomial zeros and its relation to generalized Raleigh iteration. *Numerische Mathematik*, 14:252–263, 1970.

[15] W. Kahan. Conserving confluence curbs ill-condition. Technical report, Department of Computer Science, University of California, Berkeley, USA, 1972.

[16] K. Madsen. A root-finding algorithm based on Newton's method. *BIT*, 13:71–75, 1973.

[17] J. V. Uspensky. *Theory of Equations*. McGraw-Hill, New York, USA, 1948.

[18] J. R. Winkler. High order terms for condition estimation of univariate polynomials. *SIAM J. Sci. Stat. Comput.*, 28(4):1420–1436, 2006.

[19] J. R. Winkler and M. Hasan. A non-linear structure preserving matrix method for the low rank approximation of the Sylvester resultant matrix. *Journal of Computational and Applied Mathematics*, 234:3226–3242, 2010.

[20] J. R. Winkler and M. Hasan. An improved non-linear method for the computation of a structured low rank approximation of the Sylvester resultant matrix. *Journal of Computational and Applied Mathematics*, 237(1):253–268, 2013.

[21] J. R. Winkler, M. Hasan and X. Y. Lao. Two methods for the calculation of the degree of an approximate greatest common divsior of two inexact polynomials. *Calcolo*, 49:241-267, 2012.

[22] J. R. Winkler and X. Y. Lao. The calculation of the degree of an approximate greatest common divsior of two polynomials. *Journal of Computational and Applied Mathematics*, 235:1587–1603, 2011.

[23] J. R. Winkler, X. Y. Lao and M. Hasan. The computation of multiple roots of a polynomial. *Journal of Computational and Applied Mathematics*, 236:3478–3497, 2012.

[24] C-D. Yan and W-H. Chieng. Method for finding multiple roots of polynomials. *Computers and Mathematics with Applications*, 51:605–620, 2006.

[25] Z. Zeng. Algorithm 835: MultRoot - A Matlab package for computing polynomial roots and multiplicities. *ACM Trans. Mathematical Software*, 30(2):218–236, 2004.

[26] Z. Zeng. Computing multiple roots of inexact polynomials. *Mathematics of Computation*, 74:869–903, 2005.