SIAM J. SCI. COMPUT.
Vol. 42, No. 1, pp. A452–A476





# THE COMPUTATION OF MULTIPLE ROOTS OF A BERNSTEIN BASIS POLYNOMIAL*

MARTIN BOURNE†, JOAB WINKLER†, AND YI SU‡

**Abstract.** This paper describes the algorithms of Musser and Gauss for the computation of multiple roots of a theoretically exact Bernstein basis polynomial $\hat{f}(y)$ when the coefficients of its given form $f(y)$ are corrupted by noise. The exact roots of $f(y)$ can therefore be assumed to be simple, and thus the problem reduces to the calculation of multiple roots of a polynomial $\tilde{f}(y)$ that is near $f(y)$, such that the backward error is small. The algorithms require many greatest common divisor (GCD) computations and polynomial deconvolutions, both of which are implemented by a structure-preserving matrix method. The motivation of these algorithms arises from the unstructured and structured condition numbers of a multiple root of a polynomial. These condition numbers have an elegant interpretation in terms of the pejorative manifold of $\hat{f}(y)$, which allows the geometric significance of the GCD computations and polynomial deconvolutions to be considered. A variant of the Sylvester resultant matrix is used for the GCD computations because it yields better results than the standard form of this matrix, and the polynomial deconvolutions can be computed in several different ways, sequentially or simultaneously, and with the inclusion or omission of the preservation of the structure of the coefficient matrix. It is shown that Gauss' algorithm yields better results than Musser's algorithm, and the reason for these superior results is explained.

**Key words.** Bernstein basis polynomials, Sylvester resultant matrix, Sylvester subresultant matrices, greatest common divisor, multiple roots

**AMS subject classifications.** 65F99, 12Y05

**DOI.** 10.1137/18M1219904

## 1. Introduction.

A multiple root of a polynomial is ill-conditioned because a perturbation in its coefficients causes it to break up into simple roots, and standard methods for their computation yield unsatisfactory results because they may return simple roots. This has led to the development of new methods for the computation of multiple roots of a polynomial, many of which involve greatest common divisor (GCD) computations and polynomial deconvolutions [6, 16, 20, 23, 26, 28, 29, 30, 31]. Both these operations are ill-conditioned, and thus simple methods for these calculations yield bad results.

The Bernstein basis is used in geometric modeling for the representation of curves and surfaces, and an important calculation is the determination of their points of intersection, which reduces to the calculation of the roots of a polynomial. Multiple roots are of particular interest because they define tangential intersections (blends), which are required for the reduction of high stresses at sharp corners and for aesthetic appeal. It may be thought that a Bernstein basis polynomial can be transformed to its power basis form, which would allow a root solver for power basis polynomials to be used. This transformation is, however, ill-conditioned [11], and this paper therefore discusses a method for the computation of multiple roots of a Bernstein basis

*Submitted to the journal's Methods and Algorithms for Scientific Computing section October 9, 2018; accepted for publication (in revised form) December 2, 2019; published electronically February 19, 2020.

https://doi.org/10.1137/18M1219904

**Funding:** This work was supported by a studentship from The Agency for Science, Technology and Research (A*STAR), Singapore, and The University of Sheffield.

†Department of Computer Science, The University of Sheffield, Regent Court, 211 Portobello, Sheffield S1 4DP, United Kingdom (mbourne1@sheffield.ac.uk, j.r.winkler@sheffield.ac.uk).

‡Institute of High Performance Computing, A*STAR, 1 Fusionopolis Way, 16-16 Connexis North, Singapore 138632 (suyi@ihpc.a-star.edu.sg).

polynomial such that the power basis is not used and all computations are performed on the Bernstein basis. This basis conversion can be avoided by the computation of the eigenvalues of the companion matrix of a Bernstein basis polynomial [21], but problems arise because of the accuracy with which multiple eigenvalues can be computed by the QR decomposition. For example, the function `roots` in MATLAB returns $\{1.0002, 1.0000 \pm 0.0002i, 0.9998\}$ for the polynomial $f(y) = (y-1)^4$.

Several methods for the computation of the roots of a Bernstein basis polynomial and the algorithms of Musser [16] and Gauss [20] are discussed in section 2. The numerical factorization of multivariate polynomials is considered by Wu and Zeng [28], and the suite of MATLAB programs NAClab which was developed by Zeng [31] for numerical algebraic computations is considered in [32]. The algorithms of Musser and Gauss use GCD computations and polynomial deconvolutions, and they are motivated by the unstructured and structured condition numbers of a multiple root of a polynomial. These condition numbers are considered in section 3, and their difference explains the suitability of the algorithms of Musser and Gauss for the computation of multiple roots of a polynomial. This discussion on condition numbers leads to section 4, which considers the pejorative manifold of a polynomial that has multiple roots, and it is shown that it provides an elegant geometric interpretation of the algorithms of Musser and Gauss.

The Sylvester resultant matrix is frequently used for the computation of the GCD of two polynomials, and it is considered in section 5. Many methods have been developed for this computation for two power basis polynomials [2, 8, 10, 13, 14, 19], but much less work has been done on this computation for two Bernstein basis polynomials [3, 4, 9, 27]. The data in practical problems is corrupted by noise and it is therefore necessary to consider an approximate greatest common divisor (AGCD) of two polynomials, and this is discussed in section 6. Polynomial deconvolution is considered in section 7, and it is shown it can be performed in several ways, which differ in the manner in which the deconvolutions are performed (sequentially or simultaneously) and the preservation, or otherwise, of the structure of the coefficient matrix in the linear algebraic equation. The complexity of the algorithms of Musser and Gauss is considered in section 8, and section 9 contains examples of these algorithms for the computation of multiple roots of a Bernstein basis polynomial. Section 10 contains a summary of the paper.

There are several novel aspects of the paper:

1. The polynomial root solver in NAClab is for power basis polynomials, but this paper considers a polynomial root solver for Bernstein basis polynomials.
2. The polynomial root solver in NAClab and the polynomial root solver described in this paper use the Sylvester matrix and its subresultant matrices. These matrices have a well-defined structure, and this structure is preserved in the polynomial root solver described in this paper, but the method of non-linear least squares is used in NAClab for the refinement of the solution.
3. The combinatorial terms in the Bernstein basis functions introduce computational difficulties because they cause the entries of the Sylvester matrix and its subresultant matrices to span many orders of magnitude. These difficulties, which do not arise when power basis polynomials are considered, are mitigated by the use of a modified form of these matrices.

The application of NAClab to the computation of the roots of a Bernstein basis polynomial requires a parameter substitution, but it yields bad results. These bad results arise from numerical difficulties associated with the parameter substitution, and they are not caused by theoretical and computational issues with NAClab, which yields very good results for power basis polynomials.

**2. Methods for computing multiple roots of a Bernstein polynomial.** The properties of the Bernstein basis have been used for the computation of the roots of a polynomial $f(y)$ in this basis. For example, Lane and Riesenfeld [15] use bisection and the variation diminishing property of the Bernstein basis to isolate and approximate the real roots of $f(y)$ in a given interval, but it is shown in [5] that problems arise with double roots and higher order roots. Another method for computing the roots of $f(y)$ is Bézier clipping [1, 7, 18], but it is designed for first order tangent intersections, and its performance degrades for higher order tangencies. The need arises, therefore, for a method for the computation of high order multiple roots of a Bernstein basis polynomial.

**2.1. Methods that use GCD computations and polynomial deconvolutions.** The methods for the computation of multiple roots of a power basis polynomial that use GCD computations and polynomial deconvolutions can be implemented in the Bernstein basis, but significant complications exist such that their simple reproduction in the Bernstein basis is not possible. In particular, numerical difficulties arise from the combinatorial terms in the Bernstein basis functions which are, for a polynomial of degree $m$,

$$\phi_i(y) = \binom{m}{i}(1-y)^{m-i}y^i, \qquad i = 0, \ldots, m, \tag{1}$$

because the entries in the matrices that arise in the computations may span several orders of magnitude. Also, the Sylvester matrix and its subresultant matrices, which are used for the GCD computations, for two power basis polynomials are formed by the concatenation of two Tœplitz matrices, but these matrices for two Bernstein basis polynomials do not have this structure [3, 4, 24], from which it follows that algorithms that require the Sylvester matrix and its subresultant matrices are more complicated for Bernstein basis polynomials.

The algorithms of Musser and Gauss are most easily understood if the polynomial $f(y)$ is written as the product of factors of degrees one, two, three, etc.,

$$f(y) = \sum_{i=0}^{m} a_i \binom{m}{i}(1-y)^{m-i}y^i = s_1(y)s_2^2(y)s_3^3(y)\cdots s_r^r(y), \tag{2}$$

where $s_1(y)$ is the product of all the linear factors of $f(y)$, $s_2^2(y)$ is the product of all the quadratic factors of $f(y)$ and in general, $s_i^i(y)$ is the product of all the factors of degree $i$ of $f(y)$. If $f(y)$ does not contain a factor of degree $k$, then $s_k(y)$ is set equal to a constant, which can be assumed to be unity.

The algorithms of Musser and Gauss are shown in Algorithms 1 and 2, respectively, and it is seen that they differ because Gauss' algorithm requires several GCD computations of a polynomial and its derivative, but Musser's algorithm requires only one computation of this type. These algorithms contain two stages that make them suitable for the computation of multiple roots of a polynomial and distinguish them from methods that fail to yield satisfactory results for this computation:

**Stage 1.** Calculate the multiplicity of each distinct root.

**Stage 2.** Calculate the value of each distinct root.

It follows from (2) that the multiplicities of the distinct roots of $f(y)$ are defined by the indices $i$ in the **while** loop in Musser's algorithm (lines 7–13) and in the **for** loop in Gauss' algorithm (lines 13–16). The multiplicity $i$ of each distinct root is therefore specified before the values of the roots with this multiplicity are calculated, assuming

**Algorithm 1.** Musser.

$f_0(y) \leftarrow f(y)$
% Perform the first GCD computation and polynomial deconvolution
$f_1(y) \leftarrow \text{GCD}\left(f_0, f_0^{(1)}\right) = s_2(y)s_3^2(y)\cdots s_r^{r-1}(y)$
$h_1(y) \leftarrow \frac{f_0(y)}{f_1(y)} = s_1(y)s_2(y)\cdots s_r(y)$
$i \leftarrow 1$
% Perform the remaining GCD computations and polynomials deconvolutions
**while** $\deg h_i(y) > 0$ **do**
  $h_{i+1}(y) \leftarrow \text{GCD}\,(f_i, h_i) = s_{i+1}(y)s_{i+2}(y)\cdots s_r(y)$
  $f_{i+1}(y) \leftarrow \frac{f_i(y)}{h_{i+1}(y)} = s_{i+2}(y)s_{i+3}^2(y)\cdots s_r^{r-i-1}(y)$
  $w(y) \leftarrow \frac{h_i(y)}{h_{i+1}(y)} = s_i(y)$
  Solve $w(y) = 0$. If $\alpha$ is a root of $w(y)$, it is a root of multiplicity $i$ of $f(y)$.
  $i \leftarrow i + 1$
**end while**

**Algorithm 2.** Gauss.

$f_0(y) \leftarrow f(y)$
$i \leftarrow 0$
% Perform several GCD computations of a polynomial and its derivative
**while** $\deg f_i(y) > 0$ **do**
  $f_{i+1}(y) \leftarrow \text{GCD}\left(f_i, f_i^{(1)}\right) = s_{i+2}(y)s_{i+3}^2(y)\cdots s_r^{r-i-1}(y)$
  $i \leftarrow i + 1$
**end while**
$r \leftarrow i$
% Compute the polynomials $h_i(y), i = 1, \ldots, r$
**for** $i \leftarrow 1, r$ **do**
  $h_i(y) \leftarrow \frac{f_{i-1}(y)}{f_i(y)} = s_i(y)s_{i+1}(y)\cdots s_r(y)$
**end for**
**for** $i \leftarrow 1, r-1$ **do**
  $w(y) \leftarrow \frac{h_i(y)}{h_{i+1}(y)} = s_i(y)$
  Solve $w(y) = 0$. If $\alpha$ is a root of $w(y)$, it is a root of multiplicity $i$ of $f(y)$.
**end for**
$w_r(y) \leftarrow h_r(y)$
Solve $w_r(y) = 0$. If $\alpha$ is a root of $w_r(y)$, it is a root of multiplicity $r$ of $f(y)$.

the roots exist, that is, $w(y) = s_i(y)$ is not a polynomial of degree zero. It also follows from (2) that the roots of the polynomials $w(y)$ and $w_r(y)$ (in Gauss' algorithm) are simple, and these algorithms are therefore divide-and-conquer algorithms.

NAClab is a suite of MATLAB programs for computations on power basis polynomials. Although the transformation between the power and Bernstein bases is not recommended because it is ill-conditioned, a change of variable allows a power basis polynomial root solver to be implemented for Bernstein basis polynomials. In particular, the substitution

$$t = y/(1-y), \qquad y \neq 1, \tag{3}$$

in $f(y)$ yields the power basis polynomial $g(t)$ whose coefficients are $a_i\binom{m}{i}$,

$$g(t) = \frac{1}{(1+t)^m} \sum_{i=0}^{m} a_i \binom{m}{i} t^i, \qquad t \neq -1,$$

and thus if $t_0$ is a root of multiplicity $p$ of $g(t)$, then $y_0 = t_0/(1+t_0)$ is a root of multiplicity $p$ of $f(y)$. Although this substitution is simple, it follows from (3) that it suffers from numerical problems when $y_0 \approx 1$ because $t_0$ is then sensitive to a small change in $y_0$. Also, it follows from the presence of the combinatorial terms $\binom{m}{i}$ in the coefficients of $g(t)$ that the numerical problems discussed above with respect to the Bernstein basis may occur when the substitution (3) is used. This substitution allows NAClab, and more generally other root solvers for power basis polynomials, to be used for the computation of multiple roots of a Bernstein basis polynomial, but the results in section 9 show that it yields bad results. As noted in section 1, these bad results are due to numerical difficulties associated with the substitution (3), and not the theoretical basis of, and software implementation in, NAClab.

**3. Condition numbers of a multiple root of a polynomial.** Sections 1 and 2 considered traditional methods, and the algorithms of Musser and Gauss, respectively, for the calculation of multiple roots of $f(y)$. The traditional methods do not use the properties of multiple roots, but these properties are exploited in methods that use GCD computations and polynomial deconvolutions, which therefore yield significantly better results. These differences in the results are explained by considering unstructured and structured condition numbers of a multiple root of $f(y)$, which are considered in Theorems 3.1 [22] and 3.2, respectively. The continuity of the roots of a polynomial due to a perturbation in its coefficients is assumed, so that small perturbations in the coefficients yield small variations in the roots.

THEOREM 3.1. *Let the polynomial $g(y)$ have real coefficients $b_i, i = 0, \ldots, m$, with respect to the Bernstein basis, which is defined in* (1)*,*

$$g(y) = \sum_{i=0}^{m} b_i \phi_i(y),$$

*and let the coefficients $b_i$ be perturbed to $b_i + \delta b_i$, where*

$$|\delta b_i| \leq \varepsilon \, |b_i| \,, \qquad i = 0, \ldots, m.$$

*Let the real root $\alpha$ of $g(y)$ have multiplicity $r$, and let one of these $r$ roots be perturbed to $\alpha + \delta\alpha$ due to the perturbations in the coefficients. The unstructured componentwise condition number of $\alpha$ is*

$$\kappa(\alpha) = \max_{|\delta b_i| \leq \varepsilon |b_i|} \frac{|\delta\alpha|}{|\alpha|} \frac{1}{\varepsilon} = \frac{1}{\varepsilon^{1-\frac{1}{r}}} \frac{1}{|\alpha|} \left( \frac{r!}{\left| g^{(r)}(\alpha) \right|} \sum_{i=0}^{m} |b_i \phi_i(\alpha)| \right)^{\frac{1}{r}}. \tag{4}$$

The structured condition number requires that $g(y)$ be written in terms of its roots $\tilde{\alpha} = \{\alpha_i\}_{i=1}^{p}$, $\alpha_j \neq \alpha_k$, because this condition number is defined by the preservation of the multiplicity $m_k$ of each distinct root $\alpha_k$ in its perturbed form,

$$g(y, \tilde{\alpha}) = \prod_{i=1}^{p} \Big( \theta(y, \alpha_i) \Big)^{m_i} = \prod_{i=1}^{p} \Big( (1-\alpha_i)y - \alpha_i(1-y) \Big)^{m_i}, \qquad \sum_{i=1}^{p} m_i = m. \tag{5}$$

The expansion of the second product yields a polynomial expressed in terms of the basis functions $(1-y)^{m-i}y^i, i = 0, \ldots, m$, that is, the Bernstein basis functions (1) after the combinatorial terms are moved from the basis functions to the coefficients. The perturbed form of $g(y, \tilde{\alpha})$ due to a change $\delta\alpha_i$ in each root $\alpha_i$, $i = 1, \ldots, p$, is

$$g(y, \tilde{\alpha} + \delta\tilde{\alpha}) = \prod_{i=1}^{p} \Big(\theta(y, (\alpha_i + \delta\alpha_i))\Big)^{m_i} \approx g(y, \tilde{\alpha}) + \sum_{i=1}^{p} \frac{\partial g(y, \tilde{\alpha})}{\partial \alpha_i} \delta\alpha_i, \tag{6}$$

to first order, and (5) and (6) lead to Theorem 3.2.

THEOREM 3.2. *The structured condition number of a root $\alpha_k$ of multiplicity $m_k$ of $g(y, \tilde{\alpha})$ is*

$$\rho(\alpha_k) = \frac{\|\theta(y, \alpha_k)\|}{m_k \, |\alpha_k|}, \tag{7}$$

*where $\theta(y, \alpha_k)$ is defined in* (5)*,*

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

The unstructured condition number $\kappa(\alpha)$ is a function of the upper bound of the relative error $\varepsilon$ in the coefficients, but $\rho(\alpha_k)$ is independent of $\varepsilon$.

*Example* 3.3. Consider a Bernstein basis polynomial of degree $m$ that has one root $\alpha$ of multiplicity $m$,

$$g(y) = \Big( -\alpha(1-y) + (1-\alpha)y \Big)^m = \sum_{i=0}^{m} b_i \binom{m}{i} (1-y)^{m-i} y^i.$$

If $m$ is sufficiently large, the unstructured condition number (4) of the root $\alpha$ is independent of $m$ and inversely proportional to the upper bound of the relative error in the coefficients,

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

The structured condition number $\rho(\alpha)$ is easily obtained from (7),

$$\rho(\alpha) = \frac{\Big(\alpha^2 + (1-\alpha)^2\Big)^{\frac{1}{2}}}{m \, |\alpha|},$$

where $\|\cdot\| = \|\cdot\|_2$, and thus unlike $\kappa(\alpha)$, $\rho(\alpha)$ is independent of $\varepsilon$. Also, $\rho(\alpha)$ is a function of the multiplicity $m$ of $\alpha$, but the calculation of $m$ is not trivial because it reduces to the determination of the rank loss of a matrix [3, 25].

Example 3.3 is an example of the stability of a multiple root of a polynomial with respect to a perturbation that preserves its multiplicity. It follows that if the multiplicity of each distinct root $\alpha_k$ of $f(y)$, which is defined in (2), is calculated initially, and the method for the computation of the values of the roots satisfies the constraint that their multiplicities be preserved, then the condition number $\rho(\alpha_k)$ shows that the method is numerically stable. It was shown in section 2 that this procedure is adopted in the algorithms of Musser and Gauss.

**4. The pejorative manifold of a polynomial.** The pejorative manifold [12] of a polynomial provides a geometric interpretation of the conditions for which the perturbations of the coefficients of $f(y)$ (a) cause its multiple roots to break up into simple roots, which are therefore unstable, and (b) cause its multiple roots to retain their multiplicities, which are therefore stable. These conditions correspond to the unstructured and structured condition numbers, respectively.

Let $f(y)$ have $p$ distinct roots $\alpha_i, i = 1, \dots, p$, such that the multiplicity of $\alpha_i$ is $m_i$,

$$f(y) = \sum_{i=0}^{m} a_i \binom{m}{i} (1-y)^{m-i} y^i = \prod_{i=1}^{p} \Big( \alpha_i (1-y) - (1-\alpha_i) y \Big)^{m_i},$$

where

$$a_0 = \prod_{i=1}^{p} \alpha_i^{m_i}, \qquad a_m = \prod_{i=1}^{p} \Big( -(1-\alpha_i) \Big)^{m_i}, \qquad \text{and} \qquad \sum_{i=1}^{p} m_i = m.$$

The pejorative manifold of a polynomial is defined by the multiplicities of its roots, and it is therefore convenient to consider the monic form $g(y)$ of $f(y)$, which is obtained by normalizing the coefficients of $f(y)$ by $a_0$,

$$g(y) = \sum_{i=0}^{m} b_i \binom{m}{i} (1-y)^{m-i} y^i = \prod_{i=1}^{p} \Big( (1-y) - \lambda_i y \Big)^{m_i}, \qquad b_0 = 1,$$

where

$$b_i = \frac{a_i}{a_0}, \qquad b_m = \prod_{i=1}^{p} (-\lambda_i)^{m_i}, \qquad \lambda_i = \frac{1-\alpha_i}{\alpha_i}, \qquad \alpha_i \neq 0.$$

DEFINITION 4.1 (pejorative manifold). *Let $\mu = \{m_1, \dots, m_p\}$ be the set of multiplicities of the roots of the polynomial $g(y)$. The pejorative manifold $\mathcal{M}_{(\mu)} \subset \mathbb{R}^m$ is the set of real coefficients $\{b_1, \dots, b_m\}$ such that $g(y)$ has $p$ distinct roots whose multiplicities are equal to $\mu$.*

*Example* 4.2. Consider the cubic Bernstein basis polynomial $f(y)$ that has real roots $\alpha_1, \alpha_2$, and $\alpha_3$,

$$f(y) = \alpha_1\alpha_2\alpha_3 \binom{3}{0} (1-y)^3 - \frac{(\alpha_1\alpha_2\beta_3 + \alpha_2\alpha_3\beta_1 + \alpha_3\alpha_1\beta_2)}{\binom{3}{1}} \binom{3}{1} (1-y)^2 y$$
$$+ \frac{(\alpha_1\beta_2\beta_3 + \alpha_2\beta_3\beta_1 + \alpha_3\beta_1\beta_2)}{\binom{3}{2}} \binom{3}{2} (1-y) y^2 - \beta_1\beta_2\beta_3 \binom{3}{3} y^3,$$

where $\beta_i = 1 - \alpha_i, i = 1, 2, 3$. The configurations of the roots of $f(y)$ that include multiple roots are (a) one cubic root and (b) one double root and one simple root, and each configuration defines a pejorative manifold.

Consider initially the case in which $f(y)$ has a real triple root $\alpha := \alpha_1 = \alpha_2 = \alpha_3$,

$$f(y) = \Big( \alpha(1-y) - (1-\alpha) y \Big)^3 = \sum_{i=0}^{3} (-1)^i \alpha^{3-i} (1-\alpha)^i \binom{3}{i} (1-y)^{3-i} y^i,$$

whose monic form is

$$g(y) = \sum_{i=0}^{3} (-\lambda)^i \binom{3}{i} (1-y)^{3-i} y^i, \qquad \lambda = \frac{1-\alpha}{\alpha}, \qquad \alpha \neq 0.$$

The coefficient of $(1-y)^3$ is one, and if the other coefficients are $(X(\lambda), Y(\lambda), Z(\lambda))$, then the pejorative manifold $\mathcal{M}_{(3)}$ of a cubic Bernstein basis polynomial that has one real cubic root is the curve $\mathcal{C}$, which is parameterized by $\lambda$,

$$\mathcal{C} : \begin{pmatrix} X(\lambda) & Y(\lambda) & Z(\lambda) \end{pmatrix} = \begin{pmatrix} -\lambda & \lambda^2 & -\lambda^3 \end{pmatrix}.$$

Consider now the situation in which $f(y)$ has a real double root $\alpha_1$ and a real simple root $\alpha_2$,

$$\begin{aligned} f(y) = \alpha_1^2 \alpha_2 \binom{3}{0} (1-y)^3 &- \frac{\left(\alpha_1^2(1-\alpha_2) + 2\alpha_1\alpha_2(1-\alpha_1)\right)}{\binom{3}{1}} \binom{3}{1} (1-y)^2 y \\ &+ \frac{\left(2\alpha_1(1-\alpha_1)(1-\alpha_2) + \alpha_2(1-\alpha_1)^2\right)}{\binom{3}{2}} \binom{3}{2} (1-y) y^2 \\ &- (1-\alpha_1)^2(1-\alpha_2) \binom{3}{3} y^3. \end{aligned}$$

The monic form of $f(y)$ is considered by defining $\lambda$ and $\mu$ as

$$\lambda = \frac{1-\alpha_1}{\alpha_1} \qquad \text{and} \qquad \mu = \frac{1-\alpha_2}{\alpha_2}, \qquad \alpha_1, \alpha_2 \neq 0,$$

and thus the pejorative manifold $\mathcal{M}_{(2,1)}$ of a cubic Bernstein basis polynomial that has one real simple root and one real double root is obtained from the second, third, and fourth coefficients of the monic form of $f(y)$,

$$\mathcal{S} : \begin{pmatrix} X(\lambda,\mu) & Y(\lambda,\mu) & Z(\lambda,\mu) \end{pmatrix} = \begin{pmatrix} \frac{-(2\lambda+\mu)}{3} & \frac{\lambda(\lambda+2\mu)}{3} & -\lambda^2\mu \end{pmatrix}.$$

The curve $\mathcal{C}$ and surface $\mathcal{S}$ are shown in Figure 1, where $\mathcal{C}$ is defined by the condition $\lambda = \mu$. If $r$ is the position vector of an arbitrary point on $\mathcal{S}$, then

$$r = \begin{bmatrix} \frac{-(2\lambda+\mu)}{3} & \frac{\lambda(\lambda+2\mu)}{3} & -\lambda^2\mu \end{bmatrix},$$

and thus the normal vector to $\mathcal{S}$ is

$$\frac{\partial r}{\partial \lambda} \times \frac{\partial r}{\partial \mu} = -\begin{bmatrix} \frac{2\lambda^2(\lambda-\mu)}{3} & \frac{2\lambda(\lambda-\mu)}{3} & \frac{2(\lambda-\mu)}{9} \end{bmatrix}.$$

It follows that the normal vector to $\mathcal{S}$ is not defined when $\lambda = \mu$, and thus, as shown in Figure 1, the curve $\mathcal{C}$ is defined by the sharp edge in $\mathcal{S}$.

Every point in the space $(X(\lambda,\mu,), Y(\lambda,\mu), Z(\lambda,\mu))$ defines a cubic Bernstein basis polynomial with real roots, and points on $\mathcal{C}$ and $\mathcal{S}$ define polynomials $f(y)$ with the restricted root structures of a real cubic root, and a real simple root and a real double root, respectively. The geometric implications of the difference between the structured and unstructured condition numbers of a multiple root $\alpha$, $\rho(\alpha)$ and $\kappa(\alpha)$, respectively, follow from the manifolds $\mathcal{C}$ and $\mathcal{S}$ in Example 4.2. In particular, the small value of $\rho(\alpha)$ implies that a small displacement from a point $\mathcal{P}$ on $\mathcal{C}$ or $\mathcal{S}$ to

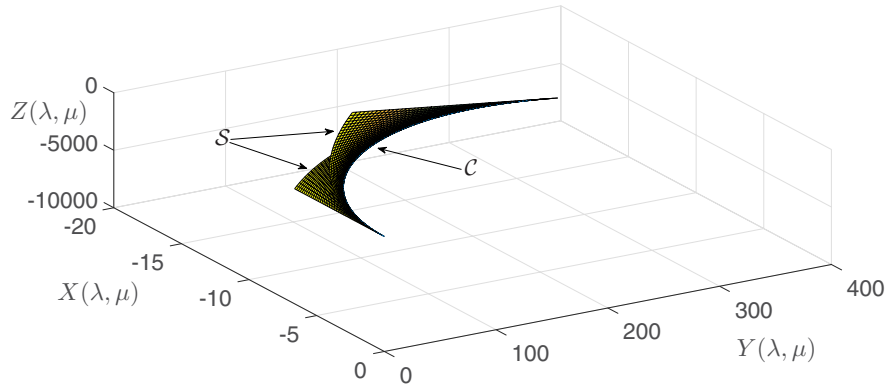


FIG. 1. *The curve $\mathcal{C}$ and surface $\mathcal{S}$, where $X(\lambda,\mu) = \frac{-(2\lambda+\mu)}{3}$, $Y(\lambda,\mu) = \frac{\lambda(\lambda+2\mu)}{3}$, $Z(\lambda,\mu) = -\lambda^2\mu$, for Example* 4.2.

a neighboring point on the same manifold causes a small change in the value of the root $\alpha$, and its multiplicity is unchanged. If, however, the displacement occurs from $\mathcal{P}$ to a point that does not lie on a manifold, then $\alpha$ breaks up into simple roots and the large value of $\kappa(\alpha)$ implies that the errors in the roots of $f(y)$ are large. This connection between condition numbers and the pejorative manifold of a polynomial will be extended in section 4.1, where it will be shown that the pejorative manifold of a polynomial provides an elegant geometric interpretation of the algorithms of Musser and Gauss for the computation of its multiple roots.

The intersection of two or more manifolds that have the same number of distinct roots is empty. For example, the roots of the polynomial with root structure $\pi = (2,2,2)$ (three double roots) cannot merge to form a polynomial with root structure $\pi = (3,2,1)$ (one cubic root, one double root, and one simple root). Manifolds of this type must therefore contain a separating surface, and Example 4.3 considers the equation of this surface for the manifolds $\mathcal{M}_{(4,1)}$ and $\mathcal{M}_{(3,2)}$.

*Example* 4.3. Let $g_1(y)$ and $g_2(y)$ be real monic Bernstein basis polynomials of degree five with root structures $\pi = (4,1)$ and $\pi = (3,2)$, respectively,

$$g_1(y) = \sum_{i=0}^{5} b_{1,i}\binom{5}{i}(1-y)^{5-i}y^i \quad \text{and} \quad g_2(y) = \sum_{i=0}^{5} b_{2,i}\binom{5}{i}(1-y)^{5-i}y^i,$$

where $b_{1,0} = b_{2,0} = 1$. If $b_1$ and $b_2$ are the vectors,

$$b_1 = \begin{bmatrix} b_{1,1} & b_{1,2} & b_{1,3} & b_{1,4} & b_{1,5} \end{bmatrix} \quad \text{and} \quad b_2 = \begin{bmatrix} b_{2,1} & b_{2,2} & b_{2,3} & b_{2,4} & b_{2,5} \end{bmatrix},$$

then $b_1 \in \mathcal{M}_{(4,1)}$ and $b_2 \in \mathcal{M}_{(3,2)}$, and the following statements hold:

1. The function
$$s(b_{1,1}, b_{1,2}, b_{1,3}, b_{1,4}) = 5b_{1,4} - 20b_{1,1}b_{1,3} + 30b_{1,2}b_{1,1}^2 - 15b_{1,1}^4$$
is negative for all points on $\mathcal{M}_{(4,1)}$ and zero on $\mathcal{M}_{(5)}$, the manifold defined by one root of multiplicity five.
2. The function
$$s(b_{2,1}, b_{2,2}, b_{2,3}, b_{2,4}) = 5b_{2,4} - 20b_{2,1}b_{2,3} + 30b_{2,2}b_{2,1}^2 - 15b_{2,1}^4$$
is positive for all points on $\mathcal{M}_{(3,2)}$ and zero on $\mathcal{M}_{(5)}$.

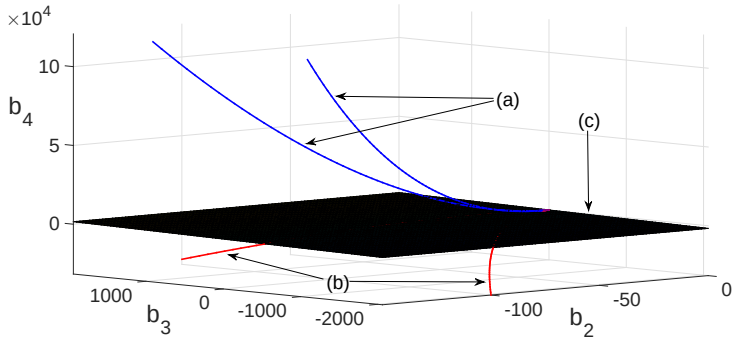


FIG. 2. (a) *The manifold* $\mathcal{M}_{(3,2)}$, $[b_2, b_3, b_4] = [b_{2,2}, b_{2,3}, b_{2,4}]$, (b) *the manifold* $\mathcal{M}_{(4,1)}$, $[b_2, b_3, b_4] = [b_{1,2}, b_{1,3}, b_{1,4}]$, *and* (c) *their separating plane, for a monic Bernstein polynomial of degree five and* $b_{1,1} = b_{2,1} = 0.2$.

TABLE 1
*The stages in the algorithms of Musser and Gauss, and the correspondence of their numerical and geometric operations, for the computation of multiple roots of* $f(y)$.

| Stage | Numerical operation | Geometric operation |
|---|---|---|
| Computation of the multiplicities of the roots of $f(y)$ | GCD computations and polynomial deconvolutions | Identification of the pejorative manifold $\mathcal{M}$ |
| Computation of the values of the distinct roots of $f(y)$ | Computation of the roots of polynomial equations. All the roots are simple. | Identification of the point on $\mathcal{M}$ that defines $f(y)$ |

It follows that the surface $s(c) = 0$ is a separating surface of $\mathcal{M}_{(4,1)}$ and $\mathcal{M}_{(3,2)}$, where $c = [c_1, c_2, c_3, c_4]$.

Figure 2 shows the manifolds $\mathcal{M}_{(4,1)}$ and $\mathcal{M}_{(3,2)}$, and their separating plane for $b_{1,1} = b_{2,1} = 0.2$. Each manifold lies strictly on one side of the separating plane, which is tangential to the manifolds at the point that defines the manifold $\mathcal{M}_{(5)}$ because the distinct roots in each manifold $\mathcal{M}_{(4,1)}$ and $\mathcal{M}_{(3,2)}$ coalesce at this point to form a root of multiplicity five.

**4.1. The algorithms of Musser and Gauss, and the pejorative manifold.** It was shown in section 3 that the difference between the unstructured and structured condition numbers of a multiple root of a polynomial provides numerical justification for the algorithms of Musser and Gauss. The first stage in these algorithms requires GCD computations and polynomial deconvolutions, and the second stage requires the solution of a set of polynomial equations, all of whose roots are simple. It was shown that the indices $i$ in the **while** loop in lines 7–13 in Musser's algorithm, and in the **for** loop in lines 13–16 in Gauss' algorithm, define the multiplicities of the roots of $f(y)$ and therefore also the pejorative manifold on which $f(y)$ lies. This correspondence between the numerical and geometric operations in the algorithms of Musser and Gauss is shown in Table 1.

If the polynomial $f(y)$ is exact, then it is represented by a point on a pejorative manifold $\mathcal{M}$, and the GCD computations and polynomial deconvolutions can be computed exactly. If, however, the coefficients of $f(y)$ are inexact, it can be assumed all its roots are simple and it is therefore not represented by a point on $\mathcal{M}$. In this case,

the GCD computations are replaced by AGCD computations, and these computations and the polynomial deconvolutions define a series of projections onto a manifold $\mathcal{M}^*$ that is justified by the given inexact polynomial. This manifold is not necessarily the manifold on which the theoretically exact form of the given polynomial lies, but the values and multiplicities of the roots computed from $\mathcal{M}^*$ are legitimate solutions if their backward error is acceptable.

**5. The Sylvester matrix and GCD computations.** The GCD computations in the algorithms of Musser and Gauss, and NAClab, use the Sylvester matrix and its subresultant matrices, and it is therefore appropriate to consider these matrices for the polynomials $f(y)$ and $g(y)$,

$$f(y) = \sum_{i=0}^{m} a_i \binom{m}{i} (1-y)^{m-i} y^i \qquad \text{and} \qquad g(y) = \sum_{i=0}^{n} b_i \binom{n}{i} (1-y)^{n-i} y^i. \tag{8}$$

The Sylvester subresultant matrix $S_k(f,g)$, $k = 1, \ldots, \min(m,n)$, of $f(y)$ and $g(y)$ is a matrix of order $(m+n-k+1) \times (m+n-2k+2)$ [3, 4, 24],

$$S_k(f,g) = D_k^{-1} T_k(f,g), \tag{9}$$

where $D_k^{-1} \in \mathbb{R}^{(m+n-k+1)\times(m+n-k+1)}$ is given by

$$D_k^{-1} = \operatorname{diag}\left[ \frac{1}{\binom{m+n-k}{0}} \quad \frac{1}{\binom{m+n-k}{1}} \quad \cdots \quad \frac{1}{\binom{m+n-k}{m+n-k}} \right], \tag{10}$$

$T_k(f,g) \in \mathbb{R}^{(m+n-k+1)\times(m+n-2k+2)}$ is given by

$$T_k(f,g) = \begin{bmatrix} a_0\binom{m}{0} & & & b_0\binom{n}{0} & & \\ a_1\binom{m}{1} & \ddots & & b_1\binom{n}{1} & \ddots & \\ \vdots & \ddots & a_0\binom{m}{0} & \vdots & \ddots & b_0\binom{n}{0} \\ \vdots & \ddots & a_1\binom{m}{1} & \vdots & \ddots & b_1\binom{n}{1} \\ a_m\binom{m}{m} & \ddots & \vdots & b_n\binom{n}{n} & \ddots & \vdots \\ & \ddots & \vdots & & \ddots & \vdots \\ & & a_m\binom{m}{m} & & & b_n\binom{n}{n} \end{bmatrix} = \begin{bmatrix} F_k(f) & G_k(g) \end{bmatrix}, \tag{11}$$

and $F_k(f) \in \mathbb{R}^{(m+n-k+1)\times(n-k+1)}$ and $G_k(g) \in \mathbb{R}^{(m+n-k+1)\times(m-k+1)}$ are Tœplitz matrices whose entries are the coefficients of $f(y)$ and $g(y)$, scaled by their combinatorial terms. The Sylvester matrix $S(f,g)$ is defined by $k = 1$, $S(f,g) = S_1(f,g)$. The application of the matrices $S_k(f,g), k = 1, \ldots, \min(m,n)$, to the calculation of the degree of the GCD of $f(y)$ and $g(y)$ is considered in Theorem 5.1 [3].

THEOREM 5.1. *The degree $t$ of the GCD of $f(y)$ and $g(y)$ is equal to the largest integer $k$ such that $S_k(f,g)$ is rank deficient,*

$$\begin{aligned} &\text{rank } S_k(f,g) < m+n-2k+2, && k = 1, \ldots, t, \\ &\text{rank } S_k(f,g) = m+n-2k+2, && k = t+1, \ldots, \min(m,n), \end{aligned} \tag{12}$$

*and the rank loss of $S(f,g)$,*

$$t = m + n - \text{rank } S(f,g). \tag{13}$$

*Furthermore, the quotient polynomials $v_k(y)$ and $u_k(y)$, which are of degrees $n-k$ and $m-k$, respectively, satisfy $f(y)v_k(y) = g(y)u_k(y)$, $k = 1, \ldots, t$, and their coefficient vectors are computed from vectors that lie in the nullspace of $S_k(f,g)$,*

$$S_k(f,g)p_k = 0, \qquad p_k = Q_k r_k \neq 0, \qquad k = 1, \ldots, t,$$

*where $r_k, k = 1, \ldots, t$, contains the coefficients of $v_k(y)$ and $u_k(y)$,*

$$r_k = \begin{bmatrix} -v_{k,0} & -v_{k,1} & \cdots & -v_{k,n-k} & u_{k,0} & u_{k,1} & \cdots & u_{k,m-k} \end{bmatrix}^T,$$

*and*

$$Q_k = \operatorname{diag}\left[ \binom{n-k}{0} \quad \binom{n-k}{1} \quad \cdots \quad \binom{n-k}{n-k} \quad \binom{m-k}{0} \quad \binom{m-k}{1} \quad \cdots \quad \binom{m-k}{m-k} \right]. \tag{14}$$

The calculation of the degree $t$ of the GCD in (12) is valid for the Sylvester matrix and its subresultant matrices $S_k(f,g) = D_k^{-1} T_k(f,g)$, and their variants,

$$\left\{ T_k(f,g), \quad D_k^{-1} T_k(f,g), \quad T_k(f,g) Q_k, \quad D_k^{-1} T_k(f,g) Q_k \right\}, \quad k = 1, \ldots, \min(m,n), \tag{15}$$

because $D_k^{-1}$ and $Q_k$, which are defined in (10) and (14), respectively, are nonsingular. It is shown in [3, 4] that the best form of the Sylvester matrix and its subresultant matrices is $D_k^{-1} T_k(f,g) Q_k$ because it yields significantly better results than the other forms in (15) for the degree and coefficients of the GCD of $f(y)$ and $g(y)$. This result follows from consideration of the combinatorial terms, which are of the forms

$$\begin{array}{lcll} \binom{m}{i} & \text{and} & \binom{n}{j} & \text{for} \quad T_k(f,g), \\ \dfrac{\binom{m}{i}}{\binom{m+n-k}{p}} & \text{and} & \dfrac{\binom{n}{j}}{\binom{m+n-k}{q}} & \text{for} \quad D_k^{-1} T_k(f,g), \\ \binom{m}{i}\binom{n-k}{p-i} & \text{and} & \binom{n}{j}\binom{m-k}{q-j} & \text{for} \quad T_k(f,g) Q_k, \\ \dfrac{\binom{m}{i}\binom{n-k}{p-i}}{\binom{m+n-k}{p}} & \text{and} & \dfrac{\binom{n}{j}\binom{m-k}{q-j}}{\binom{m+n-k}{q}} & \text{for} \quad D_k^{-1} T_k(f,g) Q_k, \end{array} \tag{16}$$

where $i = 0, \ldots, m, p = i, \ldots, n-k+i$, and $j = 0, \ldots, n, q = j, \ldots, m-k+j$. The minimum range of magnitude (the ratio of the maximum value to the minimum value) of these combinatorial terms for the values of $(i,p)$ and $(j,q)$ occurs for the form $D_k^{-1} T_k(f,g) Q_k$, which is therefore the preferred form, but this range may still be significant such that numerical problems may still occur. These numerical problems of the matrices in (15) can be reduced by processing $f(y)$ and $g(y)$ by three operations before the matrices are formed, but differences between them still exist such that some matrices yield better results than other matrices [3, 4]. These three preprocessing operations are:

**First operation.** The normalization of the nonzero entries in the left partition of the matrices (15), that is, the partition that contains the coefficients $a_i$ of $f(y)$, by their geometric mean, and similarly, the normalization of the nonzero entries in the right partition (the partition that contains the coefficients $b_i$ of $g(y)$) by their geometric mean.

**Second operation.** The replacement of $g(y)$ by $\alpha g(y)$ where $\alpha$ is a constant. This operation follows from the scale invariance property of the GCD of $f(y)$ and $g(y)$, $\operatorname{GCD}(f,g) \sim \operatorname{GCD}(f, \alpha g)$, where $\sim$ denotes equivalence to within an arbitrary nonzero scalar multiplier.

**Third operation.** A change in the independent variable from $y$ to $w$,

$$y = \theta w, \tag{17}$$

where $\theta$ is a constant and the optimal values of $\alpha$ and $\theta$ are obtained from the solution of a linear programming problem.

These operations yield revised polynomials $\tilde{f}_k(w)$ and $\tilde{g}_k(w)$, $k = 1, \ldots, \min(m, n)$,

$$\tilde{f}_k(w) = \sum_{i=0}^{m} \left(\tilde{a}_i \theta_k^i\right) \binom{m}{i} (1 - \theta_k w)^{m-i} w^i, \qquad \tilde{a}_i = \frac{a_i}{\lambda_k},$$

$$\tilde{g}_k(w) = \alpha_k \sum_{i=0}^{n} \left(\tilde{b}_i \theta_k^i\right) \binom{n}{i} (1 - \theta_k w)^{n-i} w^i, \qquad \tilde{b}_i = \frac{b_i}{\nu_k},$$

where $\lambda_k$ and $\nu_k$ are the geometric means calculated in the first preprocessing operation, and the subscript $k$ is included in these means, and in $\alpha_k$ and $\theta_k$, to emphasize that they must be calculated for each subresultant matrix, that is, for each value of $k$. The GCD computations are therefore performed using the matrices (15) that are functions of the polynomials $(\tilde{f}_k(w), \tilde{g}_k(w))$, rather than the polynomials $(f(y), g(y))$. The application of these matrices to the computation of an AGCD of $\tilde{f}(w) = \tilde{f}_1(w)$ and $\tilde{g}(w) = \tilde{g}_1(w)$ is discussed in section 6.

The data in practical problems is inexact, and thus the GCD computations must be replaced by AGCD computations. An AGCD of two or more polynomials is not unique, and it therefore differs from their GCD, which is unique up to a nonzero scalar multiplier. An AGCD of two polynomials is considered in section 6.

**6. An AGCD of two polynomials.** The definition of an AGCD of two or more polynomials includes some or all of the concepts of nearness, maximum degree, and minimum distance [30]. These properties are included in the following definition of an AGCD of the polynomials $p(y)$ and $q(y)$ [14].

DEFINITION 6.1 (an AGCD). *A polynomial $d(y)$ of degree $t$ is an AGCD of $p(y)$ and $q(y)$ if it is the polynomial of maximum degree that is an exact divisor of $p(y) + \tilde{p}(y)$ and $q(y) + \tilde{q}(y)$ for perturbations $\|\tilde{p}\| \leq \varepsilon_p$ and $\|\tilde{q}\| \leq \varepsilon_q$, and $\|\tilde{p}\|^2 + \|\tilde{q}\|^2$ is minimized over all polynomials of degree $t$.*

Two methods for the calculation of the degree $t$ of an AGCD of $\tilde{f}(w)$ and $\tilde{g}(w)$ are described [3, 25]. In particular, $t$ can be computed from their Sylvester matrix and subresultant matrices, as shown in (12), or from the singular value decomposition of the matrices in the set (15) for $k = 1$, as shown in (13). The methods that use subresultant matrices define a measure $\mu_k$ for the distance to singularity of each subresultant matrix, and the maximum change in $\mu_k$ between two successive subresultant matrices is calculated. This leads to the following expression for the calculation of $t$,

$$t = \arg \max_{k=1,\ldots,\min(m,n)-1} \frac{\kappa(S_k(\tilde{f}_k, \tilde{g}_k))}{\kappa(S_{k+1}(\tilde{f}_{k+1}, \tilde{g}_{k+1}))}, \tag{18}$$

where $\kappa(X)$ is the condition number of $X$. This expression for the degree of an AGCD is stated in terms of the Sylvester matrix and its subresultant matrices, but it is clear that this matrix can be replaced by any of the matrices in the set (15). The best matrix is determined by the effectiveness with which computations can be performed on matrices whose entries vary widely in magnitude and its insensitivity to perturbations in the coefficients of the polynomials.

*Example* 6.2. Consider the Bernstein forms of the polynomials $f(y)$ and $g(y)$ whose GCD $d(y)$ is of degree 23,

$$f(y) = (y-0.10)^6(y-0.56)^8(y-0.75)^{10}(y-0.82)^3(y+0.27)^3(y-1.37)^3 \times (y-1.46)^2,$$
$$g(y) = (y-0.10)^2(y-0.56)^8(y-0.75)^{10}(y-0.99)^4(y-2.12)(y-1.20)^3 \times (y-1.37)^3.$$

Uniformly distributed random noise was added to the coefficients $a_i$ and $b_j$ of, respectively, $f(y)$ and $g(y)$, such that the upper bound of the relative error in each coefficient was a uniformly distributed random variable in the interval $\mathcal{I} = [10^{-10}, 10^{-8}]$,

$$\delta a_i = a_i r_i \varepsilon_i, \qquad i = 0, \ldots, 35, \tag{19}$$
$$\delta b_j = b_j r_j \varepsilon_j, \qquad j = 0, \ldots, 31, \tag{20}$$

where $\varepsilon_i, \varepsilon_j \in \mathcal{I}$, and $r_i$ and $r_j$ are uniformly distributed random variables in the interval $[-1, 1]$. The nonconstant value of the upper bound of the relative error in the coefficients $a_i$ and $b_j$ makes it difficult to impose a threshold for the determination of the rank of each matrix in the set (15), and it therefore provides a stringent test for the numerical methods that implement Theorem 5.1.

The polynomials $f(y)$ and $g(y)$ were preprocessed, as described above, and each of the matrices in (15) was formed. Figure 3 shows the singular values of $D_k^{-1}T_k(\tilde{f}_k, \tilde{g}_k)$, $k = 1, \ldots, 31$, and it is clear that (a) the matrix $D_1^{-1}T_1(\tilde{f}_1, \tilde{g}_1)$ is ill-conditioned and of full rank and thus (13) is not satisfied, and (b) the change in the condition number of successive subresultant matrices, as shown in (18), returns the correct degree of the GCD, $t = 23$.

Figure 4 shows the singular values of the matrices $D_k^{-1}T_k(\tilde{f}_k, \tilde{g}_k)Q_k$, and they yield better results than the matrices $D_k^{-1}T_k(\tilde{f}_k, \tilde{g}_k)$ because (a) the rank loss of $D_1^{-1}T_1(\tilde{f}_1, \tilde{g}_1)Q_1$ is 23, which is the degree of the GCD of $f(y)$ and $g(y)$, and (b) the maximum change in the condition number of the subresultant matrices occurs when $k = 23$. The matrices $T_k(\tilde{f}_k, \tilde{g}_k)$ and $T_k(\tilde{f}_k, \tilde{g}_k)Q_k$ yielded unsatisfactory results because (12) and/or (13) were not satisfied.

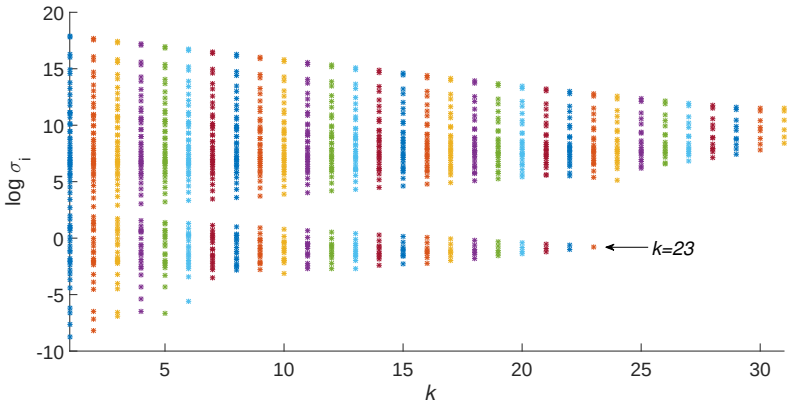


FIG. 3. *The singular values* $\sigma_i$ *of* $D_k^{-1}T_k(\tilde{f}_k, \tilde{g}_k)$, $k = 1, \ldots, 31$, *with the inclusion of the three preprocessing operations, for Example* 6.2.

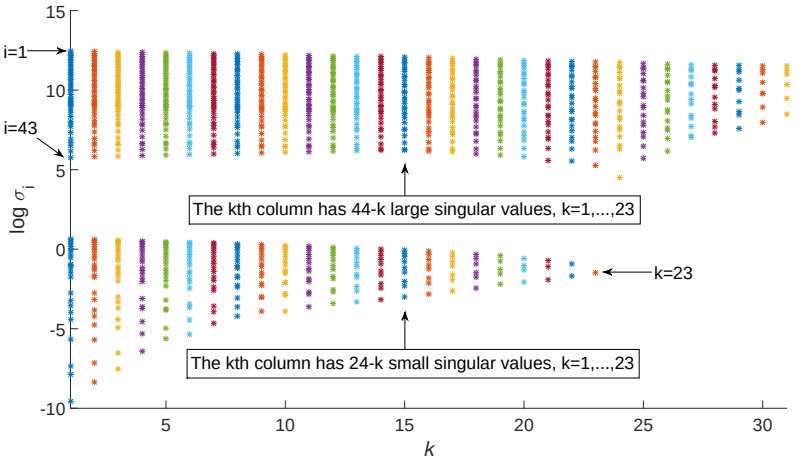


FIG. 4. *The singular values* $\sigma_i$ *of* $D_k^{-1}T_k(\tilde{f}_k, \tilde{g}_k)Q_k$, $k = 1, \ldots, 31$, *with the inclusion of the three preprocessing operations, for Example* 6.2.

Several other points are noted:

1. The superior results obtained with the matrices $D_k^{-1}T_k(\tilde{f}_k, \tilde{g}_k)Q_k$ are consistent with the results in [3].
2. The matrices $D_k^{-1}T_k(\tilde{f}_k, \tilde{g}_k)Q_k$ are the most robust in the presence of noise because a higher noise level can be imposed on the coefficients of $f(y)$ and $g(y)$ before incorrect results are obtained.
3. The clear gap between the subresultant matrices that are, and are not, rank deficient in Figure 4 does not require a threshold for its determination, even though the upper bound of the relative error in the coefficients of $f(y)$ and $g(y)$ is a random variable that spans two orders of magnitude, as shown in (19) and (20). This is a measure of the efficacy of $D_k^{-1}T_k(\tilde{f}_k, \tilde{g}_k)Q_k$ for the calculation of the degree of the GCD of $f(y)$ and $g(y)$.

The results in Figures 3 and 4 are typical of the results obtained from many other examples, and the best form of the Sylvester matrix and its subresultant matrices for GCD computations is thus $D_k^{-1}T_k(\tilde{f}_k, \tilde{g}_k)Q_k$. These results are therefore consistent with the numerical considerations of the range of magnitudes of the combinatorial terms (16) in the variants of the Sylvester matrix and its subresultant matrices.

The calculation of the degree of the GCD of $f(y)$ and $g(y)$ from (12) requires that a change in the rank of two successive subresultant matrices exist, but this change does not occur in the following situations:

1. The polynomials $f(y)$ and $g(y)$ are coprime, in which case all the subresultant matrices have full rank.
2. If $\text{GCD}\,(f, g) = f(y)$ or $\text{GCD}\,(f, g) = g(y)$, all the subresultant matrices are rank deficient.

The first case arises when all the roots of the given polynomial are simple, a situation that is not considered in this paper because the algorithms of Musser and Gauss are explicitly designed for the computation of multiple roots and they do not have any advantages, with respect to traditional methods, for the computation of simple roots. The second case reveals an important difference between the algorithms of Musser and Gauss, and it is considered in detail in the examples in section 9.

**6.1. Bounds on the degree of the GCD of a polynomial and its derivative.** The algorithm of Gauss requires many computations of the GCD of a polynomial $f_i(y)$ and its derivative $f_i^{(1)}(y)$,

$$f_{i+1}(y) = \mathrm{GCD}\left(f_i, f_i^{(1)}\right), \qquad i = 0, 1, 2, \ldots, \tag{21}$$

and an essential part of this computation is the determination of the degree of $f_{i+1}(y)$. If $f_i(y)$ is of degree $p$, then Theorem 5.1 suggests that the rank deficiency, or otherwise, of $p-1$ subresultant matrices must be determined, but Theorem 6.3 shows that significantly fewer subresultant matrices need be considered because bounds on the degree of $f_{i+1}(y)$ in terms of the degree of $f_i(y)$ can be established.

THEOREM 6.3. *Let the polynomials $f_i(y), i = 0, 1, 2, \ldots,$ satisfy* (21)*, and let the degree of $f_i(y)$ be $M_i$. The degree $M_{i+1}$ of $f_{i+1}(y)$ satisfies*

$$2M_i - M_{i-1} \le M_{i+1} \le M_i - 1, \qquad i = 1, 2, \ldots, \tag{22}$$

*which defines lower and upper bounds for the orders of the subresultant matrices that need be considered for the GCD computations* (21).

*Proof.* Let the polynomial $f_i(y)$ have $d_i$ distinct roots $\alpha_j, j = 1, \ldots, d_i$, such that $\alpha_j$ is of multiplicity $m_j$,

$$f_i(y) = (y-\alpha_1)^{m_1}(y-\alpha_2)^{m_2}\ldots(y-\alpha_{d_i})^{m_{d_i}}, \qquad \sum_{j=1}^{d_i} m_j = M_i,$$

since $f_i(y)$ is of degree $M_i$. It follows that

$$f_i^{(1)}(y) = (y-\alpha_1)^{m_1-1}(y-\alpha_2)^{m_2-1}\ldots(y-\alpha_{d_i})^{m_{d_i}-1}h_i(y),$$

where $h_i(y)$ is of degree $d_i - 1$, and thus from (21),

$$f_{i+1}(y) = (y-\alpha_1)^{m_1-1}(y-\alpha_2)^{m_2-1}\ldots(y-\alpha_{d_i})^{m_{d_i}-1}.$$

It follows that

$$M_{i+1} = \sum_{j=1}^{d_i}(m_j - 1) = M_i - d_i, \tag{23}$$

and since $1 \le d_i \le d_{i-1}$, lower and upper bounds on $M_{i+1}$ can be established:

$$M_i - d_{i-1} \le M_{i+1} \le M_i - 1.$$

It follows from (23) that $d_{i-1} = M_{i-1} - M_i$, and substitution of this equation into the lower bound for $M_{i+1}$ yields (22). ∎

*Example* 6.4. Consider the Bernstein form of the polynomial $f_0(y)$,

$$f_0(y) = (y-0.17523547)^5(y-1.50)^7(y+0.75)^{10}(y-0.10)^3(y+1.2354)^3,$$

and the polynomials $f_{i+1}(y) = \mathrm{GCD}\,(f_i, f_i^{(1)}), i = 0, \ldots, 10,$

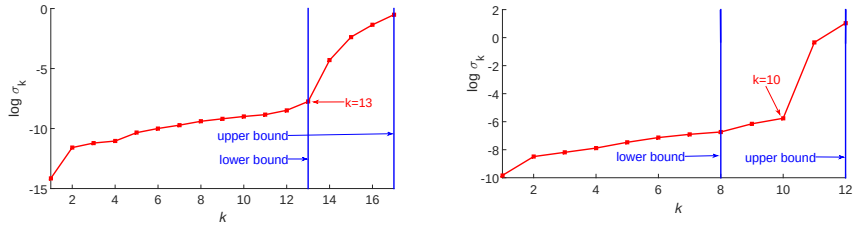


FIG. 5. *The minimum singular value* $\sigma_k$ *of* (a) $D_k^{-1}T_k(\tilde{f}_{2,k}, \tilde{f}_{2,k}^{(1)})Q_k$ *and* (b) $D_k^{-1}T_k(\tilde{f}_{3,k}, \tilde{f}_{3,k}^{(1)})Q_k$, *for Example* 6.4. *The lower and upper bounds of the degree of the GCD, and the degree of the GCD, are marked in the figures.*

$$f_i(y) = \begin{cases} \big((y-0.17523547)^{5-i}(y-1.50)^{7-i}(y+0.75)^{10-i} \times \\ (y-0.10)^{3-i}(y+1.2354)^{3-i}\big), & i=0,1,2, \\ (y-0.17523547)^{5-i}(y-1.50)^{7-i}(y+0.75)^{10-i}, & i=3,4, \\ (y-1.50)^{7-i}(y+0.75)^{10-i}, & i=5,6, \\ (y+0.75)^{10-i}, & i=7,8,9, \\ 1, & i=10. \end{cases}$$

The polynomials $f_0(y)$ and $f_0^{(1)}(y)$ were preprocessed, thereby yielding the polynomials $\tilde{f}_0(w)$ and $\tilde{f}_0^{(1)}(w)$. Figures 5(a) and 5(b) show the minimum singular values $\sigma_k$ of $D_k^{-1}T_k(\tilde{f}_{2,k}, \tilde{f}_{2,k}^{(1)})Q_k$ and $D_k^{-1}T_k(\tilde{f}_{3,k}, \tilde{f}_{3,k}^{(1)})Q_k$, respectively, where $\tilde{f}_{i,k} = \tilde{f}_{i,k}(w)$ denotes the $i$th polynomial in the set $\{f_i(y), i = 0, \ldots, 10\}$ in the $k$th subresultant matrix, after it has been preprocessed. The lower and upper bounds of the degree of the GCD, and the degree of the GCD, are shown in the figures, and it is clear that the bounds reduce the range over which the degree of the GCD is sought.

**6.2. The coefficients of an AGCD.** The calculation of the degree $t$ of an AGCD $d(y)$ of $f(y)$ and $g(y)$ was considered in sections 6 and 6.1, and it allows the coefficients of $d(y)$ to be determined. It is shown in [4] that these coefficients can be computed by applying a nonlinear structure-preserving matrix method to an equation of the form $Ax = b$, where $A$ and $b$ are derived from the $t$th subresultant matrix. The reference includes details on the theoretical development of the equation to be solved, the iterative method for its solution, and computational results.

**7. Deconvolution.** The algorithms of Musser and Gauss require that several polynomial deconvolutions be performed, and this section considers this operation.

The polynomials $f(y)$ and $g(y)$ are defined in (8), and let the coefficients of the polynomial $h(y) = f(y)g(y)$ be $c_i, i = 0, \ldots, m+n$. The deconvolution of $f(y)$ from $h(y)$ can be written as

$$\Big(D^{-1}T(f)\Big)b = c, \tag{24}$$

where

$$D^{-1} = \operatorname{diag}\left[\ \frac{1}{\binom{m+n}{0}} \quad \frac{1}{\binom{m+n}{1}} \quad \cdots \quad \frac{1}{\binom{m+n}{m+n}}\ \right] \in \mathbb{R}^{(m+n+1)\times(m+n+1)},$$

$T(f) \in \mathbb{R}^{(m+n+1)\times(n+1)}$ is a Tœplitz matrix whose entries are the scaled Bernstein coefficients $a_i\binom{m}{i}$, and

$$b = \begin{bmatrix} b_0\binom{n}{0} & b_1\binom{n}{1} & \cdots & b_n\binom{n}{n} \end{bmatrix} \in \mathbb{R}^{n+1},$$
$$c = \begin{bmatrix} c_0 & c_1 & \cdots & c_{m+n} \end{bmatrix} \in \mathbb{R}^{m+n+1}.$$

Previous work [27] and the formation of the modified form of the Sylvester matrix and its subresultant matrices $D_k^{-1}T_k(\tilde{f}_k, \tilde{g}_k)Q_k$, as discussed in section 5, show that it is advantageous to express $b$ as the product of a diagonal matrix $Q \in \mathbb{R}^{(n+1)\times(n+1)}$ and a vector $p \in \mathbb{R}^{n+1}$ of the coefficients $b_i$,

$$b = Qp, \qquad Q = \operatorname{diag}\begin{bmatrix}\binom{n}{0} & \binom{n}{1} & \cdots & \binom{n}{n}\end{bmatrix}, \qquad p = \begin{bmatrix} b_0 & b_1 & \cdots & b_n \end{bmatrix}^T,$$

and thus (24) can be written as

$$\Big(D^{-1}T(f)Q\Big)p = c, \tag{25}$$

where $D^{-1}T(f)Q$ is of order $(m+n+1)\times(n+1)$.

Line 10 in Musser's algorithm and lines 11 and 14 in Gauss' algorithm consist of sequences of deconvolutions,

$$h_i(y) = \frac{f_{i-1}(y)}{f_i(y)} \qquad \text{and} \qquad s_i(y) = \frac{h_i(y)}{h_{i+1}(y)}, \qquad i = 1, 2, \ldots, \tag{26}$$

and there is therefore coupling between successive deconvolutions. This leads to several methods for performing these deconvolutions:

**Method 1: Separate deconvolutions.** The simplest method of performing each deconvolution is least squares, such that the coupling between successive deconvolutions is not imposed. Each deconvolution is therefore independent and of the form (25).

**Method 2: Batch deconvolution.** The deconvolutions (26) are written in a form that combines the individual deconvolutions in Method 1 into one equation. For example, the first set of deconvolutions in (26) can be written as

$$\Big(P_i(f_i)\Big)h_i = f_{i-1}, \qquad P_i(f_i) = D_i^{-1}T(f_i)Q_i, \qquad i = 1, 2, \ldots, r,$$

where $r$ is the number of deconvolutions, $D_i$ and $Q_i$ are diagonal matrices of combinatorial terms, and $T(f_i)$ is a Tœplitz matrix whose entries are the coefficients of $f_i(y)$. This leads to a linear algebraic equation whose coefficient matrix is block-diagonal,

$$\begin{bmatrix} P_1(f_1) & & & \\ & P_2(f_2) & & \\ & & \ddots & \\ & & & P_r(f_r) \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_r \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_{r-1} \end{bmatrix}. \tag{27}$$

This equation is solved by the method of least squares.

**Method 3: Batch deconvolution with structure.** The polynomials $f_i(y)$ may be inexact, and an improved solution of (27) can therefore be obtained by the preservation of the Tœplitz structure of the matrices $T(f_i)$. This preservation is implemented by the method of structured total least norm

(STLN) [17], which requires the addition of perturbations to the polynomials $f_i(y)$ such that the perturbed form of (27) has an exact solution. The application of the method of STLN to (27) therefore yields the equation

$$Ax = b, \tag{28}$$

where

$$A = \operatorname{diag}\begin{bmatrix} P_1(f_1+z_1) & P_2(f_2+z_2) & \cdots & P_r(f_r+z_r) \end{bmatrix},$$
$$x = \begin{bmatrix} \bar{h}_1 & \bar{h}_2 & \cdots & \bar{h}_r \end{bmatrix}^T,$$
$$b = \begin{bmatrix} f_0+z_0 & f_1+z_1 & \cdots & f_{r-1}+z_{r-1} \end{bmatrix}^T,$$

and $z_i = z_i(y), i = 0, \ldots, r$, are polynomials such that the degree of $z_i(y)$ is less than or equal to the degree of $f_i(y)$. The polynomials $z_i(y)$ that constrain (28) to have an exact solution are not unique, but uniqueness is imposed by selecting the polynomials $z_i(y)$ of minimum norm. This leads to a least squares equality problem, which can be solved by the QR decomposition.

**Method 4: Batch deconvolution with constraints.** This method is an extension of Method 2 because constraints are imposed on the polynomials $h_i(y)$ in (27).

*Example* 7.1. Consider the polynomial

$$f(y) = (y-2)^7(y-3)^{12},$$

for which the polynomials $h_i(y)$ in line 11 in Gauss' algorithm satisfy

$$h_1(y) = h_2(y) = \cdots = h_7(y) = (y-2)(y-3), \tag{29}$$
$$h_8(y) = h_9(y) = \cdots = h_{12}(y) = y-3. \tag{30}$$

Some of the polynomials in the solution vector in (27) are therefore equal, and thus this equation can be written as

$$\begin{bmatrix} P_1(f_1) & & \\ \vdots & & \\ P_6(f_6) & & \\ P_7(f_7) & & \\ & P_8(f_8) & \\ & \vdots & \\ & P_{11}(f_{11}) & \\ & P_{12}(f_{12}) & \end{bmatrix} \begin{bmatrix} h_1 \\ h_8 \end{bmatrix} = \begin{bmatrix} f_0 \\ \vdots \\ f_5 \\ f_6 \\ f_7 \\ \vdots \\ f_{10} \\ f_{11} \end{bmatrix}. \tag{31}$$

It cannot be guaranteed that the solution of (27) satisfies (29) and (30), but this problem does not exist when these constraints are imposed.

**Method 5: Constrained batch deconvolution and STLN.** The combination of Methods 3 and 4 enables the method of STLN to be applied to (31).

Example 7.2 considers the different forms of deconvolution when they are applied to a sequence of polynomials $f_i(y)$ that yield the polynomials $h_i(y)$,

$$f_{i+1}(y) = \operatorname{GCD}\left(f_i, f_i^{(1)}\right), \qquad h_{i+1}(y) = \frac{f_i(y)}{f_{i+1}(y)}, \qquad i = 0, 1, \ldots \tag{32}$$

It was shown in section 5 that $f(y)$ and $g(y)$ must be processed before computations are performed on their Sylvester matrix and its subresultant matrices. The objective of the third preprocessing operation (17) is a reduction in the ratio of the maximum entry to the minimum entry of the variants of the Sylvester matrix and its subresultant matrices (15), and an identical reduction is required for the coefficient matrix $D^{-1}T(f)Q$ in (25) because of the combinatorial terms in each matrix in this product.

*Example* 7.2. Noise was added to the Bernstein form of the polynomial

$$f(y) = (y - 2.1234565487)(y - 1.589212457)^4(y - 0.7213)^{10} \times (y - 6.5432)^7(y + 0.72)^{20},$$

such that the upper bound of the relative error in each coefficient was $10^{-8}$, and the polynomials $h_i(y)$ were generated from (32) using the five methods for deconvolution discussed above. The experiment was performed with and without the preprocessing operation, and the relative error in each polynomial $h_i(y)$ was computed.

Figures 6 and 7 show the relative error of each polynomial $h_i(y)$ with and without preprocessing, respectively. The figures show that the inclusion of the preprocessing

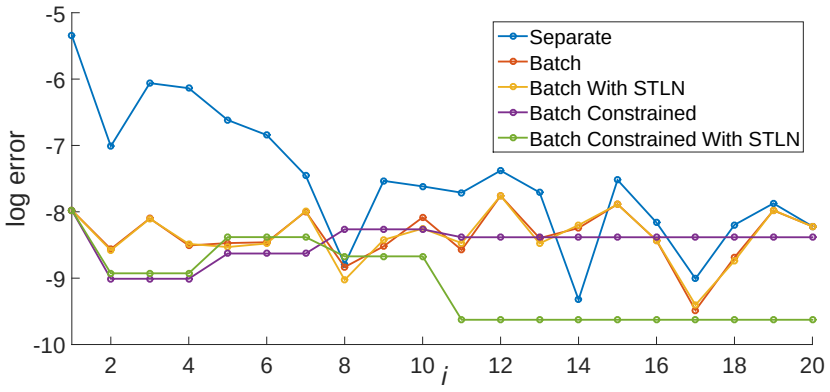


FIG. 6. *The error in the polynomial $h_i(y)$ against $i$, with preprocessing, for Example* 7.2.

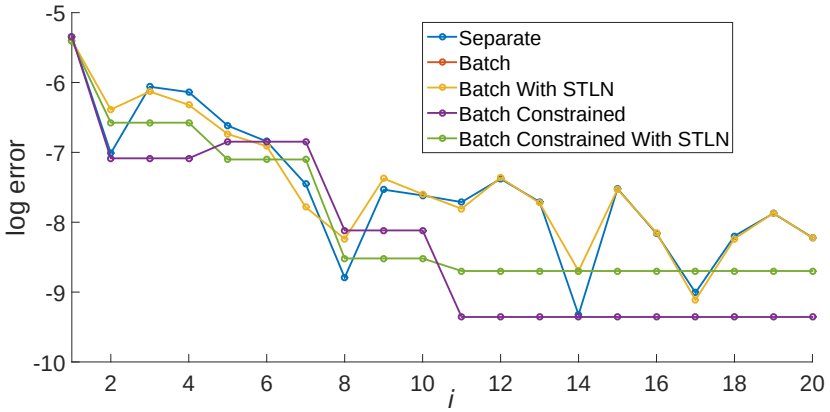


FIG. 7. *The error in the polynomial $h_i(y)$ against $i$, without preprocessing, for Example* 7.2.

operation (17) yields improved results for small values of $i$, but the difference between the results obtained with and without preprocessing decreases as $i$ increases.

**8. Complexity.** The algorithms of Musser and Gauss contain many AGCD computations and polynomial deconvolutions. The AGCD computations on Bernstein basis polynomials are more complicated than their equivalents for power basis polynomials because the Tœplitz structure of the matrices that arise in the power basis is not preserved in the Bernstein basis, as shown in (9), (10), and (11). Many computations showed that the best variant of the set (15) is $D_k^{-1}T_k(\tilde{f}_k, \tilde{g}_k)Q_k$ because the adverse effects of the combinatorial terms are minimized, but the development of fast algorithms that exploit the complicated structure of these matrices is not trivial. It follows from (25) that these comments are also appropriate for the polynomial deconvolutions because the Tœplitz structure of the coefficient matrix for power basis computations is not retained when Bernstein basis polynomials are considered. It follows that generic algorithms must be used for computations on Bernstein basis polynomials, which increases the complexity of the algorithms of Musser and Gauss.

**9. Examples.** This section contains examples in which the results from Musser's algorithm, Gauss' algorithm, and NAClab are compared. The preprocessing operations are implemented for Musser's and Gauss' algorithms, and thus all computations are formed on the polynomial $\tilde{f}_0(w)$ rather than the given polynomial $f_0(y)$. The substitution (3) is required for NAClab in order to transform the given Bernstein basis polynomial to a power basis polynomial.

*Example* 9.1. Noise was added to each coefficient $a_i$ of the Bernstein basis form of the polynomial $f_0(y)$,

$$f_0(y) = \sum_{i=0}^{32} a_i \binom{32}{i} (1-y)^{32-i} y^i = (y-0.10)^{15}(y-0.20)^{15}(y+0.50)^2,$$

such that the relative error in each coefficient was a uniformly distributed random variable in the interval $\mathcal{I} = [10^{-10}, 10^{-8}]$,

$$\delta a_i = a_i r_i \varepsilon_i, \qquad i = 0, \ldots, 32, \tag{33}$$

where $\varepsilon_i \in \mathcal{I}$ and $r_i$ is a uniformly distributed random variable in the interval $[-1, 1]$. The matrices $D_k^{-1}T_k(\tilde{f}_i, \tilde{f}_i^{(1)})Q_k$ were used for the AGCD computations, and the deconvolutions were performed using Method 5 (see section 7). The computed roots from Gauss' algorithm are shown in Table 2, and it is seen that the multiplicities are correct and their relative errors are small. Also, the backward error of the computed roots is $6.16 \times 10^{-10}$.

The experiment was repeated, but $T_k(\tilde{f}_i, \tilde{f}_i^{(1)})Q_k$ and $D_k^{-1}T_k(\tilde{f}_i, \tilde{f}_i^{(1)})$ were used for the AGCD computations, and the same method (Method 5) was used for the

TABLE 2
*The results using Gauss' algorithm with the matrices $D_k^{-1}T_k(\tilde{f}_i, \tilde{f}_i^{(1)})Q_k$ for the AGCD computations and Method* 5 *for the deconvolutions, for Example* 9.1.

| Exact root | Computed root | Exact and computed multiplicities | Relative error of root |
|---|---|---|---|
| 0.10 | 0.100000000474675 | 15, 15 | $4.75 \times 10^{-9}$ |
| 0.20 | 0.200000000940534 | 15, 15 | $4.70 \times 10^{-9}$ |
| −0.50 | −0.500000012859648 | 2, 2 | $2.57 \times 10^{-8}$ |

TABLE 3
*The results using Gauss' algorithm with the matrices $T_k(\tilde{f}_i, \tilde{f}_i^{(1)})Q_k$ for the AGCD computations and Method* 5 *for the deconvolutions, for Example* 9.1.

| Exact root | Computed root | Exact and computed multiplicities | Relative error of root |
|---|---|---|---|
| 0.10 | 0.100000115526782 | 15, 15 | $1.16 \times 10^{-6}$ |
| 0.20 | 0.200000033021166 | 15, 15 | $1.65 \times 10^{-7}$ |
| −0.50 | −0.499997632965895 | 2, 2 | $4.73 \times 10^{-6}$ |

TABLE 4
*The results using Gauss' algorithm with the matrices $D_k^{-1}T_k(\tilde{f}_i, \tilde{f}_i^{(1)})$ for the AGCD computations and Method* 5 *for the deconvolutions, for Example* 9.1.

| Exact root | Computed root | Exact and computed multiplicities | Relative error of root |
|---|---|---|---|
| 0.10 | 0.144336829837851 | 15, 25 | 0.44 |
| 0.20 | 0.166258286195648 | 15, 5 | 0.17 |
| −0.50 | −0.480417676698209 | 2, 2 | 0.04 |

deconvolutions. The results are shown in Tables 3 and 4, respectively, and it is seen that, unlike the matrices $T_k(\tilde{f}_i, \tilde{f}_i^{(1)})Q_k$, the matrices $D_k^{-1}T_k(\tilde{f}_i, \tilde{f}_i^{(1)})$ have not preserved the multiplicity structure of the roots. Furthermore, the backward errors of the roots using $T_k(\tilde{f}_i, \tilde{f}_i^{(1)})Q_k$ and $D_k^{-1}T_k(\tilde{f}_i, \tilde{f}_i^{(1)})$ were $1.61 \times 10^{-7}$ and $5.85 \times 10^{-3}$, respectively, and thus $D_k^{-1}T_k(\tilde{f}_i, \tilde{f}_i^{(1)})$ returned unsatisfactory results.

The roots and multiplicities computed by Musser's algorithm were, respectively, $(-0.25585, 0.14516, 0.43464)$ and $(5, 5, 22)$, and the backward error was 1.97, which is unsatisfactory and therefore requires that the algorithm be considered. In particular, lines 3, 4, and 8 in the algorithm return the polynomials $\tilde{f}_1(w), \tilde{h}_1(w)$, and $\tilde{h}_2(w)$,

$$\tilde{f}_1(w) = \text{GCD}\left(\tilde{f}_0, \tilde{f}_0^{(1)}\right) = (w - 0.10)^{14}(w - 0.20)^{14}(w + 0.5),$$
$$\tilde{h}_1(w) = \frac{\tilde{f}_0(w)}{\tilde{f}_1(w)} = (w - 0.10)(w - 0.20)(w + 0.5),$$
$$\tilde{h}_2(w) = \text{GCD}\left(\tilde{f}_1, \tilde{h}_1\right) = \tilde{h}_1(w).$$

All the subresultant matrices for this GCD computation are therefore rank deficient, and thus a change from rank deficiency to full rank of these matrices does not exist, which is the problem mentioned in section 6. It is therefore very difficult to determine the degree of $\tilde{h}_2(w)$, and furthermore, this problem occurs for other values of $i$ in line 8 of Musser's algorithm.

*Example* 9.2. Noise was added to each coefficient $a_i$ of the Bernstein form of the polynomial $f_0(y)$,

$$f_0(y) = (y - 1.50)^7(y - 0.17523547)^5(y - 0.10)^3(y + 0.75)^{10}(y + 1.2354)^3,$$

such that the relative error in $a_i$ was a uniformly distributed random variable in the interval $\mathcal{I} = [10^{-10}, 10^{-9}]$. The perturbations $\delta a_i$ therefore satisfied (33), except for the change in $\mathcal{I}$. The roots and their multiplicities obtained from Gauss' algorithm are shown in Table 5. The matrices $D_k^{-1}T_k(\tilde{f}_i, \tilde{f}_i^{(1)})Q_k$ were used for the AGCD computations, and the deconvolutions were performed using Method 5. The multiplicities of the roots were correct and the backward error was $9.59 \times 10^{-4}$.

TABLE 5
*The results using Gauss' algorithm with the matrices* $D_k^{-1}T_k(\tilde{f}_i, \tilde{f}_i^{(1)})Q_k$ *for the AGCD computations and Method* 5 *for the deconvolutions, for Example* 9.2.

| Exact root | Computed root | Exact and computed multiplicities | Relative error of root |
|---|---|---|---|
| 1.50 | 1.499 618 583 628 316 | 7, 7 | $5.49 \times 10^{-4}$ |
| 0.175 235 47 | 0.175 241 655 501 572 | 5, 5 | $1.14 \times 10^{-4}$ |
| 0.10 | 0.099 993 088 983 664 | 3, 3 | $1.21 \times 10^{-4}$ |
| −0.75 | −0.749 333 600 186 529 | 10, 10 | $9.46 \times 10^{-4}$ |
| −1.2354 | −1.233 964 468 230 947 | 3, 3 | $1.20 \times 10^{-3}$ |

TABLE 6
*The results using Gauss' algorithm with the matrices* $D_k^{-1}T_k(\tilde{f}_i, \tilde{f}_i^{(1)})Q_k$ *for the AGCD computations and Method* 5 *for the deconvolutions, for Example* 9.3.

| Exact root | Computed root | Exact and computed multiplicities | Relative error of root |
|---|---|---|---|
| 0.10 | 0.099987910107710 | 5, 5 | $1.21 \times 10^{-4}$ |
| 0.30 | 0.299963076284371 | 4, 4 | $1.23 \times 10^{-4}$ |
| 0.50 | 0.500107437647762 | 3, 3 | $2.15 \times 10^{-4}$ |
| 0.70 | 0.699982235427634 | 4, 4 | $2.54 \times 10^{-5}$ |
| 0.90 | 0.899984758853522 | 5, 5 | $1.69 \times 10^{-5}$ |

The experiment was repeated but Method 4 (batch deconvolution with constraints) was used for the deconvolutions, and very similar results were obtained. The matrices $D_k^{-1}T_k(\tilde{f}_i, \tilde{f}_i^{(1)})$ and $T_k(\tilde{f}_i, \tilde{f}_i^{(1)})Q_k$ were then used for the AGCD computations, and very similar results were obtained, using Methods 4 and 5 for the deconvolutions.

Musser's algorithm returned an unsatisfactory result because all the subresultant matrices for some or all of the AGCD computations were rank deficient, as shown in Example 9.1, which makes it very difficult to determine the degree of an AGCD.

*Example* 9.3. Noise was added to each coefficient $a_i$ of the Bernstein form of the polynomial $f_0(y)$,

$$f_0(y) = (y - 0.10)^5(y - 0.30)^4(y - 0.50)^3(y - 0.70)^4(y - 0.90)^5,$$

such that the relative error in $a_i$ was a uniformly distributed random variable in the interval $[10^{-10}, 10^{-9}]$, as in Example 9.2. The procedure described in Examples 9.1 and 9.2 was repeated, and the result from Gauss' algorithm, using the matrices $D_k^{-1}T_k(\tilde{f}_i, \tilde{f}_i^{(1)})Q_k$ to perform the AGCD computations and Method 5 for the deconvolutions, is shown in Table 6. Good results were obtained because the multiplicities of the roots were retained, even though the roots are separated by 0.2. The backward error of the computed roots was $1.88 \times 10^{-5}$.

Musser's algorithm and NAClab returned real simple roots and complex conjugate pairs of roots. The cause of the failure of Musser's algorithm to return a satisfactory result is the rank deficiency of all the subresultant matrices in some of the AGCD computations, as explained in Example 9.1.

The examples show that Gauss' algorithm yields better results than Musser's algorithm, and this was confirmed by other examples.

**10. Summary.** This paper has described the algorithms of Musser and Gauss for the computation of multiple roots of a Bernstein basis polynomial $f(y)$. Their motivation arises from the structured condition number $\rho(\alpha)$ of a multiple root $\alpha$ of $f(y)$, and it was shown that this condition number is many orders of magnitude smaller than the unstructured condition number $\kappa(\alpha)$. This suggests that if the multiplicities of the roots of $f(y)$ are computed initially, then a method for the computation of the values of the distinct roots is stable if the multiplicities of the roots are preserved in this method. This procedure is adopted in the algorithms of Musser and Gauss, which require a series of GCD computations and polynomial deconvolutions. It was shown that the pejorative manifold of a polynomial that has one or more multiple roots establishes a geometric interpretation of the GCD computations and polynomial deconvolutions.

**Acknowledgment.** The authors wish to thank Michael Burr for helpful discussions on the pejorative manifold of a polynomial.

REFERENCES

[1] M. Bartoň and B. Jüttler, *Computing roots of polynomials by quadratic clipping*, Comput. Aided Geom. Design, 24 (2007), pp. 125–141.

[2] D. A. Bini and P. Boito, *A fast algorithm for approximate polynomial GCD based on structured matrix computations*, in Numerical Methods for Structured Matrices and Applications: The Georg Heinig Memorial Volume, D. A. Bini, V. Mehrmann, V. Olshevsky, E. Tyrtshnikov, and M. V. Barel, eds., Birkhäuser, Basel, 2010, pp. 155–173.

[3] M. Bourne, J. R. Winkler, and Y. Su, *The computation of the degree of an approximate greatest common divisor of two Bernstein polynomials*, Appl. Numer. Math., 111 (2017), pp. 17–35.

[4] M. Bourne, J. R. Winkler, and Y. Su, *A non-linear structure-preserving matrix method for the computation of the coefficients of an approximate greatest common divisor of two Bernstein polynomials*, J. Comput. Appl. Math., 320 (2017), pp. 221–241.

[5] J. P. Boyd, *Computing real roots of a polynomial in Chebyshev series form through subdivision with linear testing and cubic solves*, Appl. Math. Comput., 174 (2006), pp. 1642–1658.

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

[7] X. Chen, W. Ma, and Y. Ye, *A rational cubic clipping method for computing real roots of a polynomial*, Comput. Aided Geom. Design, 38 (2015), pp. 40–50.

[8] R. M. Corless, P. M. Gianni, B. M. Trager, and S. M. Watt, *The singular value decomposition for polynomial systems*, in Proceedings of the International Symposium on Symbolic and Algebraic Computation, ACM Press, New York, 1995, pp. 195–207.

[9] R. M. Corless and L. R. Sevyeri, *Approximate GCD in a Bernstein basis*, ACM Commun. Comput. Algebra, 53 (2019), pp. 103–106.

[10] R. M. Corless, S. M. Watt, and L. Zhi, *QR factoring to compute the GCD of univariate approximate polynomials*, IEEE Trans. Signal Process., 52 (2004), pp. 3394–3402.

[11] R. T. Farouki, *On the stability of transformations between power and Bernstein polynomial forms*, Comput. Aided Geom. Design, 8 (1991), pp. 29–36.

[12] W. Kahan, *Conserving Confluence Curbs Ill-Condition*, tech. report, Department of Computer Science, University of California, Berkeley, CA, 1972.

[13] E. Kaltofen, Z. Yang, and L. Zhi, *Structured low rank approximation of a Sylvester matrix*, in Trends in Mathematics, D. Wang and L. Zhi, eds., Birkhäuser-Verlag, Basel, Switzerland, 2006, pp. 69–83.

[14] N. Karmarkar and Y. N. Lakshman, *Approximate polynomial greatest common divsiors and nearest singular polynomials*, in Proceedings of the International Symposium on Symbolic and Algebraic Computation, ACM Press, New York, 1996, pp. 35–39.

[15] J. M. Lane and R. F. Riesenfeld, *Bounds on a polynomial*, BIT, 21 (1981), pp. 112–117.

[16] D. Musser, *Algorithms for Polynomial Factorization*, PhD thesis, The University of Wisconsin - Madison, Madison, WI 1971.

[17] J. B. Rosen, H. Park, and J. Glick, *Total least norm formulation and solution for structured problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 110–128.

[18] T. Sederberg and T. Nishita, *Curve intersection using Bézier clipping*, Comput.-Aided Design, 9 (1990), pp. 538–549.
[19] A. Terui, *GPGCD: An iterative method for calculating approximate GCD of univariate polynomials*, Theoret. Comput. Sci., 479 (2013), pp. 127–149.
[20] J. V. Uspensky, *Theory of Equations*, McGraw-Hill, New York, NY, 1948.
[21] J. R. Winkler, *A companion matrix resultant for Bernstein polynomials*, Linear Algebra Appl., 362 (2003), pp. 153–175.
[22] J. R. Winkler, *High order terms for condition estimation of univariate polynomials*, SIAM J. Sci. Comput., 28 (2006), pp. 1420–1436.
[23] J. R. Winkler, *Structured matrix methods for the computation of multiple roots of a polynomial*, J. Comput. Appl. Math., 272 (2014), pp. 449–467.
[24] J. R. Winkler and R. N. Goldman, *The Sylvester resultant matrix for Bernstein polynomials*, in Curve and Surface Design: Saint-Malo 2002, T. Lyche, M. Mazure, and L. L. Schumaker, eds., Nashboro Press, Brentwood, TN, 2003, pp. 407–416.
[25] J. R. Winkler, M. Hasan, and X. Y. Lao, *Two methods for the calculation of the degree of an approximate greatest common divisor of two inexact polynomials*, Calcolo, 49 (2012), pp. 241–267.
[26] J. R. Winkler, X. Y. Lao, and M. Hasan, *The computation of multiple roots of a polynomial*, J. Comput. App. Math., 236 (2012), pp. 3478–3497.
[27] J. R. Winkler and N. Yang, *Resultant matrices and the computation of the degree of an approximate greatest common divisor of two inexact Bernstein basis polynomials*, Comput. Aided Geom. Design, 30 (2013), pp. 410–429.
[28] W. Wu and Z. Zeng, *The numerical factorization of polynomials*, Found. Comput. Math., 17 (2017), pp. 259–286.
[29] C.-D. Yan and W.-H. Chieng, *Method for finding multiple roots of polynomials*, Comput. Math. Appl., 51 (2006), pp. 605–620.
[30] Z. Zeng, *Computing multiple roots of inexact polynomials*, Math. Comput., 74 (2005), pp. 869–903.
[31] Z. Zeng, *The numerical greatest common divisor of univariate polynomials*, Contemp. Math., 556 (2011), pp. 187–217.
[32] Z. Zeng and T.-Y Li, *NAClab: A* Matlab *Toolbox for Numerical Algebraic Computation*, Extended abstract, 2013.