Università di Pisa

Dipartimento di Informatica

Technical Report : TR-97-08

A hybrid approach to some stability problems for bivariate polynomials

Gemignani L.

May 02, 1997

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

A hybrid approach to some stability problems for bivariate polynomials

Luca Gemignani †

Dipartimento di Informatica, Università di Pisa

Corso Italia 40, 56125 Pisa, Italy.

(6 May 1997)

Frequently, in control system design, we are asked to locate the roots of a bivariate polynomial of the following form

$$H(s,k)=\sum_{i=0}^{n} Q_i(s)k^i \in \mathbf{Z}[k,s]=\mathbf{Z}[k][s]=\mathbf{D}[s],$$

where $Q_i(s)\in \mathbf{Z}[s]$ for each i and, moreover, k is a free parameter ranging in some real interval. For a fixed value $\bar{k}$ of k, the zero distribution of the univariate polynomial $p(s)=H(s,\bar{k})$ with respect to the imaginary axis can be found by determining the inertia of a Bezout matrix $B=B(\bar{k})$ whose entries are expressed in terms of the coefficients of $p(s)$. This evaluation is usually accomplished by computing a block factorization of B, namely, $U^TBU=D$ where D is a block diagonal matrix with lower triangular blocks with respect to the antidiagonal. It is intended in this paper to propose an efficient hybrid approach for determining the zero-distribution of $H(s,k)$ with respect to the imaginary axis for any value of k. We develop a fast fraction-free method for factoring the Bezout matrix $B(k)$ with entries over $\mathbf{D}$ determined by $H(s,k)\in\mathbf{D}[s]$. In this way, we easily compute the sequence $\{\phi_i(k)\}$ of the trailing principal minors of $B(k)$. For almost any value $\bar{k}$ of k the associated sign sequence $\{sign(\phi_i(\bar{k}))\}$ specifies the inertia of $B(\bar{k})$ and, therefore, the zero-distribution of $H(s,\bar{k})$. The function $sign(\phi_i(k))$ is finally obtained by numerically computing rational approximations of the real zeros of $\phi_i(k)\in\mathbf{Z}[k]$.

1. Introduction

A great deal of informations is available in literature on methods for determining the stability of a univariate polynomial $p(s)\in\mathbf{Z}[s]$ of degree n (see [17] and the references given therein). For instance, a lot of different procedures for solving the classical Routh-Hurwitz problem, that is the problem of evaluating the number of zeros of $p(s)$ with positive, zero and negative real parts, had appeared. Most of them rely on the evaluation of the inertia of an associated $n\times n$ Bezout matrix B constructed starting from two polynomials $u(s)$ and $v(s)$ whose coefficients are determined by those of $p(s)$ [7]. Since this

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

computation is generally ill-conditioned, the proposed methods either explicitly assume exact arithmetic or implicitly do so by not considering the effects of rounding errors.

As shown in [11], the inertia of B is determined by the coefficients of the quotients generated by some variation of Euclid's algorithm applied to $u(s)$ and to $v(s)$. This means that, in a polynomial setting, our task can be accomplished, with regard for coefficient growth, by means of subresultant algorithms for computing normal or abnormal polynomial remainder sequences [19].

On the other hand, differently from these algorithms, one may treat the problem by using structured matrix computations. Computationally efficient algorithms for the evaluation of the inertia of B follow from the fast methods for computing an inverse block triangular factorization of JBJ, where J denotes the permutation matrix having unit antidiagonal entries. In the view of a famous law of Sylvester [14] the matrices B and JBJ have the same inertia.

Fast triangular factorization procedures generally work on the quotient field. Recently, many authors have developed suitable variants for factoring structured matrices with entries over integral domains without moving to the quotient fields [6], [10], [12] and [21]. The cost of such algorithms depend on the coefficient growth which can be kept down by sistematically removing factors known to exist from the columns of the upper triangular matrices.

Although the resulting methods are closely related to the subresultant ones [6], [10] by sharing in general the same cost estimates, they present several computational advantages with respect to the polynomial approach. In fact, they uniformly treat both normal and abnormal cases by clearly demonstrating new and known divisibility properties. Furthermore, they immediately permit the symbolic solution of linear algebra problems involving matrices having Hankel and Toeplitz structures.

In this paper we are concerned with zero-location problems for bivariate polynomials. Specifically, we are interested in the solution of the Routh-Hurwitz problem for bivariate polynomials

$$H(s,k) = \sum_{i=0}^{n} Q_i(s)k^i \in \mathbf{Z}[k,s] = \mathbf{Z}[k][s] = \mathbf{D}[s],$$

where $Q_i(s) \in \mathbf{Z}[s]$ for each i and, moreover, k is a free parameter ranging in some real interval, say $k \in [a,b]$. Such a problem frequently arises in control theory where the stability of $H(s,k)$ determines the stability of the feedback system which involves the free parameter k. This is the case, e.g., of root-locus problems in classical control, of higher order root-locus problems of multivariable control and of pole-selection problems of optimal control [15].

Our approach extends the matrix techniques for the univariate case. In particular, from $H(s,k) \in \mathbf{D}[s]$ there follows that a Bezout matrix B with entries over $\mathbf{D}$ can be easily associated with $H(s,k)$ in such a way that for almost any given value of k the inertia of B gives the zero-distribution of $H(s,k)$ with respect to the imaginary axis. The inertia of JBJ and, therefore, of B, can be computed by means of a formula involving the signs of its principal minors $\phi_i(k)$ which are polynomials belonging to $\mathbf{Z}[k]$. We propose to find these polynomials by determining a block triangular factorization $LJBJL^T = D$ which holds for almost any $k \in [a,b]$. Then the sign of $\phi_i(k)$ is specified by evaluating the real zeros of this polynomial numerically.

It is the purpose of this paper to compute the above factorization by the use of algorithms which keep the growth of coefficients down to a reasonable size. This has been successfully done by means of fraction-free Gaussian elimination methods [1]. Differently, in [12] the author proposed a recursive procedure which computes an inverse triangular factorization of a Bezout matrix with integer entries by using certain scaled recurrence relations among the columns of L^T. The resulting method finds both L^T and D with a cost which is asymptotically at least an order of magnitude lower than any known alternative fraction-free method. This approach is here generalized by considering Bezout matrices over more general integral domains and, moreover, by avoiding any assumption on the coefficients of the polynomials which define such matrices.

The remainder of this paper is organized as follows. In Section 2 we introduce some preliminary results about the computation of a block triangular factorization of Bezout matrices and its use for evaluating inertia. In Section 3 we derive our fast fraction-free method for factoring Bezout matrices with entries over **D**. Section 4 provides a cost analysis of the algorithm. Section 5 gives our hybrid method for the computation of the inertia of Bezout matrices associated with bivariate polynomials along with some examples. Finally, the last section includes a discussion of future research directions.

2. Preliminaries

Let $u(s) \in \mathbf{Z}[s]$ and $v(s) \in \mathbf{Z}[s]$ be two polynomial of degree n and m, $n \geq m$, respectively,

$$u(s) = \sum_{i=0}^{n} u_i s^i, \quad v(s) = \sum_{i=0}^{m} v_i s^i, \quad u_n,\ v_m \neq 0.$$

A $n \times n$ Bezout matrix $B = (b_{i,j}) \in \mathbf{Z}^{n \times n}$ associated with $u(s)$ and $v(s)$ is defined by

$$\frac{u(x)v(y) - u(y)v(x)}{x - y} = \sum_{i,j=1}^{n} b_{i,j} x^{i-1} y^{j-1}.$$

The following matrix representation of B can easily be proved [14],

$$\begin{pmatrix} u_1 & u_2 & \dots & u_n \\ u_2 & \iddots & \iddots & \\ \vdots & \iddots & \bigcirc & \\ u_n & & & \end{pmatrix} \begin{pmatrix} v_0 & v_1 & v_2 & \dots \\ & v_0 & v_1 & \ddots \\ & \bigcirc & \ddots & \ddots \\ & & & v_0 \end{pmatrix} -$$

$$\begin{pmatrix} v_1 & v_2 & \dots & v_n \\ v_2 & \iddots & \iddots & \\ \vdots & \iddots & \bigcirc & \\ v_n & & & \end{pmatrix} \begin{pmatrix} u_0 & u_1 & u_2 & \dots \\ & u_0 & u_1 & \ddots \\ & \bigcirc & \ddots & \ddots \\ & & & u_0 \end{pmatrix}, \tag{2.1}$$

where we set $v_j = 0$ of $j > m$. This decomposition clearly shows that the entries of B are polynomials of degree 2 in the coefficients of $u(s)$ and $v(s)$.

The inertia of B is defined by a triplet $In(B) = (\pi(B), \nu(B), \delta(B))$, where $\pi(B)$, $\nu(B)$, and $\delta(B)$ are, respectively, the numbers of eigenvalues of B with positive, negative, and zero real parts. Sylvester's law of inertia [14] says that two Hermitian matrices A_1 and A_2 have the same inertia whenever they are congruent by means of a nonsingular matrix C, that is, $A_1 = C^H A_2 C$. This observation suggests that the fast procedures for determining a triangular factorization of a Bezout matrix can also be used for the efficient evaluation of its inertia.

The following theorem summarizes some results concerning an inverse block triangular factorization of the matrix JBJ over the quotient field $\mathbf{Q}$. Here and hereafter J denotes the $n \times n$ permutation matrix having unit antidiagonal entries. It is worth of mentioning that the assumption $u_n \neq 0$ ensures that JBJ is congruent to a proper Hankel matrix, that is, to a Hankel matrix of rank r such that its principal submatrix of order r is nonsingular [8].

PROPOSITION 2.1. *Let $0 = m_0 < m_1 < m_2 < \ldots < m_L \leq n$ be the integers such that the $k \times k$ leading principal submatrix $(JBJ)_k$ of JBJ is nonsingular if and only if $k = m_i$ for a certain i, $1 \leq i \leq L$. Set $\delta_i = m_i - m_{i-1}$ for $1 \leq i \leq L$, and $\delta_{L+1} = n - m_L$. Let $\mathcal{D}_n$ be the class of the $n \times n$ block diagonal matrices $Diag(D_1, \ldots, D_L, 0)$ where D_i is a $\delta_i \times \delta_i$ lower triangular (with respect to the antidiagonal) Hankel matrix.*

1 There exists an upper triangular matrix R having unit diagonal entries,

$$R = (\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_n), \quad \mathbf{r}_1 = \begin{pmatrix} 1 \\ \mathbf{0} \end{pmatrix}, \ \mathbf{r}_i = \begin{pmatrix} \hat{\mathbf{r}}_i \\ 1 \\ \mathbf{0} \end{pmatrix} \ \text{for } i > 1,$$

such that

$$R^T JBJR = D \in \mathcal{D}_n. \tag{2.2}$$

2 For $i \geq 0$ the vectors $\mathbf{r}_i$, $\mathbf{r}_0 = \mathbf{0}$, satisfy the following recurrence relations:

$$u_n \mathbf{r}_{m_i+1+k} = F\mathbf{r}_{m_i+k}, \quad 1 \leq k \leq \delta_{i+1} - 1, \tag{2.3}$$

$$u_n \mathbf{r}_{m_{i+1}+1} = F\mathbf{r}_{m_{i+1}} + \sum_{k=1}^{\delta_{i+1}} c_{i+1,k}\mathbf{r}_{m_i+k} - \beta_i \mathbf{r}_{m_{i-1}+1}. \tag{2.4}$$

Here F denotes a scaled Frobenius matrix associated with $u(s)$,

$$F = u_n \begin{pmatrix} \mathbf{0}^T & 0 \\ I_{n-1} & \mathbf{0} \end{pmatrix} + \mathbf{e}_1 \begin{pmatrix} -u_{n-1} \\ \vdots \\ -u_0 \end{pmatrix}^T,$$

where $\mathbf{e}_1$ is the first column of the $n \times n$ identity matrix I_n.

The result of 1) is classical and it follows from the theory of Hankel matrices associated with sequences of formal orthogonal polynomials [13]. Recurrence relations (2.3) and (2.4) have been used by the author as a mean for solving the gcd problem for polynomials with integer coefficients [12].

From (2.2) there follows that

$$(JBJ)_{m_i} \begin{pmatrix} \hat{\mathbf{r}}_{m_{i-1}+1} \\ 1 \\ \mathbf{0} \end{pmatrix} = d_1^{(i)} \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}, \quad 2 \leq i \leq L, \tag{2.5}$$

and

$$(JBJ)_{m_i}\hat{\mathbf{r}}_{m_i+1} = -\begin{pmatrix} \tilde{b}_{1,m_i+1} \\ \vdots \\ \tilde{b}_{m_i,m_i+1} \end{pmatrix}, \quad 1 \le i \le L, \tag{2.6}$$

where we set $JBJ = \tilde{B} = (\tilde{b}_{i,j})$. In general, the block triangular factorization (2.2) is not unique but, however, (2.6) says that the vectors $\hat{\mathbf{r}}_{m_i+1}$ are univocally specified.

The computation of the coefficients $c_{i+1,k}$ and β_i of (2.4) can be carried out by means of certain orthogonality conditions among the columns of R. Let us introduce the symmetric bilinear form

$$\langle \mathbf{w}, \mathbf{z} \rangle = \mathbf{w}^T JBJ\mathbf{z}$$

on the space of real n-vectors. Relation (2.2) gives

$$\langle \mathbf{r}_{m_i+h}, \mathbf{r}_{m_j+l} \rangle = 0, \tag{2.7}$$

for $m_i \ne m_j$ and $1 \le h \le \delta_{i+1}$, $1 \le l \le \delta_{j+1}$. Moreover, for $1 \le h, l \le \delta_{i+1}$, we have that

$$\langle \mathbf{r}_{m_i+h}, \mathbf{r}_{m_i+l} \rangle = 0 \tag{2.8}$$

whenever $l + h < \delta_{i+1} + 1$. Otherwise, if $l + h \ge \delta_{i+1} + 1$, then we find that

$$\langle \mathbf{r}_{m_i+h}, \mathbf{r}_{m_i+l} \rangle = d^{(i+1)}_{l+h-\delta_{i+1}} \tag{2.9}$$

where we set

$$D_i = \begin{pmatrix} & & & d_1^{(i)} \\ & \bigcirc & d_1^{(i)} & d_2^{(i)} \\ & \iddots & \iddots & \vdots \\ d_1^{(i)} & d_2^{(i)} & \dots & d_{\delta_i}^{(i)} \end{pmatrix}, \quad d_1^{(i)} \ne 0.$$

By combining the orthogonality properties (2.7), (2.8) and (2.9) with relation (2.4) we obtain that

$$\beta_i \langle \mathbf{r}_{m_{i-1}+1}, \mathbf{r}_{m_i} \rangle = \langle \mathbf{r}_{m_i}, F\mathbf{r}_{m_{i+1}} \rangle.$$

In [12] it is shown that

$$F^T JBJ - JBJF = \mathbf{g}\mathbf{e}_n^T - \mathbf{e}_n\mathbf{g}^T, \tag{2.10}$$

holds for a suitable vector $\mathbf{g}$. Using this result, it can be easily seen that

$$\langle \mathbf{r}_{m_i}, F\mathbf{r}_{m_{i+1}} \rangle = \langle F\mathbf{r}_{m_i}, \mathbf{r}_{m_{i+1}} \rangle,$$

and, therefore,

$$\langle \mathbf{r}_{m_i}, F\mathbf{r}_{m_{i+1}} \rangle = u_n \langle \mathbf{r}_{m_i+1}, \mathbf{r}_{m_{i+1}} \rangle.$$

In this way we obtain that

$$\beta_i = u_n \frac{d_1^{(i+1)}}{d_1^{(i)}}. \tag{2.11}$$

Analogously, we find that the coefficients $c_{i+1,k}$ satisfy the following equations:

$$\sum_{k=\delta_{i+1}-h+1}^{\delta_{i+1}} c_{i+1,k} \langle \mathbf{r}_{m_i+h}, \mathbf{r}_{m_i+k} \rangle = -\langle \mathbf{r}_{m_i+h}, F\mathbf{r}_{m_{i+1}} \rangle, \tag{2.12}$$

with $1 \le h \le \delta_{i+1}$. Notice that these equations define a nonsingular $\delta_{i+1} \times \delta_{i+1}$ triangular linear system in the unknowns $c_{i+1,k}$, $1 \le k \le \delta_{i+1}$.

Sylvester's law implies that

$$In(B) = In(D) = \sum_{i=1}^{L} In(D_i) + (0, 0, n - m_L).$$

The computation of the inertia of the lower triangular (with respect to the antidiagonal) Hankel matrices D_i can be performed according to the following Iohvidov's rule [16]:

$$\pi(D_i) = \begin{cases} \dfrac{\delta_i}{2} & \text{if } \delta_i \text{ is even,} \\ \dfrac{\delta_i + \beta}{2} & \text{if } \delta_i \text{ is odd,} \end{cases}$$

and

$$\nu(D_i) = \begin{cases} \dfrac{\delta_i}{2} & \text{if } \delta_i \text{ is even,} \\ \dfrac{\delta_i - \beta}{2} & \text{if } \delta_i \text{ is odd,} \end{cases}$$

where $\beta = sign(d_1^{(i)})$, $d_1^{(i)}$ is the entry on the antidiagonal of D_i and

$$\mathrm{sign}(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x < 0. \end{cases}$$

Now, let us observe that β can be evaluated by using the signs of the principal minors of JBJ. In fact we find that

$$\det(D_i) = (d_1^{(i)})^{\delta_i} (-1)^{\lfloor \delta_i/2 \rfloor},$$

and, hence,

$$\beta = sign(d_1^{(i)}) = (-1)^{\lfloor \delta_i/2 \rfloor} sign(\det(D_i)) = (-1)^{\lfloor \delta_i/2 \rfloor} \frac{sign(\det(JBJ)_{m_i})}{sign(\det(JBJ)_{m_{i-1}})},$$

whenever δ_i is an odd integer. By combining this result with Iohvidov's rule we obtain that the inertia of JBJ, and, therefore, the inertia of B, is completely specified by the sequence

$$\{sign(\det(JBJ)_i)\}_{i=1}^{n}, \tag{2.13}$$

where we set $sign(\det(JBJ)_i) = 0$ if $(JBJ)_i$ is singular.

In the next section we will consider the problem of determining the leading principal minors $\phi_i(k)$ of a permuted Bezout matrix JBJ with entries over $\mathbf{D} = \mathbf{Z}[k]$ in an efficient way. By computing the real zeros of $\phi_i(k)$, $1 \le i \le n$, numerically, then we may obtain the sequence (2.13) for any value of k. In the view of the above results, this gives a description of the inertia of JBJ for any value of k such that $u_n = u_n(k) \ne 0$.

3. Factoring Bezout matrices over integral domains

Proposition 2.1 leads to a fast method for factoring Bezout matrices over a field. In the sequel of this paper we consider the problem of factoring Bezout matrices associated with polynomials whose coefficients belong to an integral domain without moving to the quotient field. Specifically, assume that $u_i = u_i(k)$ and $v_i = v_i(k)$ are polynomials $\in \mathbf{Z}[k]$,

$u_n = u_n(k)$ not identically zero. Thus, we have that $u(s),\ v(s) \in \mathbf{Z}[k,s] = \mathbf{Z}[k][s] = \mathbf{D}[s]$. Moreover, the permuted Bezout matrix $JBJ = \tilde{B}$ associated with $u(s)$ and $v(s)$ has entries from $\mathbf{D}$.

It's clear that one might use the results of the previous section in order to factorize $\tilde{B}$ over the quotient field $\mathbf{Z}(k)$ (rational functions in k over $\mathbf{Z}$), namely,

$$R^T \tilde{B} R = D, \quad R,\ D \in \mathbf{Z}(k)^{n\times n},\ D \in \mathcal{D}_n.$$

The cost of the resulting algorithm depends on the coefficient growth which can be kept down by removing greatest common divisors. However, gcd computations are expensive and so they should be avoided. An alternative approach which does not require gcd computations consists of finding an inverse triangular factorization of $\tilde{B}$ over $\mathbf{D}$ whereby factors known to exist are systematically removed from the columns of the upper triangular factor. This can always be done by using fraction-free Gaussian elimination methods (Bareiss, (1968)). In the considered case, however, it seems far better to use a sort of structured fraction-free Gaussian elimination based on recurrences (2.3) and (2.4) suitably scaled. In fact, this yields a method asymptotically at least an order faster than the previous ones.

Let us introduce the polynomials $\tilde{b}_i$ defined by

$$\tilde{b}_i = \det(\tilde{B})_{m_i} \in \mathbf{D}, \quad 1 \le i \le L,$$

and $\tilde{b}_0 = 1$. Cramer's rule applied to the linear systems (2.6) implies that the vectors

$$\bar{\mathbf{r}}_{m_i+1} = \tilde{b}_i \begin{pmatrix} \hat{\mathbf{r}}_{m_i+1} \\ 1 \\ \mathbf{0} \end{pmatrix}, \quad 1 \le i \le L-1,$$

belong to $\mathbf{D}^n$. Here and hereafter we consider

$$\bar{\mathbf{r}}_1 = \mathbf{r}_1.$$

Moreover, if $m_L < n$, we set

$$\bar{\mathbf{r}}_{m_L+1} = \tilde{b}_L \begin{pmatrix} \hat{\mathbf{r}}_{m_L+1} \\ 1 \\ \mathbf{0} \end{pmatrix};$$

otherwise, if $m_L = n$, then we define

$$\bar{\mathbf{r}}_{m_L+1} = \tilde{b}_L \hat{\mathbf{r}}_{m_L+1}.$$

Finally, the remaining vectors $\bar{\mathbf{r}}_{m_i+1+j} \in \mathbf{D}^n$, $1 \le j \le \delta_{i+1} - 1$, $0 \le i \le L$, are given by

$$\bar{\mathbf{r}}_{m_i+1+j} = F \bar{\mathbf{r}}_{m_i+j}, \quad 1 \le j \le \delta_{i+1} - 1.$$

This implies that

$$\bar{\mathbf{r}}_{m_i+1+j} = u_n^j \tilde{b}_i \mathbf{r}_{m_i+1+j}, \quad 1 \le j \le \delta_{i+1} - 1, \tag{3.1}$$

holds for $0 \le i \le L$.

The $n \times n$ upper triangular matrix $\bar{R}$,

$$\bar{R} = (\bar{\mathbf{r}}_1, \bar{\mathbf{r}}_2, \ldots, \bar{\mathbf{r}}_n),$$

defines an inverse triangular factorization of $\tilde{B}$ over $\mathbf{D}$. In fact, from

$$\bar{R} = R Diag(S_1, \ldots, S_{L+1}), \quad S_i = \tilde{b}_{i-1} Diag(1, \ldots, u_n^{\delta_i - 1}),$$

there easily follows that

$$\bar{R}^T \tilde{B} \bar{R} = \bar{D}, \quad \bar{D} = Diag(\bar{D}_1, \ldots, \bar{D}_L, 0) \in \mathcal{D}_n.$$

The entry $\bar{d}_1^{(i+1)}$, located on the antidiagonal of $\bar{D}_{i+1}$, is such that

$$\bar{d}_1^{(i+1)} = \langle \bar{\mathbf{r}}_{m_i+1}, \bar{\mathbf{r}}_{m_{i+1}} \rangle = \tilde{b}_i^2 u_n^{\delta_{i+1}-1} d_1^{(i+1)},$$

and, therefore,

$$\bar{d}_1^{(i+1)} = w_i = \tilde{b}_i^2 u_n^{\delta_{i+1}-1} d_1^{(i+1)} \in \mathbf{D}.$$

Since we have

$$\tilde{b}_{i+1} = \prod_{j=1}^{i+1} \det(D_j),$$

we find that

$$\frac{\tilde{b}_{i+1}}{\tilde{b}_i} = \det(D_{i+1}) = (-1)^{L_{i+1}} (d_1^{(i+1)})^{\delta_{i+1}},$$

where $L_{i+1} = \lfloor \delta_{i+1}/2 \rfloor$. Multiplying each side of (2.4) by

$$w_i^{\delta_{i+1}} \tilde{b}_i u_n^{\delta_{i+1}-1} = (\tilde{b}_i^2 u_n^{\delta_{i+1}-1})^{\delta_{i+1}} (-1)^{L_{i+1}} \tilde{b}_{i+1} u_n^{\delta_{i+1}-1},$$

and then using (3.1) we arrive at the following recursions among the vectors $\bar{\mathbf{r}}_j$:

$$(\tilde{b}_i^2 u_n^{\delta_{i+1}-1})^{\delta_{i+1}} (-1)^{L_{i+1}} u_n^{\delta_{i+1}} \bar{\mathbf{r}}_{m_{i+1}+1} =$$

$$= w_i^{\delta_{i+1}} (F \bar{\mathbf{r}}_{m_{i+1}} + \sum_{k=1}^{\delta_{i+1}} c_{i+1,k} u_n^{\delta_{i+1}-k} \bar{\mathbf{r}}_{m_i+k}) - \frac{\beta_i w_i^{\delta_{i+1}} \tilde{b}_i u_n^{\delta_{i+1}-1}}{\tilde{b}_{i-1}} \bar{\mathbf{r}}_{m_{i-1}+1}.$$

Since we have

$$\frac{\beta_i w_i^{\delta_{i+1}} \tilde{b}_i u_n^{\delta_{i+1}-1}}{\tilde{b}_{i-1}} = \frac{w_i^{\delta_{i+1}+1} \tilde{b}_{i-1} u_n^{\delta_i}}{\tilde{b}_i w_{i-1}},$$

we can thus write

$$(\tilde{b}_i^2 u_n^{\delta_{i+1}-1})^{\delta_{i+1}} (-1)^{L_{i+1}} u_n^{\delta_{i+1}} \bar{\mathbf{r}}_{m_{i+1}+1} =$$

$$= w_i^{\delta_{i+1}} (F \bar{\mathbf{r}}_{m_{i+1}} + \sum_{k=1}^{\delta_{i+1}} c_{i+1,k} u_n^{\delta_{i+1}-k} \bar{\mathbf{r}}_{m_i+k}) - \frac{w_i^{\delta_{i+1}+1} \tilde{b}_{i-1} u_n^{\delta_i}}{\tilde{b}_i w_{i-1}} \bar{\mathbf{r}}_{m_{i-1}+1}. \quad (3.2)$$

Quite different recurrences come from using relations (2.5) instead of (2.6).

Regarding at the computation of the coefficients $\bar{c}_{i+1,k}$ given by

$$\bar{c}_{i+1,k} = w_i^{\delta_{i+1}} u_n^{\delta_{i+1}-k} c_{i+1,k}, \quad 1 \le k \le \delta_{i+1}, \quad (3.3)$$

we notice that in the view of (2.12) they satisfy the equations

$$\sum_{k=\delta_{i+1}-h+1}^{\delta_{i+1}} \bar{c}_{i+1,k} \langle \bar{\mathbf{r}}_{m_i+h}, \bar{\mathbf{r}}_{m_i+k} \rangle = -w_i^{\delta_{i+1}} \langle \bar{\mathbf{r}}_{m_i+h}, F \bar{\mathbf{r}}_{m_{i+1}} \rangle,$$

with $1 \le h \le \delta_{i+1}$. Such equations define a $\delta_{i+1} \times \delta_{i+1}$ linear system

$$T \bar{\mathbf{c}} = -w_i^{\delta_{i+1}} \mathbf{f}, \quad T \in \mathbf{D}^{\delta_{i+1} \times \delta_{i+1}}, \ \mathbf{f} \in \mathbf{D}^{\delta_{i+1}}, \quad (3.4)$$

whose coefficient matrix $T = (t_{i,j})$ is a lower triangular Toeplitz matrix, that is, $t_{i,j} = t_{i-j+1}$ and $t_h = 0$ whenever $h \leq 0$. It can be easily shown that

$$t_1 = d_1^{(i+1)} \tilde{b}_i^2 u_n^{\delta_{i+1}-1} = w_i,$$

which gets

$$\det(T) = w_i^{\delta_{i+1}}.$$

Cramer's rule applied to the linear system (3.4) then yields the next result.

PROPOSITION 3.1. *The coefficients $\bar{c}_{i+1,k}$, $1 \leq k \leq \delta_{i+1}$, defined by (3.3) belong to* $\mathbf{D}$.

Taking into account this proposition, we conclude that

$$\frac{w_i^{\delta_{i+1}+1} \tilde{b}_{i-1} u_n^{\delta_i}}{\tilde{b}_i w_{i-1}} \bar{\mathbf{r}}_{m_{i-1}+1} \in \mathbf{D}^n,$$

and, therefore, (3.2) needs computations on $\mathbf{D}$ only. Below we describe a fast algorithm which performs these computations. In the initialization phase one should determine $m_1 = \delta_1$ and $w_0 = \bar{d}_1^{(1)}$. This can be done by means of a direct inspection of the matrix $\tilde{B}$. In fact, we observe that the $m_1 \times m_1$ leading principal submatrix of $\bar{R}$ coincides with S_1 and, therefore, the $m_1 \times m_1$ leading principal submatrix of $\tilde{B}$ has the same structure as $\bar{D}_1$. Specifically, it is a lower triangular Hankel matrix with respect to the antidiagonal with the antidiagonal entry equal to $d_1^{(1)} \in \mathbf{D}$.

Algorithm

Input: a Bezout matrix $B \in \mathbf{D}^{n \times n}$.

Output: the entries of the vectors $\bar{\mathbf{r}}_i$, $1 \leq i \leq n$, which define an inverse block triangular factorization of $\tilde{B} = JBJ$ over $\mathbf{D}$; the sequence $\{\tilde{b}_i\}_{i=0}^{L}$.

Initialize $\bar{\mathbf{r}}_1 = \mathbf{e}_1$, $\bar{\mathbf{r}}_0 = \mathbf{0}$, $m_0 = 0$, $m_{-1} = -1$, $h = 1$, $\tilde{b}_0 = 1$.
Determine m_1 and $d_1^{(1)}$ by direct inspection of the matrix $\tilde{B}$. Thus we have w_0, too.
Compute $\bar{\mathbf{r}}_{1+i}$, $1 \leq i < m_1$, as defined by (3.1). Then find $\bar{\mathbf{r}}_{m_1+1}$ by means of (3.2) and (3.4). Its last nonzero entry gives $\tilde{b}_1$.
For $j = m_1 + 1 \ldots, n-1$, do:

1 evaluate

$$\bar{s} = \bar{\mathbf{r}}_{m_h+1}^T \tilde{B} \bar{\mathbf{r}}_j.$$

If $\bar{s} \neq 0$ then go to 2; otherwise, go to 3;

2 set $m_{h+1} = j$, $\delta_{h+1} = m_{h+1} - m_h$ and $\bar{d}_1^{(h+1)} = w_h = \bar{s}$. Determine the coefficients $\bar{c}_{h+1,i}$, $1 \leq i \leq \delta_{h+1}$ by means of (3.4). Then compute $\bar{\mathbf{r}}_{m_{h+1}+1}$ and, therefore, $\tilde{b}_{h+1}$ by using (3.2). Set $h = h + 1$;

3 compute

$$\bar{\mathbf{r}}_{j+1} = F\bar{\mathbf{r}}_j.$$

endfor

Evaluate

$$\bar{s} = \bar{\mathbf{r}}_n^T \tilde{B} \bar{\mathbf{r}}_n.$$

If $\bar{s} = \bar{d}_1^{(L)} \neq 0$, then set $\delta_L = n - m_h$ and compute $\tilde{b}_L = \det(\tilde{B})_n$ by

$$\tilde{b}_L = (-1)^{\lfloor \delta_L/2 \rfloor} \frac{(\bar{d}_1^{(L)})^{\delta_L} \tilde{b}_{L-1}}{(\tilde{b}_{L-1}^2 u_n^{\delta_L - 1})^{\delta_L}}.$$

In the next section we will perform the cost analysis of this algorithm.

4. The cost analysis

The overall cost of the above algorithm clearly depends on that of the arithmetic operations in $\mathbf{D} = \mathbf{Z}[k]$. In particular, we observe that the number of multiplications/divisions at each step is comparable with the number of additions. Thus, in assessing the overall cost, we can restrict our attention to count the number of multiplications/divisions over $\mathbf{D}$ at each step of our algorithm.

If we assume that the classical algorithm is used for multiplication of two polynomials $a(k) \in \mathbf{D}$ and $b(k) \in \mathbf{D}$ of degree less than n_1 and n_2, respectively, then its arithmetic cost is bounded by $O(n_1 n_2)$. This gets the boolean cost of $O(\nu(m) n_1 n_2)$ whenever the input polynomials have m-bit integer coefficients and, moreover, $\nu(m)$ bit-operations suffice to multiply two integers modulo 2^m. These estimates can be reduced by using fast methods based on FFT. To be specific, one may firstly evaluate the coefficients of the product numerically by means of FFT over $\mathbf{C}$ with $d-$bit, $d \geq m$, precision computation. Then, rounding off will give us the output with no error. Since computing FFT is a well-conditioned problem and, moreover, numerically stable algorithms exist, it can be shown that $d = O(m + \log n)$ suffices [4], where $n = \max\{n_1, n_2\}$. This yields the upper bound of $O(\nu(m + \log n) n \log n)$ boolean operations for fast multiplication in $\mathbf{D}$.

Exact divisions over $\mathbf{D}$ are equivalent in a matrix setting to the solution of lower triangular Toeplitz linear systems. There easily follows that the coefficients of the quotient of $a(k)$ and $b(k)$ can be also determined at the cost of $O(n_1 n_2)$ arithmetic operations. This bound can be greatly improved using a fast method based on evaluation-interpolation on the Fourier set $\{\omega_i\}$. However, some extra care should be taken with respect to the multiplication. In fact, an upper bound on the magnitude of the coefficients of the quotient now involves the entries of the inverse of a lower triangular Toeplitz matrix. In the worst case this means mn-bit integer coefficients in the result. Similar problems arise in the evaluation-interpolation scheme where we measure the absolute error of the IDFT computation in terms of the norm of the vector with entries $a(\omega_i)/b(\omega_i)$. In this way, we find that the exact computation of the coefficients of the quotient generated by the division of $a(k)$ and $b(k)$ can be performed at the cost of $O(\nu(mn) n_1 n_2)$ or $O(\nu(mn + n \log n) n \log n)$ boolean operations.

Of course, these comparisons do not take into account the constants of proportionality of each method. In practice, we recall that fast methods for polynomial multiplication over $\mathbf{C}$ are better than the classical one approximately when $n_1 + n_2 \geq 600$ [20]. Further, the implementation of these fast methods for polynomial multiplication and division requires some bound on the coefficients of the polynomials involved. An apriori analysis clearly leads to unsatisfactory bounds expecially for what concerns the division operation over

D. In practice, it is far better to carry out the divisions in finite precision arithmetic and, then make use of the results in order to bound the magnitude of the desired coefficients. Below, we show that the arithmetic cost of such auxiliary computations is negligible.

In the sequel of this section we derive asymptotic estimates for the arithmetic cost of our algorithm under the assumption that classical methods are used for performing the arithmetic in **D**. In addition, suppose that the coefficients of the starting polynomials $u(s)$ and $v(s)$ are polynomials of degree less than K with integer coefficients. The matrix formula (2.1) says that the entries of $\tilde{B}$ have degree less than $2K$. Concerning the polynomials involved at the j-th step of our algorithm, we easily find that the entries of $\bar{\mathbf{r}}_{m_h+1}$ have degree bounded by $2Km_h$. There follows that the degree of the entries of $\bar{\mathbf{r}}_j$ is less than $2Km_h + K(j - m_h - 1) = K(m_h + j - 1)$ and, therefore, the degree of $\bar{s} = w_h$ is bounded by $K(3m_h + m_{h+1} + 1)$. Finally, to assess the degree of the coefficients $\bar{c}_{h+1,i}$, notice that they are the solution of the linear system (3.4) where both the coefficient matrix T and the known vector $\mathbf{f}$ have entries of degree less than $K(2m_h + 2m_{h+1} + 1)$. The determinant of T will cancel with $w_h^{\delta_{h+1}}$ and so we find the upper bound $K(2m_h + 2m_{h+1} + 1)\delta_{h+1}$.

The evaluation of the entries of T and $\mathbf{f}$ can be carried out using only $O(n\delta_{h+1})$ multiplications. In fact, from (2.10) we obtain that $\tilde{B}\bar{\mathbf{r}}_{j+1} = \tilde{B}F\bar{\mathbf{r}}_j$ can be . computed at the cost of $O(n)$ multiplications of polynomials of degree bounded by that of the entries of the result. A forward substitution algorithm yields the solution of the lower triangular linear system (3.4) at the cost of $O(\delta_{h+1}^2)$ multiplications of polynomials. The cost of finding the coefficients $\bar{c}_{h+1,i}$ is so bounded by

$$O(n\delta_{h+1}K^2m_{h+1}^2 + \delta_{h+1}^3K^2m_{h+1}^2)$$

arithmetic operations. Furthermore,

$$O(m_{h+1}^3\delta_{h+1}^2K^2)$$

arithmetic operations also suffice for computing the coefficients of $\bar{\mathbf{r}}_{m_{h+1}+1}$. Notice that we perform $O(m_{h+1})$ divisions of polynomials of degree less than $O(Km_{h+1}\delta_{h+1})$ and, therefore,

$$O(m_{h+1}^2K\delta_{h+1}(\log k + \log m_{h+1}))$$

arithmetic operations suffice to bound the magnitude of the coefficients of the results.

To estimate the total cost of our algorithm, two cases are taken into consideration. Firstly, we assume that $\delta = \max_i\{\delta_i\} = 1$; this case is generally referred as to the normal case in the literature. Then, the overall cost results to be bounded by $O(n^4K^2)$ arithmetic operations. Clearly, this upper bound generalizes to the case where δ is bounded by a constant independent of n, too. Secondly, consider the case when the step size is not bounded. Since we have

$$\sum_i \delta_i = n,$$

one easily obtains that the overall cost is now bounded by $O(n^5K^2)$ arithmetic operations.

For comparison, fraction-free methods based on Bareiss techniques require $O(n^5K^2)$ arithmetic operations even if we consider the normal case [1].

In order to provide some insight about the boolean cost, one may suppose that the lenght of the coefficients of an exact quotient of polynomials does not exceed the maximum of the length of the coefficients of the dividend [5]. It is further assumed that the lenght of the coefficients of a sum or difference of polynomials is upper bounded by the maximum lenght of the coefficients of the summands, and that the lenght of the coefficients of a product is less than the sum of the maximum lenghts of the coefficients of the factors. Using Hadamard's inequality, we see that the entries of $\bar{\mathbf{r}}_{m_h+1}$ have integer coefficients whose lenght is bounded by

$$O(m_{h+1}(\log m_{h+1} + \log c + \log K)),$$

where the input polynomials of $\tilde{B}$ have integer coefficients bounded in magnitude by c. The same bound holds for the coefficients of the entries of T and $\mathbf{f}$. The integer coefficients of $c_{i+1,k}$ are then seen to be bounded in lenght by

$$O(m_{h+1}\delta(\log m_{h+1} + \log c + \log K)).$$

We conclude that the coefficient growth is usually almost linear. However, in certain pathological cases where $\delta = O(n)$, a quadratic growth can be expected.

5. A hybrid method for the inertia computation

In this section we describe a two-stage procedure for determining the inertia of a Bezout matrix B with entries over $\mathbf{D} = \mathbf{Z}[k]$ where k ranges in some real interval, say $k \in [a, b]$.

In the first phase we apply the algorithm stated in Section 3 in order to find the two sequences

$$\{\tilde{b}_i\}_{i=1}^{L}, \quad \{\delta_i\}_{i=1}^{L}.$$

All the computations are performed over $\mathbf{Z}$ with no error and this part represents the symbolic step of our procedure.

In the second stage we numerically find rational approximations to the real zeros of the polynomials $\tilde{b}_i$, $1 \leq i \leq L$. By direct inspection, we are able to evaluate the sequence (2.13) for any different real value of k. Thus we may determine the inertia of $\tilde{B}$ for any different value of k such that $k \in I = \{k \in [a, b] \ : \ u_n(k) \neq 0\}$. Notice that the sequence $\{\delta_i(\bar{k})\}$, which is to be used to find the inertia of $\tilde{B}$ at the point $k = \bar{k}$, does not generally coincide with the sequence $\{\delta_i\}$ computed at the previous step. However, the former sequence can be immediately retrieved from that of the signs of $\tilde{b}_i$ evaluated at $k = \bar{k}$.

The core of the resulting computational scheme proceeds as follows.

Input: a Bezout matrix $B \in \mathbf{D}^{n\times n}$.

Output: the functions $\pi : \ I \rightarrow \{0, 1, \ldots, n\}$, $\nu : \ I \rightarrow \{0, 1, \ldots, n\}$ and $\delta : \ I \rightarrow \{0, 1, \ldots, n\}$, such that the triplet $(\pi(\bar{k}), \nu(\bar{k}), \delta(\bar{k}))$ defines the inertia of $\tilde{B}$ at the point $k = \bar{k} \in I$.

1 Apply the algorithm of Section 3 to $\tilde{B}$ and so find the integer numbers m_i and the polinomials $\phi_{m_i}(k) = \tilde{b}_i$ for $1 \leq i \leq L$. If $j \in \{1, \ldots, n\}$ and $j \neq m_i$ for any m_i, then set $\phi_j(k) = 0$.

2 Compute numerical (rational) approximations of the real zeros of the polynomials $\phi_{m_i}(k)$ and denote by Δ the set of such zeros lying in the interval $[a, b]$. Evaluate the sequence $\{\phi_i(k)\}_{i=1}^{n}$ of (2.13), and hence the inertia of $\tilde{B}$, at each point of $\Delta \bigcap I$. In this way, reconstruct the desired functions.

In order to give a complete description of the inertia of B for any value of $k \in [a, b]$, this procedure can be complemented with the computation of the inertia of the matrix B evaluated at the rational approximations of the zeros of $u_n(k)$ lying in $[a, b]$. This can be done by invoking a suitable modification of the algorithm stated at the end of Section 3 at the overall cost of $O(Kn^2)$ arithmetic operations.

As we know by the previous section, the degree of $\phi_{m_i}(k)$ is upper bounded by Km_i and therefore Δ has surely less than Kn^2 elements. Moreover, by assuming that the computation of the roots of an n-degree polynomial can be carried out using only $O(n^2)$ arithmetic operations, then we find that step 2 has the cost of $O(K^2n^3)$ arithmetic operations. Therefore, under these assumptions, we may conclude that the overall cost is dominated by that of the symbolic factorization of $\tilde{B}$.

Our hybrid method has been implemented by using the routines provided by *Mathematica* for numerical and algebraic manipulations. In particular, at the second phase we have used the function *NSolve* for the numerical solution of the polynomial equations.

The following three simple examples illustrate our approach.

Example 1. Let us consider the 4×4 Bezout matrix B associated with the polynomials

$$u(s) = 3 + 4s^3 + 4ks^4, \quad v(s) = 1 + 2s + 2ks^2,$$

where k is a free parameter ranging in the interval $[-2, 2]$. We find that

$$JBJ = \begin{pmatrix} 0 & 8k^2 & 8k & 4k \\ 8k^2 & 16k & 8+4k & 4 \\ 8k & 8+4k & 4 & -6k \\ 4k & 4 & -6k & -6 \end{pmatrix}.$$

Thus we have that

$$\phi_1(k) = 0, \ \phi_2(k) = -64k^2, \ \phi_3(k) = 256k^4, \ \phi_4(k) = -1280k^4 + 1536k^5 + 2304k^6.$$

Nsolve yields the following approximations of the nonzero roots of $\phi_4(k) = 0$,

$$x_1 = -1.14983, \quad x_2 = 0.483163.$$

For $2 \leq k < x_1$, the sequence (2.13) is given by $\{0, -1, +1, +1\}$ which leads to $In(B) = (2, 2, 0)$. Analogously, we arrive at the following table which describes the inertia of B when $k \neq 0$.

$$In(B) = \begin{cases} (2,2,0) & \text{if } -2 \leq k < x_1\ , \\ (1,2,1) & \text{if } k = x_1, \\ (1,3,0) & \text{if } x_1 < k < 0, \\ (1,3,0) & \text{if } 0 < k < x_2, \\ (1,2,1) & \text{if } k = x_2, \\ (2,2,0) & \text{if } x_2 \leq k \leq 2. \end{cases}$$

If $k = 0$, JBJ reduces to a nonproper matrix. Applying the algorithm of Section 3, suitably modified for dealing with the scalar case, we find that $In(B) = (1, 2, 1)$ for $k = 0$.

Example 2. Consider the 4×4 Bezout matrix B associated with the polynomials

$$u(s) = 4s^3 + 4ks^4, \quad v(s) = 6s + 6ks^2.$$

The program now returns

$$\phi_1(k) = 0, \ \phi_2(k) = -576k^4, \ \phi_3(k) = 0, \ \phi_4(k) = 0,$$

according to the fact that $u(s)$ and $v(s)$ have a greatest common divisor of degree 2. The problem of computing gcd of polynomials from the inverse triangular factorization of the associated Bezout matrix has been already faced in the scalar case ($B \in \mathbf{Z}^{n\times n}$) in [12]. The development of efficient methods for the multivariate case is under investigation.

Example 3. Let $a(s)$ be a real polynomial. It can be easily seen that the zeros of $a(s)$ belong to the left complex half-plane if and only if the zeros of $p(s) = a(is)$, $i^2 = -1$, completely belong to the upper complex half-plane. A classical result which goes back to Hermite [7] says that this happens if and only if $A = -i\tilde{B}$ is positive definite, where $\tilde{B}$ here denotes the Bezout matrix associated with $p(s)$ and $\bar{p}(s)$. Writing $a(s) = h(s^2) + sg(s^2)$, after some straightforward calculations we obtain that this condition is equivalent to ensure that the Bezout matrix associated with $sg(-s^2)$ and $h(-s^2)$ is positive definite. Based on this result, we have considered the Routh-Hurwitz problem for the polynomial

$$a(s) = k(s+1)^4 + (k-1)s + k,$$

where k is a real parameter. We find that all the roots of $a(s) = 0$ belong to the left half-plane when $k < x_1 = -0.0568731$ or $k > x_2 = 0.279095$. For $k = x_1, x_2$, *Nsolve* returns solutions of $a(s) = 0$ with real parts between 10^{-9} and 10^{-7}.

6. Conclusions and further extensions

In this paper we have developed an efficient recursive procedure for computing an inverse triangular factorization of a Bezout matrix B with entries over an integral domain $\mathbf{D}$ without moving to the quotient field. We can thus determine the sequence of the trailing principal minors of B that completely specifies the inertia of B. Such computation is important in most of the basic algorithms for zero-location problems of polynomials. A hybrid method for computing the inertia of Bezout matrices whose entries are univariate polynomial has been also discussed. There are however several computational issues, concerning both the symbolic that the numerical manipulations, that should be further investigated. An extension of the fast algorithms of [3] and [12] to Bezout matrices over integral domains, rather than fields, seems to be possible. In this way, one may determine a triangular factorization of B using well-known fast fraction-free methods for Gaussian elimination applied to suitable generators only. Replacing inner products and vector operations by matrix ones should produce a speed-up of the resulting algorithm. Moreover, this approach makes feasible to compute the block diagonal matrix D independently of the triangular factors with a great save in cost. Such an observation has been already used for the inertia computation over fields [9] and for the Cauchy index computation of polynomials with integer coefficients [18]. Finally, regarding at the numerical computation of the zeros of the trailing principal minors of B, we notice that these polynomials have usually long coefficients. Therefore, investigations of the use of multiprecision floating point packages for the polynomial root computation, such as that developed by Bini [2] under the European Community Project Frisco for the design and the analysis of multivariate polynomial systems solvers, would also be interesting.

REFERENCES

1 E. Bareiss, *Sylvester's identity and multistep integer-preserving Gaussian elimination,* Math. Comp., **22** (1968), pp. 565-578.
2 D. Bini, *Numerical computation of polynomial zeros by means of Aberth's method*, Numerical Algorithms (to appear).

3 D. Bini, L. Gemignani, *Fast parallel computation of polynomial remainder sequences via Bezout and Hankel matrices*, SIAM J. Comput., **24** (1995), pp. 63-77.
4 D. Bini, V. Pan, *Polynomial and matrix computations, volume 1: Fundamental algorithms*, Birkhäuser, Boston, MA, 1994.
5 W. S. Brown, *On Euclid's algorithm and the computation of polynomial greatest common divisors*, J. ACM, **18** (1971), pp. 478-504.
6 S. Cabay, P. Kossowski, *Power series remainder sequences and Padé fractions over an integral domain*, J. Symbolic Computation, **10** (1990), pp. 139-163.
7 M. Fujiwara, *On algebraic equations whose roots lie in a circle or in a half plane*, Math. Z., **24** (1969), pp. 161-169.
8 F . R. Gantmakher, *Theory of matrices*, Chelsea Publishing Co., 1959.
9 L. Gemignani, *Computing the inertia of Bezout and Hankel matrices*, CALCOLO, **28** (1991), pp. 267-274.
10 L. Gemignani, *Solving Hankel systems over the integers*, J. Symbolic Computation, **18** (1994), pp. 573-584.
11 L. Gemignani, *Computationally efficient applications of the Euclidean algorithm to zero location*, Linear Algebra Appl., **249** (1996), pp. 79-91.
12 L. Gemignani, *GCD of polynomials and Bezout matrices*, TR 97-03, Computer Science Dep., University of Pisa. To appear on the proceedings of ISSAC97.
13 W. B. Gragg, A. Lindquist, *On the partial realization problem*, Linear Algebra Appl., **50** (1983), pp. 277-319.
14 G. Heinig, K. Rost, *Algebraic methods for Toeplitz-like matrices and operators*, Birkhäuser, Boston, MA, (1984).
15 J. G. Hsieh, J. P. Su, C. C. Hsieh, *An application of the path-following method to root-locus problems*, Control Theory and Advanced Technology, **5** (1989), pp. 323-343.
16 I. S. Iohvidov, *Hankel and Toeplitz Matrices and Forms*, Birkhäuser, Boston, MA, (1984).
17 H. Lev-Ari, Y. Bistritz, T. Kailath, *Generalized Bezoutians and families of efficient zero-location procedures*, IEEE Transactions on CIrcuits and Systems, **38** (1991), pp. 170-186.
18 T. Lickteig, M. F. Roy, *Cauchy index computation*, to appear on CALCOLO.
19 R. Loos, *Generalized polynomial remainder sequences*, Computer algebra symbolic and algebraic computations, B. Buchberger, G . E. Collins, R. Loos eds., Springer (1982), pp. 115-137.
20 R. T. Moenck, *Practical fast polynomial multiplication*, in Proc. SYM-SAC'76, ed. R. D. Jenks, ACM Press (1976), pp. 166-176.
21 J. R. Sendra, J. Llovet, *GCD of polynomials and Hankel matrices*, J. Symbolic Computation, **13** (1992), pp. 25-39.