# A non-linear structure preserving matrix method for the low rank approximation of the Sylvester resultant matrix

Joab R. Winkler, [a] Madina Hasan [a]

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

j.winkler@dcs.shef.ac.uk, acp07mah@sheffield.ac.uk

**Abstract**

A non-linear structure preserving matrix method for the computation of a structured low rank approximation $S(\tilde{f}, \tilde{g})$ of the Sylvester resultant matrix $S(f, g)$ of two inexact polynomials $f = f(y)$ and $g = g(y)$ is considered in this paper. It is shown that considerably improved results are obtained when $f(y)$ and $g(y)$ are processed prior to the computation of $S(\tilde{f}, \tilde{g})$, and that these preprocessing operations introduce two parameters. These parameters can either be held constant during the computation of $S(\tilde{f}, \tilde{g})$, which leads to a linear structure preserving matrix method, or they can be incremented during the computation of $S(\tilde{f}, \tilde{g})$, which leads to a non-linear structure preserving matrix method. It is shown that the non-linear method yields a better structured low rank approximation of $S(f, g)$ and that the assignment of $f(y)$ and $g(y)$ is important because $S(\tilde{f}, \tilde{g})$ may be a good structured low rank approximation of $S(f, g)$, but $S(\tilde{g}, \tilde{f})$ may be a poor structured low rank approximation of $S(g, f)$ because its numerical rank is not defined. Examples that illustrate the differences between the linear and non-linear structure preserving matrix methods, and the importance of the assignment of $f(y)$ and $g(y)$, are shown.

*Key words:* Sylvester matrix, structured low rank approximation

## 1 Introduction

Resultant matrices arise in several disciplines that require the processing of curves and surfaces, including computer graphics [7], computer vision [12] and computer aided geometric design. They are frequently used in geometric problems because they can be used to determine if two polynomial curves intersect, and thus the points of intersection are calculated only if the curves intersect.

In particular, a resultant matrix, the entries of which are functions of the coefficients of the polynomials, is singular if and only if the curves intersect. Although design intent may require that the curves intersect, inexact data may imply they do not intersect, in which case the design intent is realised by perturbing the coefficients of the polynomials slightly such that their resultant matrix becomes singular, that is, a structured low rank approximation of the given resultant matrix is required. This paper compares the methods of structured total least norm (STLN) [13] and structured non-linear total least norm (SNTLN) [14] for the calculation of a structured low rank approximation of the Sylvester resultant matrix, which is one type of resultant matrix.

The Sylvester resultant matrix $S(f,g) \in \mathbb{R}^{(m+n)\times(m+n)}$ of the polynomials $f = f(y)$ and $g = g(y)$,

$$f(y) = \sum_{i=0}^{m} a_i y^{m-i} \qquad \text{and} \qquad g(y) = \sum_{i=0}^{n} b_i y^{n-i}, \qquad a_0, b_0 \neq 0, \tag{1}$$

is

$$S(f,g) = \begin{bmatrix} a_0 & & & b_0 & & \\ a_1 & \ddots & & b_1 & \ddots & \\ \vdots & \ddots & a_0 & \vdots & \ddots & b_0 \\ a_{m-1} & \ddots & a_1 & b_{n-1} & \ddots & b_1 \\ a_m & \ddots & \vdots & b_n & \ddots & \vdots \\ & \ddots & a_{m-1} & & \ddots & b_{n-1} \\ & & a_m & & & b_n \end{bmatrix}, \tag{2}$$

where the coefficients $a_i$ of $f(y)$ occupy the first $n$ columns and the coefficients $b_i$ of $g(y)$ occupy the last $m$ columns.

The calculation of a structured low rank approximation of $S(f,g)$ is closely related to the calculation of an approximate greatest common divisor (AGCD) of $f(y)$ and $g(y)$. For example, Bini and Boito [1] discuss three methods, based on the structure of the Sylvester $S(f,g)$ and Bézout $B(f,g)$ resultant matrices, for AGCD computations. The QR decomposition of $S(f,g)$ is used by Corless *et. al.* [3], and Zarowski *et. al.* [18], and the singular value decomposition of $S(f,g)$ is used in [2]. The QR and singular value decompositions do not retain the structure of $S(f,g)$, and they must therefore be compared with methods that preserve the structure of $S(f,g)$, which are discussed in [8,10,15,16]. Other methods have also been used to calculate an AGCD of two polynomials, for example, optimisation techniques are used by Karmarkar and Lakshman [9],

and Padé approximations are used by Pan [11].

Many methods for the calculation of an AGCD of two inexact polynomials involve two stages. In particular, the degree of an AGCD of the polynomials is determined initially, after which the coefficients of the AGCD are calculated. The computation of the degree of an AGCD of $f(y)$ and $g(y)$ is equivalent to the determination of the rank loss of a resultant matrix, and methods for this computation are considered in [17]. It is assumed in this paper, however, that the degree of an AGCD is known. This assumption is also made in [8,10,15,16], and a linear structure preserving method is used in these references to compute a structured low rank approximation of $S(f, g)$.

If the ratio of the maximum coefficient (in magnitude) to the minimum coefficient (in magnitude) of $\{f(y), g(y)\}$ is large, the polynomials must be processed before a structured low rank approximation $S(\tilde{f}, \tilde{g})$ of $S(f, g)$ is computed. These preprocessing operations introduce two parameters, which can either be held constant, or incremented, during the computation of $S(\tilde{f}, \tilde{g})$. A linear structure preserving matrix method is used if they are held constant, but a non-linear structure preserving matrix method is required if they are incremented. Considerably improved results are obtained when the preprocessing operations are included in the computation of $S(\tilde{f}, \tilde{g})$, and the non-linear method yields better results than the linear method because the numerical rank of $S(\tilde{f}, \tilde{g})$ is, in general, more clearly defined. Furthermore, it is shown that the assignment of the polynomials to $f(y)$ and $g(y)$ is important because the numerical rank of a structured low rank approximation of $S(f, g)$ may be defined, but the numerical rank of a structured low rank approximation of $S(g, f)$ may not be defined.

Subresultant matrices, which are derived from $S(f, g)$ and are important for the calculation of $S(\tilde{f}, \tilde{g})$, are discussed in Section 2, and the preprocessing operations on $f(y)$ and $g(y)$ are considered in Section 3. Section 4 contains a brief comparison of STLN and SNTLN, and the application of SNTLN to the computation of $S(\tilde{f}, \tilde{g})$ is discussed in Section 5. Section 6 contains examples that show the differences in the results using STLN and SNTLN, and the importance of the polynomial order, $(f, g)$ or $(g, f)$, for the computation of a structured low rank approximation of the Sylvester matrix of $f(y)$ and $g(y)$. A summary of the paper is contained in Section 7.

## 2 Subresultant matrices

This section discusses subresultant matrices, which are derived from $S(f, g)$ by deleting some of its rows and columns. These matrices are required for the calculation of $S(\tilde{f}, \tilde{g})$, and they are most easily introduced by expressing the

product of two polynomials as a matrix-vector product.

If $\hat{f}(y)$ and $\hat{g}(y)$ are the theoretically exact forms of $f(y)$ and $g(y)$ respectively, and the degree of their greatest common divisor (GCD) is $\hat{d}$, then there exist quotient polynomials $u_k(y)$ and $v_k(y)$, and a common divisor polynomial $d_k(y)$, such that for $k = 1, \ldots, \hat{d}$,

$$d_k(y) = \frac{\hat{f}(y)}{u_k(y)} = \frac{\hat{g}(y)}{v_k(y)}, \quad \deg v_k < \deg \hat{g} = n, \quad \deg u_k < \deg \hat{f} = m, \quad (3)$$

where

$$u_k(y) = \sum_{i=0}^{m-k} u_{k,i} y^{m-k-i} \qquad \text{and} \qquad v_k(y) = \sum_{i=0}^{n-k} v_{k,i} y^{n-k-i}.$$

It follows from (3) that there exists a non-zero polynomial $t_k(y)$ such that

$$t_k(y) = v_k(y)\hat{f}(y) = u_k(y)\hat{g}(y), \qquad k = 1, \ldots, \hat{d},$$

and if $\mathbf{t}_k \in \mathbb{R}^{m+n-k+1}$ is the vector of coefficients of $t_k(y)$, then

$$\mathbf{t}_k = C_k(\hat{f})\mathbf{v}_k = D_k(\hat{g})\mathbf{u}_k, \quad (4)$$

where $C_k(\hat{f}) \in \mathbb{R}^{(m+n-k+1)\times(n-k+1)}$, $D_k(\hat{g}) \in \mathbb{R}^{(m+n-k+1)\times(m-k+1)}$, and $\mathbf{u}_k$ and $\mathbf{v}_k$ are the vectors of coefficients of $u_k(y)$ and $v_k(y)$ respectively. It follows from (4) that

$$\begin{bmatrix} C_k & D_k \end{bmatrix} \begin{bmatrix} \mathbf{v}_k \\ -\mathbf{u}_k \end{bmatrix} = S_k \begin{bmatrix} \mathbf{v}_k \\ -\mathbf{u}_k \end{bmatrix} = 0, \qquad k = 1, \ldots, \hat{d}, \quad (5)$$

where $C_k = C_k(\hat{f})$, $D_k = D_k(\hat{g})$, $S_k = S_k(\hat{f}, \hat{g}) \in \mathbb{R}^{(m+n-k+1)\times(m+n-2k+2)}$ and $S_1(\hat{f}, \hat{g}) = S(\hat{f}, \hat{g})$. The matrix $S_k(\hat{f}, \hat{g})$ is the $k$th subresultant matrix, which is formed by deleting the last $(k-1)$ rows of $S(\hat{f}, \hat{g})$, the last $(k-1)$ columns of $C_1(\hat{f})$, and the last $(k-1)$ columns of $D_1(\hat{g})$.

The polynomials $\hat{f}(y)$ and $\hat{g}(y)$ have common divisors of degrees $1, 2, \ldots, \hat{d}$, because the degree of their GCD is $\hat{d}$, but they do not have a common divisor of degree $\hat{d}+1$, and thus

$$\begin{aligned} &\operatorname{rank} S_k(\hat{f}, \hat{g}) < m + n - 2k + 2, \qquad k = 1, \ldots, \hat{d} \\ &\operatorname{rank} S_k(\hat{f}, \hat{g}) = m + n - 2k + 2, \qquad k = \hat{d}+1, \ldots, \min(m, n). \end{aligned}$$

It follows that (5) can be transformed, for $k = 1, \ldots, \hat{d}$, from a homogeneous equation to a linear algebraic equation by setting $v_{k,0} = -1$, that is, the coefficient of $y^{n-k}$ is set equal to $-1$. Equation (5) therefore becomes

$$A_k x = c_k, \qquad k = 1, \ldots, \hat{d}, \tag{6}$$

where $c_k \in \mathbb{R}^{m+n-k+1}$ is the first column of $S_k$, $A_k \in \mathbb{R}^{(m+n-k+1)\times(m+n-2k+1)}$ is formed from the remaining $m + n - 2k + 1$ columns of $S_k$,

$$S_k = \left[ c_k \,\middle|\, A_k \right], \tag{7}$$

and

$$x = \begin{bmatrix} v_{k,1} & \cdots & v_{k,n-k} & -u_{k,0} & \cdots & -u_{k,m-k} \end{bmatrix}^T \in \mathbb{R}^{m+n-2k+1}.$$

Equation (6) has an infinite number of solutions for $k = 1, \ldots, \hat{d} - 1$, exactly one solution for $k = \hat{d}$, and no solution for $k = \hat{d} + 1, \ldots, \min(m, n)$. Also, the homogeneous equation (5) is transformed to the linear algebraic equation (6) by the substitution $v_{k,0} = -1$, but it is easily seen that $u_k(y)$ and $v_k(y)$ are unchanged, apart from a scalar multiplier applied to each of them, had the substitution $u_{k,0} = 1$ been made. This equivalence between the two substitutions is valid because the given polynomials $\hat{f}(y)$ and $\hat{g}(y)$ are exact and all computations are performed symbolically. It will be shown in Section 5.1, however, that if inexact polynomials are specified, only an AGCD can be computed and the choice of substitution, $v_{k,0} = -1$ or $u_{k,0} = 1$, is important when a structured low rank approximation of the Sylvester matrix of the inexact polynomials $f(y)$ and $g(y)$ is computed.

## 3 Preprocessing operations

This section considers three preprocessing operations that are required for the computation of a structured low rank approximation of $S(f, g)$. These operations are the normalisation of each polynomial by the geometric mean of its coefficients, the weighting of $g(y)$ by a parameter $\alpha$, and a parameter substitution, and they are considered in Sections 3.1, 3.2 and 3.3 respectively.

### 3.1 Normalisation by the geometric mean

The Sylvester matrix $S(f,g)$ of $f(y)$ and $g(y)$ is shown in (2), and its partitioned structure is immediately apparent. If $f(y)$ and $g(y)$ are not normalised, then $S(f,g)$ may be unbalanced if, for example, the coefficients of $f(y)$ are significantly larger than the coefficients of $g(y)$, in which case computational problems may occur. This problem can be overcome by normalising each polynomial, and normalisation by the 2-norm of the coefficients is used in [1] and [3]. In this paper, normalisation by the geometric mean of the coefficients is used because it provides a 'better average' when the coefficients of the polynomials vary over several orders of magnitude. The polynomials (1) are therefore redefined as

$$f(y) = \sum_{i=0}^{m} \tilde{a}_i y^{m-i}, \qquad \tilde{a}_i = \frac{a_i}{\left(\prod_{j=0}^{m} |a_j|\right)^{\frac{1}{m+1}}}, \tag{8}$$

and

$$g(y) = \sum_{i=0}^{n} \tilde{b}_i y^{n-i}, \qquad \tilde{b}_i = \frac{b_i}{\left(\prod_{j=0}^{n} |b_j|\right)^{\frac{1}{n+1}}}, \tag{9}$$

where it is assumed that all the coefficients $a_i$ and $b_i$ are non-zero. More generally, the geometric mean is calculated with respect to the non-zero coefficients only.

### 3.2 Relative scaling of the polynomials

It follows from (2) that

$$\operatorname{rank} S(f,g) = \operatorname{rank} S(f,\alpha g), \qquad \alpha \in \mathbb{R}\backslash 0, \tag{10}$$

which states that the GCD of two polynomials is defined up to an arbitrary scalar multiplier, $\text{GCD}(f,g) \sim \text{GCD}(f,\alpha g)$. Equation (10) is not, however, satisfied when computations are performed in a floating point environment because the numerical rank of $S(f,\alpha g)$ is a function of $\alpha$ [15,16]. It follows from (8) and (9) that the parameter $\alpha$ can be interpreted as the weight of $g(y)$ relative to the unit weight of $f(y)$, and the importance of $\alpha$ for the computation of a structured low rank approximation of $S(f,\alpha g)$ is shown in [15,16]. A method for the calculation of an optimal value of $\alpha$ was not considered in these references, and thus the second preprocessing operation involves the computation of an optimal value of $\alpha$, and this is now considered.

It is shown in [4,5] that problems can occur in algorithms for the computation of the roots of a polynomial when the coefficients of the polynomial vary widely in magnitude. It is therefore desirable to minimise the ratio of the maximum coefficient (in magnitude) to the minimum coefficient (in magnitude), and since the coefficients of $f(y)$ and $\alpha g(y)$, that is, the arguments of $S(f, \alpha g)$, are $\tilde{a}_i$ and $\alpha\tilde{b}_i$ respectively, an optimal value $\alpha$ minimises the ratio

$$\frac{\max\left\{ \max_{i=0,\ldots,m} |\tilde{a}_i| , \max_{j=0,\ldots,n} \left|\alpha\tilde{b}_j\right| \right\}}{\min\left\{ \min_{i=0,\ldots,m} |\tilde{a}_i| , \min_{j=0,\ldots,n} \left|\alpha\tilde{b}_j\right| \right\}}. \tag{11}$$

This minimisation problem can be written as:

Minimise $\frac{t}{s}$

Subject to

$$\begin{aligned}
&t \geq |\tilde{a}_i| , \quad i = 0, \ldots, m \\
&t \geq \alpha \left|\tilde{b}_j\right| , \; j = 0, \ldots, n \\
&s \leq |\tilde{a}_i| , \quad i = 0, \ldots, m \\
&s \leq \alpha \left|\tilde{b}_j\right| , \; j = 0, \ldots, n \\
&s > 0 \\
&\alpha > 0.
\end{aligned}$$

The transformations

$$T = \log t, \quad S = \log s, \quad \mu = \log \alpha, \quad \tilde{\alpha}_i = \log |\tilde{a}_i| \quad \text{and} \quad \tilde{\beta}_j = \log \left|\tilde{b}_j\right| ,$$

enable this constrained minimisation problem to be written as:

Minimise $T - S$

Subject to

$$\begin{aligned}
T \qquad &\geq \quad \tilde{\alpha}_i, \;\; i = 0, \ldots, m \\
T - \mu &\geq \quad \tilde{\beta}_j, \;\; j = 0, \ldots, n \\
-S \qquad &\geq -\tilde{\alpha}_i, \;\; i = 0, \ldots, m \\
-S + \mu &\geq -\tilde{\beta}_j, \;\; j = 0, \ldots, n,
\end{aligned} \tag{12}$$

which is a linear programming problem, where the objective function is

$$T - S = \begin{bmatrix} 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} T \\ S \\ \mu \end{bmatrix}.$$

There are $2(m+n+2)$ constraints in the linear programming problem (12), and if a coefficient $a_i$ or $b_j$ is equal to zero, then the corresponding constraints are deleted. The solution $\alpha_0$ of (12) is the optimal value of $\alpha$.

The parameter $\alpha$ scales the coefficients of $g(y)$ relative to the coefficients of $f(y)$, and it is shown in the next section that the ratio of coefficients (11) can be reduced further by scaling the independent variable $y$.

### *3.3 Scaling the independent variable*

The ratio of the maximum coefficient (in magnitude) to the minimum coefficient (in magnitude) of the polynomials $\{f(y), \alpha_0 g(y)\}$ can be reduced further by the substitution

$$y = \theta w, \tag{13}$$

where $w$ is the new independent variable and $\theta$ is a real constant to be determined. This substitution is justified provided it does not increase the condition numbers of the roots of an arbitrary polynomial, and it is shown in [17] that this requirement is satisfied. The substitution (13) transforms the polynomials $f(y)$ and $g(y)$, which are defined in (8) and (9) respectively, to

$$f_\theta(w) = \sum_{i=0}^{m} \left(\tilde{a}_i \theta^{m-i}\right) w^{m-i} \quad \text{and} \quad g_\theta(w) = \sum_{i=0}^{n} \left(\tilde{b}_i \theta^{n-i}\right) w^{n-i}, \tag{14}$$

and thus following (11), the optimal value $\theta_0$ of $\theta$ is the value of $\theta$ that minimises the ratio

$$\frac{\max\left\{\max_{i=0,\ldots,m} \left|\tilde{a}_i \theta^{m-i}\right|, \max_{j=0,\ldots,n} \left|\alpha_0 \tilde{b}_j \theta^{n-j}\right|\right\}}{\min\left\{\min_{i=0,\ldots,m} \left|\tilde{a}_i \theta^{m-i}\right|, \min_{j=0,\ldots,n} \left|\alpha_0 \tilde{b}_j \theta^{n-j}\right|\right\}}. \tag{15}$$

This minimisation problem can, like the minimisation problem (11), be solved by methods used in linear programming. In particular, it can be written as:

Minimise $\frac{t}{s}$

Subject to

$$
\begin{aligned}
t &\geq |\tilde{a}_i|\, \theta^{m-i}, \quad && i = 0, \ldots, m \\
t &\geq \left|\alpha_0 \tilde{b}_j\right| \theta^{n-j}, && j = 0, \ldots, n \\
s &\leq |\tilde{a}_i|\, \theta^{m-i}, \quad && i = 0, \ldots, m \\
s &\leq \left|\alpha_0 \tilde{b}_j\right| \theta^{n-j}, && j = 0, \ldots, n \\
s &> 0 \\
\theta &> 0.
\end{aligned}
$$

The transformations

$$T = \log t, \quad S = \log s, \quad \phi = \log \theta, \quad \tilde{\alpha}_i = \log |\tilde{a}_i| \quad \text{and} \quad \tilde{\beta}_j = \log \left|\alpha_0 \tilde{b}_j\right|,$$

enable this constrained minimisation problem to be written as:

Minimise $T - S$

Subject to

$$
\begin{aligned}
T - (m-i)\phi &\geq \quad \tilde{\alpha}_i, \quad i = 0, \ldots, m \\
T - (n-j)\phi &\geq \quad \tilde{\beta}_j, \quad j = 0, \ldots, n \\
-S + (m-i)\phi &\geq -\tilde{\alpha}_i, \quad i = 0, \ldots, m \\
-S + (n-j)\phi &\geq -\tilde{\beta}_j, \quad j = 0, \ldots, n,
\end{aligned}
\tag{16}
$$

which is almost identical to the linear programming problem (12).

The minimisations (11) and (15) transform the polynomials (14) to

$$f_{\theta_0}(w) = \sum_{i=0}^{m} \left(\tilde{a}_i \theta_0^{m-i}\right) w^{m-i} \qquad \text{and} \qquad g_{\theta_0}(w) = \sum_{i=0}^{n} \left(\tilde{b}_i \theta_0^{n-i}\right) w^{n-i},$$

where $\theta_0$ is the solution of (16). The coefficients of $f_{\theta_0}(w)$ and $g_{\theta_0}(w)$ define the entries of the Sylvester matrix on which all computations are performed, and it was noted in Section 3.1 that it is advantageous to normalise these polynomials by the geometric mean of their coefficients. This yields the polynomials

$$\bar{f}(w) = \sum_{i=0}^{m} \left( a_i^* \theta_0^{m-i} \right) w^{m-i} \qquad \text{and} \qquad \bar{g}(w) = \sum_{i=0}^{n} \left( b_i^* \theta_0^{n-i} \right) w^{n-i}, \tag{17}$$

where

$$a_i^* = \frac{\tilde{a}_i}{\left( \prod_{j=0}^{m} \left| \tilde{a}_j \theta_0^{m-j} \right| \right)^{\frac{1}{m+1}}} \qquad \text{and} \qquad b_i^* = \frac{\tilde{b}_i}{\left( \prod_{j=0}^{n} \left| \tilde{b}_j \theta_0^{n-j} \right| \right)^{\frac{1}{n+1}}},$$

and $\tilde{a}_i$ and $\tilde{b}_i$ are defined in (8) and (9) respectively. The multiplicities of the roots of $f(y)$ and $g(y)$ are preserved by the transformation (13), and thus $S(\bar{f}, \bar{g})$ can be used to calculate an AGCD of $f(y)$ and $g(y)$. The roots of $f(y)$ and $g(y)$ are not, however, equal to the roots of $\bar{f}(w)$ and $\bar{g}(w)$, respectively, if $\theta_0 \neq 1$.

Algorithm 3.1 shows the operations that are performed on the given inexact polynomials (1) before a structured low rank approximation of the Sylvester matrix $S(\bar{f}, \bar{g})$ is computed.

---

**Algorithm 3.1: Preprocessing operations**

**Input** Inexact polynomials $f(y)$ and $g(y)$, which are defined in (1).

**Output** Polynomials $\bar{f}(w)$ and $\bar{g}(w)$, which are defined in (17).

**Begin**

(1) Normalise the coefficients of $f(y)$ and $g(y)$ by the geometric mean of their coefficients, as shown in (8) and (9).
(2) Solve the linear programming problem (12) in order to compute $\alpha_0$.
(3) Solve the linear programming problem (16) in order to compute $\theta_0$.
(4) Calculate the coefficients $a_i^* \theta_0^{m-i}$ and $b_i^* \theta_0^{n-i}$ of $\bar{f}(w)$ and $\bar{g}(w)$, respectively.

**End**

---

## 4 Structured matrix methods

It is assumed that $f(y)$ and $g(y)$, and therefore $\bar{f}(w)$ and $\bar{g}(w)$, are inexact and coprime, and thus $S(\bar{f}, \bar{g})$ has full rank. The computation of a structured

low rank approximation of $S(\bar{f}, \bar{g})$ requires the determination of a Sylvester matrix $S(\delta\bar{f}, \delta\bar{g})$ such that

$$S(\bar{f} + \delta\bar{f}, \bar{g} + \delta\bar{g}) = S(\bar{f}, \bar{g}) + S(\delta\bar{f}, \delta\bar{g}),$$

is rank deficient, where $\delta\bar{f} = \delta\bar{f}(w)$ and $\delta\bar{g} = \delta\bar{g}(w)$ are perturbation polynomials that are added to $\bar{f}(w)$ and $\bar{g}(w)$, respectively, in order to induce rank deficiency in $S(\bar{f}+\delta\bar{f}, \bar{g}+\delta\bar{g})$. The structured nature of the Sylvester matrix implies that structured matrix methods can be used to compute $S(\bar{f}+\delta\bar{f}, \bar{g}+\delta\bar{g})$, and this computation can be achieved by STLN, which preserves the affine structure of $S(\bar{f}, \bar{g})$, or SNTLN, which preserves the structure of $S(\bar{f}, \bar{g})$ when its elements are differentiable non-linear functions of one or more parameters. These methods are considered in Sections 4.1 and 4.2 respectively.

### *4.1 Linear structure preserving matrix method*

The method of STLN assumes that $\alpha_0$ and $\theta_0$ are constant, and thus they are not updated in the iterative scheme for the computation of the coefficients of $\delta\bar{f}(w)$ and $\delta\bar{g}(w)$. The polynomials (17) are therefore written as

$$\bar{f}(w) = \sum_{i=0}^{m} \bar{a}_i w^{m-i} \qquad \text{and} \qquad \bar{g}(w) = \sum_{i=0}^{n} \bar{b}_i w^{n-i}, \tag{18}$$

whose coefficients are

$$\bar{a}_i = a_i^* \theta_0^{m-i} = \frac{\tilde{a}_i \theta_0^{m-i}}{\left(\prod_{j=0}^{m} \left|\tilde{a}_j \theta_0^{m-j}\right|\right)^{\frac{1}{m+1}}}, \tag{19}$$

and

$$\bar{b}_i = b_i^* \theta_0^{n-i} = \frac{\tilde{b}_i \theta_0^{n-i}}{\left(\prod_{j=0}^{n} \left|\tilde{b}_j \theta_0^{n-j}\right|\right)^{\frac{1}{n+1}}}. \tag{20}$$

The method of STLN allows a structured low rank approximation of $S(\bar{f}, \alpha_0\bar{g})$ to be computed, where $\bar{f}(w)$ and $\bar{g}(w)$, and their coefficients, are defined in (18) and (19,20), respectively, and only these coefficients are updated in the iterative scheme for the computation of the coefficients of $\delta\bar{f}(w)$ and $\delta\bar{g}(w)$. In particular, $\alpha_0$ and $\theta_0$ are constant, and only the coefficients $\bar{a}_i$ and $\bar{b}_i$ are updated, which implies that a linear structure preserving matrix method can be used.

This method is more complex than the linear structure preserving matrix method because more parameters are updated in the iterative scheme for the computation of the coefficients of $\delta\bar{f}(w)$ and $\delta\bar{g}(w)$. In particular, the initial values of $\alpha$ and $\theta$ in this scheme are $\alpha_0$ and $\theta_0$, that is, the solutions of the linear programming problems (12) and (16) respectively, and the polynomials (17) are written as

$$\bar{f}(w) \approx \sum_{i=0}^{m} \left(\bar{a}_i \theta^{m-i}\right) w^{m-i} \qquad \text{and} \qquad \bar{g}(w) \approx \sum_{i=0}^{n} \left(\bar{b}_i \theta^{n-i}\right) w^{n-i}, \tag{21}$$

where

$$\bar{a}_i = a_i^* = \frac{\tilde{a}_i}{\left(\prod_{j=0}^{m} \left|\tilde{a}_j \theta_0^{m-j}\right|\right)^{\frac{1}{m+1}}}, \tag{22}$$

and

$$\bar{b}_i = b_i^* = \frac{\tilde{b}_i}{\left(\prod_{j=0}^{n} \left|\tilde{b}_j \theta_0^{n-j}\right|\right)^{\frac{1}{n+1}}}. \tag{23}$$

The constant $\theta_0$ is retained in the denominators of these expressions for $\bar{a}_i$ and $\bar{b}_i$ because it simplifies the update procedure for $\theta$ between successive iterations.

The differences between the polynomials (18) and (21) are important:

- Only the coefficients $\bar{a}_i$ and $\bar{b}_i$, which are defined in (19) and (20) respectively, are updated when STLN is used.
- The coefficients $\bar{a}_i\theta^{m-i}$ and $\bar{b}_i\theta^{n-i}$, where $\bar{a}_i$ and $\bar{b}_i$ are defined in (22) and (23) respectively, and $\alpha$ are updated when SNTLN is used.

The next section considers the method of SNTLN for the calculation of a structured low rank approximation of $S(\bar{f}, \alpha\bar{g})$.

## 5 The method of SNTLN

This section describes the method of SNTLN for the determination of a structured low rank approximation of $S(\bar{f}, \alpha\bar{g})$, where $\bar{f}(w)$ and $\bar{g}(w)$ are defined

in (21) and the inclusion of $\alpha$ follows from (10). The Sylvester matrix $S(\bar{f}, \alpha\bar{g})$ of $\bar{f}(w)$ and $\alpha\bar{g}(w)$ is

$$\begin{bmatrix}
\bar{a}_0\theta^m & & & \alpha\bar{b}_0\theta^n & & \\
\bar{a}_1\theta^{m-1} & \ddots & & \alpha\bar{b}_1\theta^{n-1} & \ddots & \\
\vdots & \ddots & \bar{a}_0\theta^m & \vdots & \ddots & \alpha\bar{b}_0\theta^n \\
\bar{a}_{m-1}\theta & \ddots & \bar{a}_1\theta^{m-1} & \alpha\bar{b}_{n-1}\theta & \ddots & \alpha\bar{b}_1\theta^{n-1} \\
\bar{a}_m & \ddots & \vdots & \alpha\bar{b}_n & \ddots & \vdots \\
 & \ddots & \bar{a}_{m-1}\theta & & \ddots & \alpha\bar{b}_{n-1}\theta \\
 & & \bar{a}_m & & & \alpha\bar{b}_n
\end{bmatrix},$$

where $\bar{a}_i$ and $\bar{b}_i$ are defined in (22) and (23) respectively, and the optimal values of $\alpha$ and $\theta$ are determined using an iterative scheme for which $\alpha_0$ and $\theta_0$ are the initial values. The subresultant matrix $S_k = S_k(\bar{f}, \alpha\bar{g})$ is partitioned as, following (7),

$$S_k = \left[\, c_k \,\middle|\, A_k \,\right] = \left[\, c_k \,\middle|\, \text{coeffs. of } \bar{f}(w) \,\middle|\, \text{coeffs. of } \alpha\bar{g}(w) \,\right],$$

where $c_k = c_k(\theta) \in \mathbb{R}^{m+n-k+1}$ and $A_k = A_k(\alpha, \theta) \in \mathbb{R}^{(m+n-k+1)\times(m+n-2k+1)}$.

The polynomials $\bar{f}(w)$ and $\bar{g}(w)$ are inexact and they are therefore perturbed in order to induce a non-constant common divisor in their perturbed forms. If the perturbations of the coefficients of $\bar{f}(w)$ and $\alpha\bar{g}(w)$ are, respectively,

$$z_i\theta^{m-i}, i = 0, \ldots, m \qquad \text{and} \qquad \alpha z_{m+1+i}\theta^{n-i}, i = 0, \ldots, n,$$

then the Sylvester matrix $B_k = B_k(\alpha, \theta, z) \in \mathbb{R}^{(m+n-k+1)\times(m+n-2k+2)}$ of the perturbations is

$$B_k = \left[\, h_k \,\middle|\, E_k \,\right] = \begin{bmatrix}
z_0\theta^m & & & \alpha z_{m+1}\theta^n & & \\
z_1\theta^{m-1} & \ddots & & \alpha z_{m+2}\theta^{n-1} & \ddots & \\
\vdots & \ddots & z_0\theta^m & \vdots & \ddots & \alpha z_{m+1}\theta^n \\
z_{m-1}\theta & \ddots & z_1\theta^{m-1} & \alpha z_{m+n}\theta & \ddots & \alpha z_{m+2}\theta^{n-1} \\
z_m & \ddots & \vdots & \alpha z_{m+n+1} & \ddots & \vdots \\
 & \ddots & z_{m-1}\theta & & \ddots & \alpha z_{m+n}\theta \\
 & & z_m & & & \alpha z_{m+n+1}
\end{bmatrix},$$

where $h_k = h_k(\theta, z) \in \mathbb{R}^{m+n-k+1}$ is the first column of $B_k$,

$$z = \begin{bmatrix} z_0 \; z_1 \; \cdots \; z_{m+n} \; z_{m+n+1} \end{bmatrix}^T \in \mathbb{R}^{m+n+2},$$

and $E_k = E_k(\alpha, \theta, z) \in \mathbb{R}^{(m+n-k+1)\times(m+n-2k+1)}$. The application of SNTLN to the computation of an AGCD of $\bar{f}(w)$ and $\bar{g}(w)$ requires that the equation

$$\left(A_k(\alpha, \theta) + E_k(\alpha, \theta, z)\right) x = c_k(\theta) + h_k(\theta, z), \qquad x \in \mathbb{R}^{m+n-2k+1},$$

which is the perturbed form of (6), be considered. The residual that is associated with an approximate solution of this non-linear equation is

$$r(\alpha, \theta, x, z) = c_k(\theta) + h_k(\theta, z) - \left(A_k(\alpha, \theta) + E_k(\alpha, \theta, z)\right) x, \tag{24}$$

and thus if $\tilde{r}$ is defined as

$$\tilde{r} := r(\alpha + \delta\alpha, \theta + \delta\theta, x + \delta x, z + \delta z),$$

then

$$\begin{aligned}
\tilde{r} =& c_k(\theta + \delta\theta) + h_k(\theta + \delta\theta, z + \delta z) \\
&- \Big(A_k(\alpha + \delta\alpha, \theta + \delta\theta) + E_k(\alpha + \delta\alpha, \theta + \delta\theta, z + \delta z)\Big)(x + \delta x) \\
=& c_k + \frac{\partial c_k}{\partial \theta}\delta\theta + h_k + \frac{\partial h_k}{\partial \theta}\delta\theta + \sum_{i=0}^{m+n+1} \frac{\partial h_k}{\partial z_i}\delta z_i - A_k x - A_k \delta x \\
&- \left(\frac{\partial A_k}{\partial \alpha} x\right)\delta\alpha - \left(\frac{\partial A_k}{\partial \theta} x\right)\delta\theta - E_k x - E_k \delta x - \left(\frac{\partial E_k}{\partial \alpha} x\right)\delta\alpha \\
&- \left(\frac{\partial E_k}{\partial \theta} x\right)\delta\theta - \left(\sum_{i=0}^{m+n+1} \frac{\partial E_k}{\partial z_i}\delta z_i\right) x,
\end{aligned}$$

to first order. It follows that

$$\begin{aligned}
\tilde{r} =& r(\alpha, \theta, x, z) - \left(\left(\frac{\partial A_k}{\partial \theta} + \frac{\partial E_k}{\partial \theta}\right) x - \left(\frac{\partial c_k}{\partial \theta} + \frac{\partial h_k}{\partial \theta}\right)\right)\delta\theta \\
&- (A_k + E_k)\delta x - \left(\left(\frac{\partial A_k}{\partial \alpha} + \frac{\partial E_k}{\partial \alpha}\right) x\right)\delta\alpha + \sum_{i=0}^{m+n+1} \frac{\partial h_k}{\partial z_i}\delta z_i \\
&- \sum_{i=0}^{m+n+1} \left(\frac{\partial E_k}{\partial z_i}\delta z_i\right) x,
\end{aligned} \tag{25}$$

where expressions for the partial derivatives are easily calculated from $c_k, h_k, A_k$ and $E_k$.

It is readily verified that

$$h_k = P_k z = \begin{bmatrix} G & 0_{m+1,n+1} \\ 0_{n-k,m+1} & 0_{n-k,n+1} \end{bmatrix} z,$$

where $P_k = P_k(\theta) \in \mathbb{R}^{(m+n-k+1)\times(m+n+2)}$,

$$G = G(\theta) = \mathrm{diag}\begin{bmatrix} \theta^m & \theta^{m-1} & \cdots & \theta & 1 \end{bmatrix} \in \mathbb{R}^{(m+1)\times(m+1)},$$

and

$$\sum_{i=0}^{m+n+1} \frac{\partial h_k}{\partial z_i} \delta z_i = P_k \delta z.$$

Also, there exists a matrix $Y_k = Y_k(\alpha, \theta, x) \in \mathbb{R}^{(m+n-k+1)\times(m+n+2)}$ such that

$$Y_k z = E_k x,$$

for all $z, x, \alpha, \theta$, and it therefore follows that on differentiating both sides of this equation with respect to $z$,

$$Y_k \delta z = \left(\delta E_k \mid_{\alpha,\theta:\mathrm{const.}}\right) x = \sum_{i=0}^{m+n+1} \left(\frac{\partial E_k}{\partial z_i} \delta z_i\right) x,$$

and thus (25) simplifies to

$$\begin{aligned} \tilde{r} = r(\alpha, \theta, x, z) &- \left(\left(\frac{\partial A_k}{\partial \theta} + \frac{\partial E_k}{\partial \theta}\right) x - \left(\frac{\partial c_k}{\partial \theta} + \frac{\partial h_k}{\partial \theta}\right)\right) \delta\theta \\ &-(A_k + E_k)\delta x - \left(\left(\frac{\partial A_k}{\partial \alpha} + \frac{\partial E_k}{\partial \alpha}\right) x\right) \delta\alpha - (Y_k - P_k)\delta z. \end{aligned} \tag{26}$$

The $j$th iteration in the Newton-Raphson method for the calculation of $z, x, \alpha, \theta$, is obtained from (26),

$$\begin{bmatrix} H_z & H_x & H_\alpha & H_\theta \end{bmatrix}^{(j)} \begin{bmatrix} \delta z \\ \delta x \\ \delta\alpha \\ \delta\theta \end{bmatrix}^{(j)} = r^{(j)}, \tag{27}$$

where $r^{(j)} = r^{(j)}(\alpha, \theta, x, z)$,

$$H_z = Y_k - P_k, \qquad H_x = A_k + E_k,$$

$$H_\alpha = \left(\frac{\partial A_k}{\partial \alpha} + \frac{\partial E_k}{\partial \alpha}\right) x, \quad H_\theta = \left(\frac{\partial A_k}{\partial \theta} + \frac{\partial E_k}{\partial \theta}\right) x - \left(\frac{\partial c_k}{\partial \theta} + \frac{\partial h_k}{\partial \theta}\right),$$

and the values of $z, x, \alpha, \theta$ at the $(j+1)$th iteration are

$$\begin{bmatrix} z \\ x \\ \alpha \\ \theta \end{bmatrix}^{(j+1)} = \begin{bmatrix} z \\ x \\ \alpha \\ \theta \end{bmatrix}^{(j)} + \begin{bmatrix} \delta z \\ \delta x \\ \delta\alpha \\ \delta\theta \end{bmatrix}^{(j)}.$$

The initial value of $z$ is $z^{(0)} = 0$ because the given data is the inexact data, and the initial values of $\alpha$ and $\theta$ are $\alpha_0$ and $\theta_0$, which are the solutions of (12) and (16), respectively.

Equation (27) is of the form

$$Cy = q, \tag{28}$$

where $C \in \mathbb{R}^{(m+n-k+1)\times(2m+2n-2k+5)}$, $y \in \mathbb{R}^{2m+2n-2k+5}$, $q \in \mathbb{R}^{m+n-k+1}$,

$$C = \begin{bmatrix} H_z & H_x & H_\alpha & H_\theta \end{bmatrix}^{(j)}, \qquad y = \begin{bmatrix} \delta z \\ \delta x \\ \delta\alpha \\ \delta\theta \end{bmatrix}^{(j)}, \qquad q = r^{(j)}. \tag{29}$$

It is necessary to calculate the smallest perturbations $z_i$ such that the perturbed polynomials have a non-constant common divisor. Since each of the

perturbations $z_i, i = 0, \ldots, m$, occurs $(n-k+1)$ times in $B_k$, and each of the perturbations $z_i, i = m+1, \ldots, m+n+1$, occurs $(m-k+1)$ times in $B_k$, it follows that the weight matrix $D \in \mathbb{R}^{(m+n+2)\times(m+n+2)}$ associated with $z$ is

$$D = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix},$$

where $D_1 \in \mathbb{R}^{(m+1)\times(m+1)}$ and $D_2 \in \mathbb{R}^{(n+1)\times(n+1)}$ are diagonal matrices,

$$D_1 = (n-k+1)I_{m+1} \qquad \text{and} \qquad D_2 = (m-k+1)I_{n+1}.$$

Also, $\alpha$ occurs $d = (n+1)\times(m-k+1)$ times in $B_k$, and thus it is necessary to minimise the function

$$\left\| \begin{bmatrix} D\left(z^{(j)} + \delta z^{(j)} - z^{(0)}\right) \\ d\left(\alpha^{(j)} + \delta\alpha^{(j)} - \alpha_0\right) \\ \theta^{(j)} + \delta\theta^{(j)} - \theta_0 \end{bmatrix} \right\| = \left\| \begin{bmatrix} D\left(z^{(j)} + \delta z^{(j)}\right) \\ d\left(\alpha^{(j)} + \delta\alpha^{(j)} - \alpha_0\right) \\ \theta^{(j)} + \delta\theta^{(j)} - \theta_0 \end{bmatrix} \right\| := \|Ey - p\|, \tag{30}$$

subject to (28), at each iteration, where $E \in \mathbb{R}^{(m+n+4)\times(2m+2n-2k+5)}$ and $p \in \mathbb{R}^{m+n+4}$ are given by

$$E = \begin{bmatrix} D & 0 & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad p = \begin{bmatrix} -Dz \\ d\left(\alpha_0 - \alpha\right) \\ \theta_0 - \theta \end{bmatrix}^{(j)},$$

and $y$ is defined in (29). It is noted that $E$ is constant and not updated between iterations.

The minimisation of (30) subject to (28) is a least squares minimisation with an equality constraint (the LSE problem),

$$\min_y \|Ey - p\| \qquad \text{subject to} \qquad Cy = q, \tag{31}$$

which can be solved by the QR decomposition [6]. This LSE problem is solved at each iteration, where $C, p$ and $q$ are updated between successive iterations. The initial value $x_0$ of $x$ in the iterative procedure for the solution of this problem is obtained by setting $\theta = \theta_0, \alpha = \alpha_0$ and $z = z^{(0)} = 0$, and thus from (24),

$$x_0 = \arg\min_w \|A_k(\alpha_0, \theta_0)w - c_k(\theta_0)\|. \quad (32)$$

The given data is the inexact polynomials $f(y)$ and $g(y)$, and all computations are performed on the transformed polynomials $\bar{f}(w)$ and $\bar{g}(w)$. The computed structured low rank approximation Sylvester matrix is $S(\tilde{f}, \tilde{g})$, where $\tilde{f}(w)$ and $\tilde{g}(w)$ can be transformed back to their equivalents in the independent variable $y$ by the substitution $w = y/\theta^*$, and $\theta^*$ is the value of $\theta$ at the termination of the iterative scheme for the solution of the LSE problem.

The convergence of the algorithm for the solution of the LSE problem has not been established, and the success or failure of the algorithm to compute $S(\tilde{f}, \tilde{g})$ is determined by an *a posteriori* test on the computed result. Specifically, the Sylvester matrix $S(\tilde{f}, \tilde{g})$ of the computed polynomials $\tilde{f}(w)$ and $\tilde{g}(w)$ is constructed in order to determine if it is, or is not, rank deficient.

Algorithm 5.1 shows the application of SNTLN for the calculation of a structured low rank approximation of $S(f, g)$.

---

**Algorithm 5.1: SNTLN for a Sylvester matrix**

**Input** Inexact polynomials $f(y)$ and $g(y)$, which are of degrees $m$ and $n$ respectively and defined in (1), and the degree $\hat{d}$ of the GCD of the exact forms of $f(y)$ and $g(y)$.

**Output** A structured low rank approximation of $S(f, g)$ of rank $m + n - \hat{d}$.

**Begin**

(1) Preprocess $f(y)$ and $g(y)$ using Algorithm 3.1.
(2) Set $k = \hat{d}$.
(3) % Initialise the data
- Set $z = z^{(0)} = 0$, which yields $E_k = \frac{\partial E_k}{\partial \alpha} = \frac{\partial E_k}{\partial \theta} = 0$ and $h_k = \frac{\partial h_k}{\partial \theta} = 0$.
- Calculate $A_k, Y_k, P_k, c_k, \frac{\partial A_k}{\partial \alpha}, \frac{\partial A_k}{\partial \theta}$ and $\frac{\partial c_k}{\partial \theta}$ for $\theta = \theta_0, \alpha = \alpha_0$ and the initial value $x_0$ of $x$, which is defined in (32). Calculate the initial value of $q$, which is equal to the residual,

$$r(\alpha_0, \theta_0, x_0, z^{(0)} = 0) = c_k - A_k x_0,$$

  and set the initial value of $p$, $p = 0$.
- Define the matrices $C$ and $E$.

(4) % The loop for the iterations
% Use the QR decomposition to solve the LSE problem at each iteration

**repeat**

**(a)** Compute the QR decomposition of $C^T$,

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

**(b)** Set $w_1 = R_1^{-T} q$.

**(c)** Partition $EQ$ as

$$EQ = \begin{bmatrix} E_1 & E_2 \end{bmatrix},$$

where $E_1 \in \mathbb{R}^{(m+n+4)\times(m+n-k+1)}$ and $E_2 \in \mathbb{R}^{(m+n+4)\times(m+n-k+4)}$.

**(d)** Compute

$$z_1 = E_2^{\dagger} \left(p - E_1 w_1\right).$$

**(e)** Compute the solution

$$y = Q \begin{bmatrix} w_1 \\ z_1 \end{bmatrix}.$$

**(f)** Set $z := z + \delta z$, $x := x + \delta x, \alpha := \alpha + \delta\alpha$ and $\theta := \theta + \delta\theta$.

**(g)** Update $A_k$, $\frac{\partial A_k}{\partial \theta}$, $\frac{\partial A_k}{\partial \alpha}$, $E_k$, $\frac{\partial E_k}{\partial \theta}$, $\frac{\partial E_k}{\partial \alpha}$, $Y_k, P_k$, $c_k$, $\frac{\partial c_k}{\partial \theta}$, $h_k$, $\frac{\partial h_k}{\partial \theta}$ (and therefore $C$) from $\alpha, \theta, x$ and $z$. Compute the residual

$$r(\alpha, \theta, x, z) = (c_k + h_k) - (A_k + E_k)x,$$

and thus update $q$. Update $p$ from $\alpha, \theta$ and $z$.

**until** $\frac{\|r(\alpha,\theta,x,z)\|}{\|c_h+h_k\|} \leq 10^{-12}$

**End**

---

### *5.1 The definition of the polynomials*

It is assumed in Section 5 that all computations are performed on $S(f, g)$, and not on $S(g, f)$. If $f(y)$ and $g(y)$ are exact polynomials and all computations are performed symbolically, then the results obtained with $S(f, g)$ are equal to, up to a scalar multiplier, the results obtained with $S(g, f)$, as explained in Section 2.

The situation is more involved when computations are performed on inexact polynomials because (6) does not possess an exact solution in this situation. In particular, the results obtained from $S(f, g)$ are not equal to the results obtained from $S(g, f)$ because the entries of $A_k$ and $c_k$ are dependent upon the order in which the polynomials are specified, that is, the order $(f, g)$ or the order $(g, f)$. It is clear that this reversal of the order of $f(y)$ and $g(y)$ does not change the normalisation of each polynomial by the geometric mean of its coefficients, and the solutions of the linear programming problems (12) and (16) need not be recomputed for $S(g, f)$. In particular, it has been shown that $\alpha_0$ and $\theta_0$ are the optimal values of $\alpha$ and $\theta$ when the polynomial order $(f, g)$ is used. When the polynomial order $(g, f)$ is used, computations are performed on $S(\bar{g}, \alpha\bar{f})$, where $1/\alpha_0$ is the initial value of $\alpha$, and $\theta_0$ is the initial value of $\theta$, when the method of SNTLN is used.

## 6 Examples

This section contains two examples that show the differences in the results between the methods of STLN and SNTLN, the importance of the order of assignment of the polynomials to $f(y)$ and $g(y)$, and the significant reduction in the ratio of the maximum coefficient (in magnitude) to the minimum coefficient (in magnitude) when the preprocessing operations discussed in Section 3 are implemented.

It is necessary to refer to the Sylvester matrices of several pairs of polynomials when the results of the examples are considered. The following notation is therefore used in all the examples:

- $\hat{f}(y)$ and $\hat{g}(y)$ are the theoretically exact polynomials, and $S(\hat{f}, \hat{g})$ and $S(\hat{g}, \hat{f})$ are calculated by normalising each polynomial by the geometric mean of its coefficients.
- $f(y)$ and $g(y)$ are calculated from $\hat{f}(y)$ and $\hat{g}(y)$ by adding noise and normalising these inexact polynomials by the geometric mean of their coefficients.
- $\bar{f}(w)$ and $\bar{g}(w)$ are the polynomials, the coefficients of which form the entries of the Sylvester matrix whose structured low rank approximation is computed. These polynomials and their coefficients are defined in (18) and (19,20) when STLN is used, and in (21) and (22,23) when SNTLN is used.
- $\tilde{f}(w)$ and $\tilde{g}(w)$ are the polynomials that are computed by the methods of STLN and SNTLN, and thus $S(\tilde{f}, \alpha^*\tilde{g})$ and $S(\tilde{g}, \tilde{f}/\alpha^*)$ are the structured low rank approximations of the Sylvester matrix of $f(y)$ and $g(y)$. The value of $\alpha^*$ depends on whether STLN or SNTLN is used:
  - STLN: $\alpha^* = \alpha_0$.
  - SNTLN: $\alpha^*$ is equal to the value of $\alpha$ when the iterative procedure for the solution of the LSE problem (31) has converged.

Normalisation is not applied to $\tilde{f}(w)$ and $\tilde{g}(w)$.

The variation of the norm of the normalised residual $r_{\text{norm}}$,

$$r_{\text{norm}} = \frac{r(\alpha, \theta, x, z)}{\|c_k(\theta) + h_k(\theta, \alpha)\|}, \tag{33}$$

where $r(\alpha, \theta, x, z)$ is defined in (24), with the number of iterations required for the solution of the LSE problem (31) is considered in the examples.

It is assumed that the degrees of the polynomials are known, and thus the dimensions of the Sylvester matrix and its subresultant matrices are defined. Furthermore, the polynomials are defined by their roots, and the coefficients of each polynomial are obtained by the convolution of the linear factors defined by its roots.

*Example 6.1*

Consider the polynomials

$$\hat{f}(y) = (y - 10^{-5})^3(y - 3.1 \times 10^{-3})^3(y - 3.2 \times 10^{-3})^3(y - 5)^{15}, \tag{34}$$
$$\hat{g}(y) = (y - 3.1 \times 10^{-3})^4(y - 3.2 \times 10^{-3})^3(y + 3.3 \times 10^{6})^{10}, \tag{35}$$

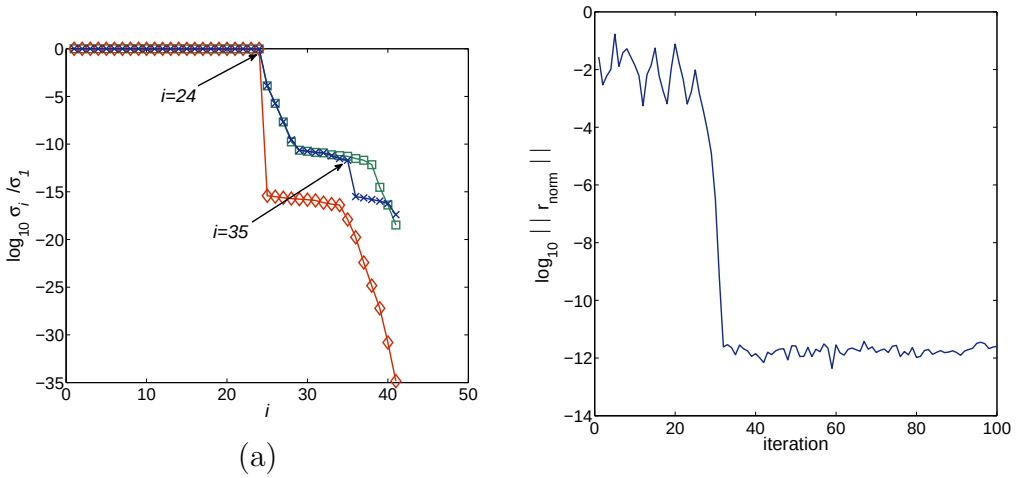
whose Sylvester matrix is of order $41 \times 41$, and since their GCD is of degree 6, it follows that rank $S(\hat{f}, \hat{g}) = 35$. Noise with a normwise signal-to-noise ratio of $10^8$ was added to the polynomials (34) and (35), which were then normalised, thereby yielding the polynomials $f(y)$ and $g(y)$.

Figure 1 shows the results obtained from STLN for $\alpha = \theta = 1$ and both these parameters are held constant, that is, the only preprocessing operation is the normalisation of each polynomial by the geometric mean of its coefficients. It is seen that

$$\text{rank } S(\hat{f}, \hat{g}) = \text{rank } S(f, g) = \text{rank } S(\tilde{f}, \tilde{g}) = 24,$$

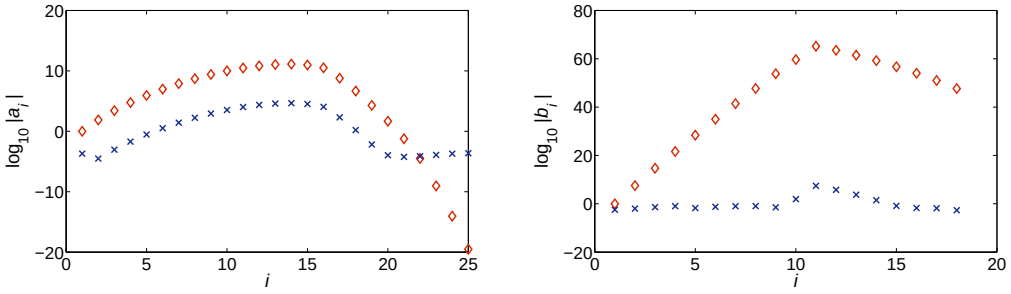
which is incorrect. Although the normalised residual (33) at convergence is about $10^{-12}$, it is seen that a small normalised residual does not imply that a correct structured low rank approximation of a Sylvester matrix has been computed.

Figure 2 shows that the preprocessing procedures considered in Section 3 cause a large reduction in the ratio of the maximum coefficient (in magnitude) to the minimum coefficient (in magnitude), particularly for $g(y)$.




(a) (b)

Fig. 1. (a) The normalised singular values of the Sylvester matrices $S(\hat{f},\hat{g})$ $\diamondsuit$; $S(f,g)$ $\square$; $S(\tilde{f},\tilde{g})$ $\times$, and (b) the normalised residual, for Example 6.1. The preprocessing operations, apart from normalising each polynomial by the geometric mean of its coefficients, are omitted.




(a) (b)

Fig. 2. The magnitude of the coefficients of (a) $f(y)$ and (b) $g(y)$ before, $\diamondsuit$, and after, $\times$, scaling by $\alpha$ and $\theta$, for Example 6.1.

The methods of STLN and SNTLN were then used to compute structured low rank approximations of $S(f,g)$. Figure 3 shows the results when STLN is applied and it is seen that the computed numerical rank of $S(\tilde{f},\alpha_0\tilde{g})$ is equal to 24, which is incorrect. Also, the results for the polynomial order $(g,f)$, which are not shown, are unsatisfactory because the numerical rank of $S(\tilde{g},\tilde{f}/\alpha_0)$ is not defined. Figures 4 and 5 show the results when SNTLN is applied, and it is seen that the numerical rank of $S(\tilde{f},\alpha^*\tilde{g})$ is not well defined because it could be equal to either 34 or 35 (Figure 4), but the numerical rank of $S(\tilde{g},\tilde{f}/\alpha^*)$ has the correct value of 35 (Figure 5). These figures show that if the numerical rank is defined as the index $i$ for which the ratio of singular values $\sigma_i/\sigma_{i+1}$ is a maximum, then the numerical rank of $S(\tilde{f},\alpha^*\tilde{g})$ and $S(\tilde{g},\tilde{f}/\alpha^*)$ is equal to 39, which is incorrect.

Figure 6 shows the variation of the normalised residual (33) for $S(f,g)$, using

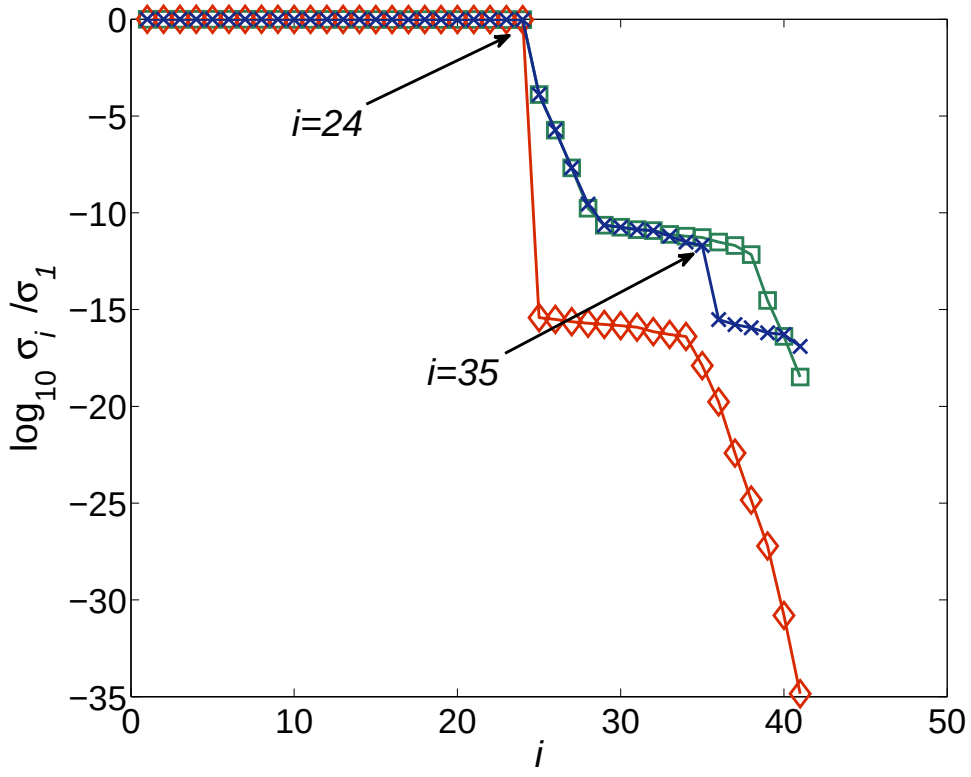



Fig. 3. The normalised singular values of the Sylvester matrices $S(\hat{f}, \hat{g})$ $\diamondsuit$; $S(f, g)$ $\square$; $S(\tilde{f}, \alpha_0 \tilde{g})$ $\times$, for Example 6.1. The polynomials $\tilde{f}(w)$ and $\tilde{g}(w)$ are calculated using STLN.

STLN and SNTLN, and it is seen that the differences in the graphs are minor. Comparison of these graphs with their equivalents for $S(g, f)$, which are shown in Figure 7, shows two significant differences:

(1) The normalised residual obtained with $S(f, g)$ is much smaller than the normalised residual obtained with $S(g, f)$ when STLN is used.
(2) Approximately twice the number of iterations are required to achieve convergence with $S(g, f)$ with respect to the number of iterations required to achieve convergence with $S(f, g)$ when SNTLN is used.

$\square$

*Example 6.2*

The procedures described in Example 6.1, including the addition of noise with a normwise signal-to-noise ratio of $10^8$, were applied to the polynomials

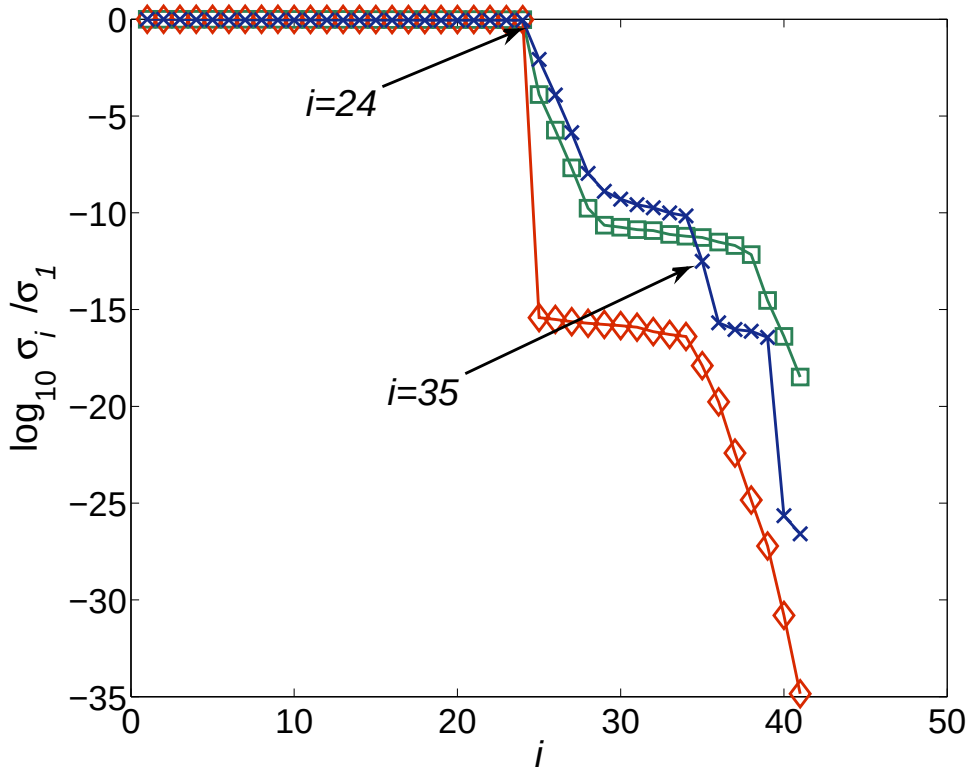



Fig. 4. The normalised singular values of the Sylvester matrices $S(\hat{f},\hat{g})$ $\diamondsuit$; $S(f,g)$ $\square$; $S(\tilde{f},\alpha^*\tilde{g})$ $\times$, for Example 6.1. The polynomials $\tilde{f}(w)$ and $\tilde{g}(w)$ are calculated using SNTLN.

$$\begin{aligned}\hat{f}(y) = &(y-1.8722181\times10^7)^5(y-0.3124444)^2\\ &\times(y-4.4199430\times10^5)^7, \qquad (36)\\ \hat{g}(y) = &(y-1.8722181\times10^7)^2(y-0.3124444)^6(y-8.8081342)^2\\ &\times(y+1.6888534)^7(y+4.5594954)^9. \qquad (37)\end{aligned}$$

The Sylvester matrix of these polynomials is of order $40\times40$ and the degree of their GCD is 4, and thus rank $S(\hat{f},\hat{g}) = 36$.

Figures 8 and 9 show the results, for the polynomial orders $(f,g)$ and $(g,f)$ respectively, obtained from STLN when $\alpha$ and $\theta$ are constant and equal to one, such that the only preprocessing operation is the normalisation of each polynomial by the geometric mean of its coefficients. It is seen that

$$\text{rank } S(\hat{f},\hat{g}) = \text{rank } S(f,g) = \text{rank } S(\tilde{f},\tilde{g}) = 26,$$

and

$$\text{rank } S(\hat{g},\hat{f}) = \text{rank } S(g,f) = \text{rank } S(\tilde{g},\tilde{f}) = 26,$$

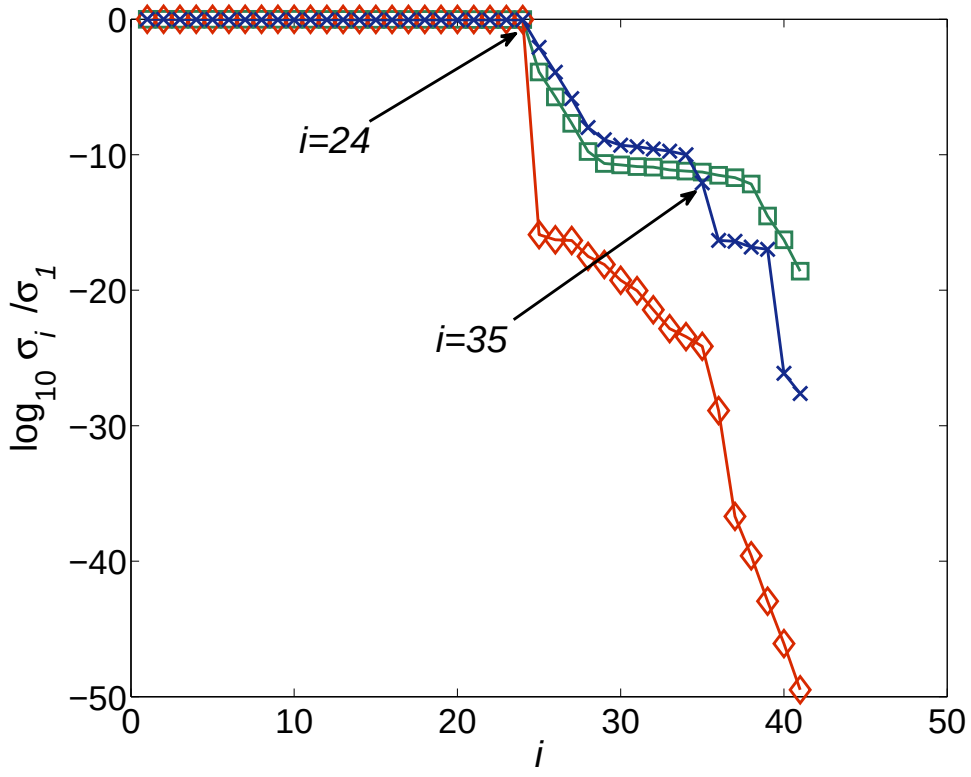
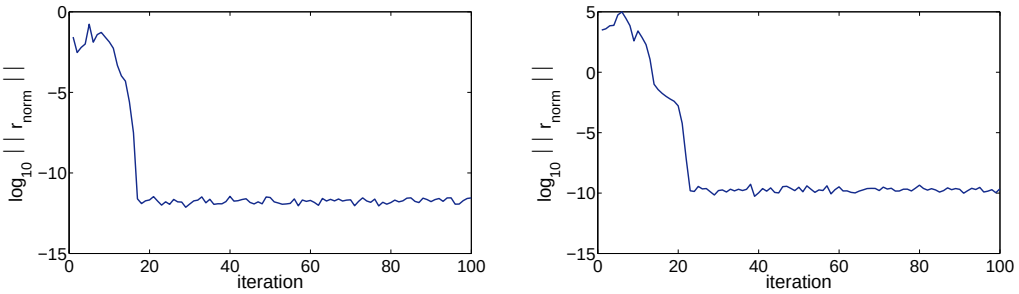



Fig. 5. The normalised singular values of the Sylvester matrices $S(\hat{g}, \hat{f})$ $\diamondsuit$; $S(g, f)$ $\square$; $S(\tilde{g}, \tilde{f}/\alpha^*)$ $\times$, for Example 6.1. The polynomials $\tilde{f}(w)$ and $\tilde{g}(w)$ are calculated using SNTLN.

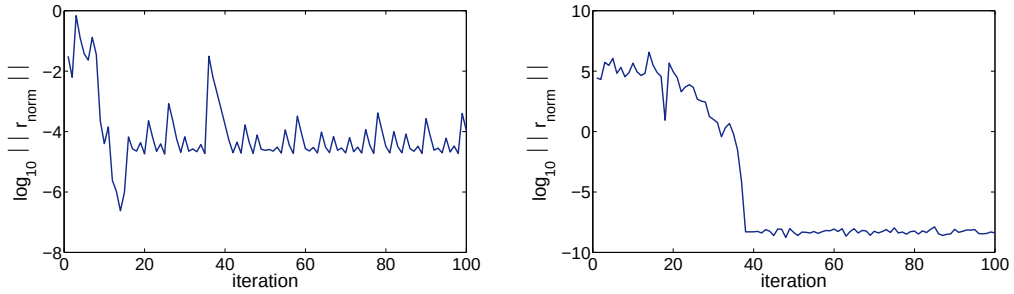



(a) (b)

Fig. 6. The variation of the normalised residual with the number of iterations, for Example 6.1, using (a) STLN and (b) SNTLN, for $S(f, g)$.

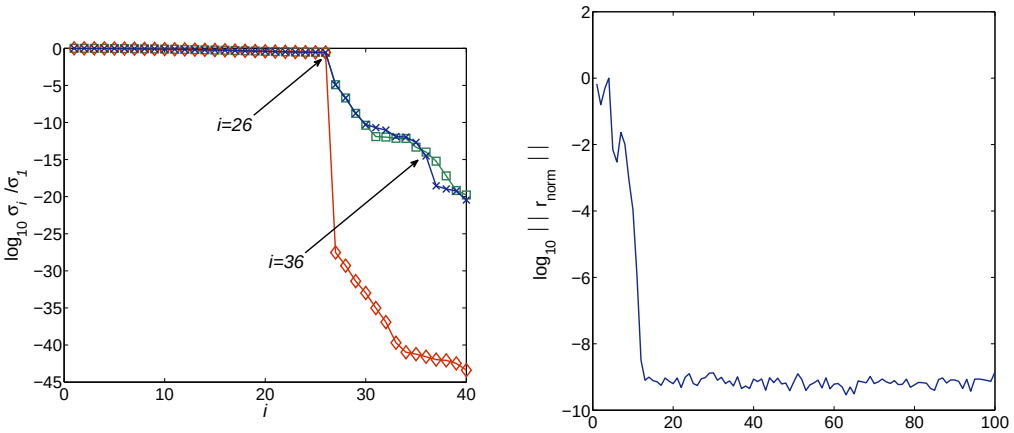
all of which are incorrect, and the normalised residuals, shown in Figures 8(b) and 9(b), at convergence are equal to about $10^{-9}$ and $10^{-10}$, respectively. This is another example that shows that a small normalised residual does not guarantee that a structured low rank approximation of a Sylvester matrix has been computed.

Figure 10 shows that the preprocessing operations summarised in Algorithm




(a) (b)

Fig. 7. The variation of the normalised residual with the number of iterations, for Example 6.1, using (a) STLN and (b) SNTLN, for $S(g, f)$.



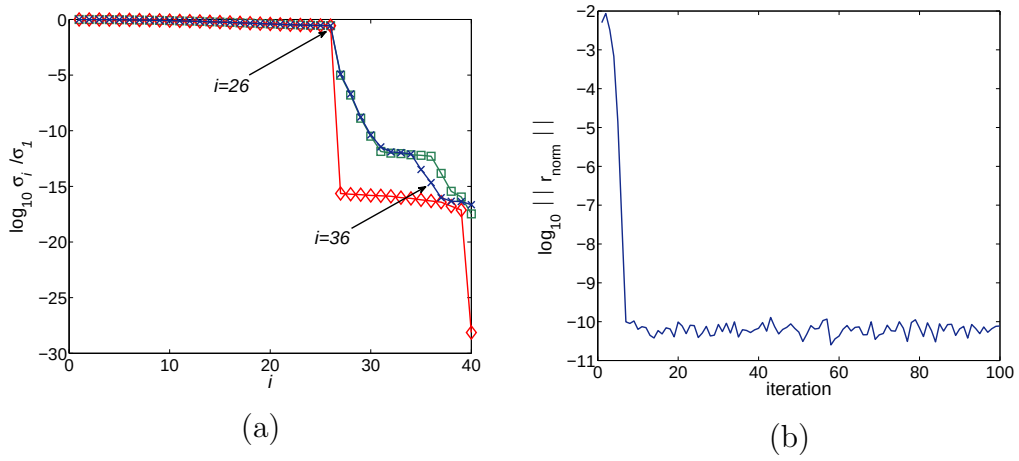

(a) (b)

Fig. 8. (a) The normalised singular values of the Sylvester matrices $S(\hat{f}, \hat{g})$ $\diamondsuit$; $S(f, g)$ $\square$; $S(\tilde{f}, \tilde{g})$ $\times$, and (b) the normalised residual, for Example 6.2. The preprocessing operations, apart from normalising each polynomial by the geometric mean of its coefficients, are omitted.

3.1 reduce considerably the ratio of the maximum magnitude of the coefficients to the minimum magnitude of the coefficients, for both polynomials.

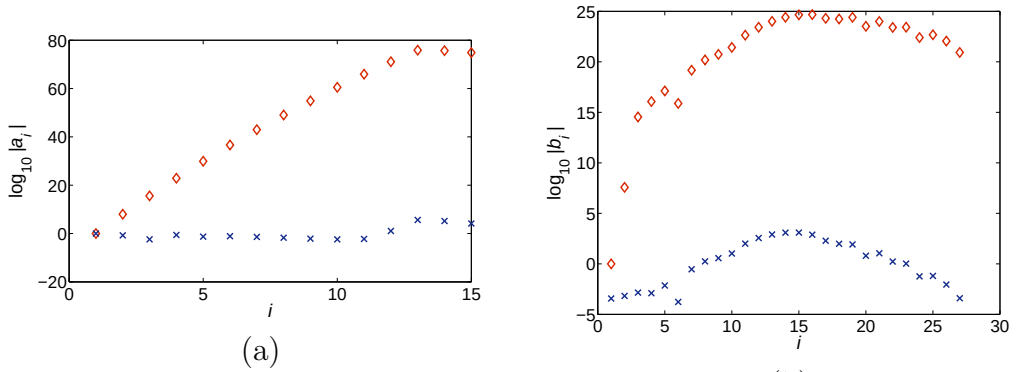
Figures 11 and 12 show the results using STLN and SNTLN, respectively, for the order $(g, f)$. Figure 11 shows that STLN does not compute a structured low rank approximation because the numerical rank could be equal to either 26 or 36, and Figure 12 shows that SNTLN does not compute a structured low rank approximation because the computed numerical rank is equal to 35. The graphs shown in these figures are very similar to their equivalents when the order $(f, g)$ is used.

Figure 13 shows that the variation of the normalised residual (33) is the same for STLN and SNTLN. In particular, the normalised residual at convergence is the same for both methods, and the number of iterations required to achieve




(a) (b)

Fig. 9. (a) The normalised singular values of the Sylvester matrices $S(\hat{g}, \hat{f})$ $\diamondsuit$; $S(g, f)$ $\square$; $S(\tilde{g}, \tilde{f})$ $\times$, and (b) the normalised residual, for Example 6.2. The preprocessing operations, apart from normalising each polynomial by the geometric mean of its coefficients, are omitted.




(a) (b)

Fig. 10. The magnitude of the coefficients of (a) $f(y)$ and (b) $g(y)$ before, $\diamondsuit$, and after, $\times$, scaling by $\alpha$ and $\theta$, for Example 6.2.

convergence is the same for both methods. $\square$

Theoretical bounds for the approximations obtained by STLN and SNTLN have not been obtained, and thus *a posteriori* checks on the computed solutions are required. These checks are performed in Examples 6.1 and 6.2 by plotting the singular values of the Sylvester matrix $S(\tilde{f}, \tilde{g})$ of the corrected polynomials $\tilde{f}(w)$ and $\tilde{g}(w)$, and verifying that it is numerically singular, which implies that these polynomials have a non-constant common divisor.

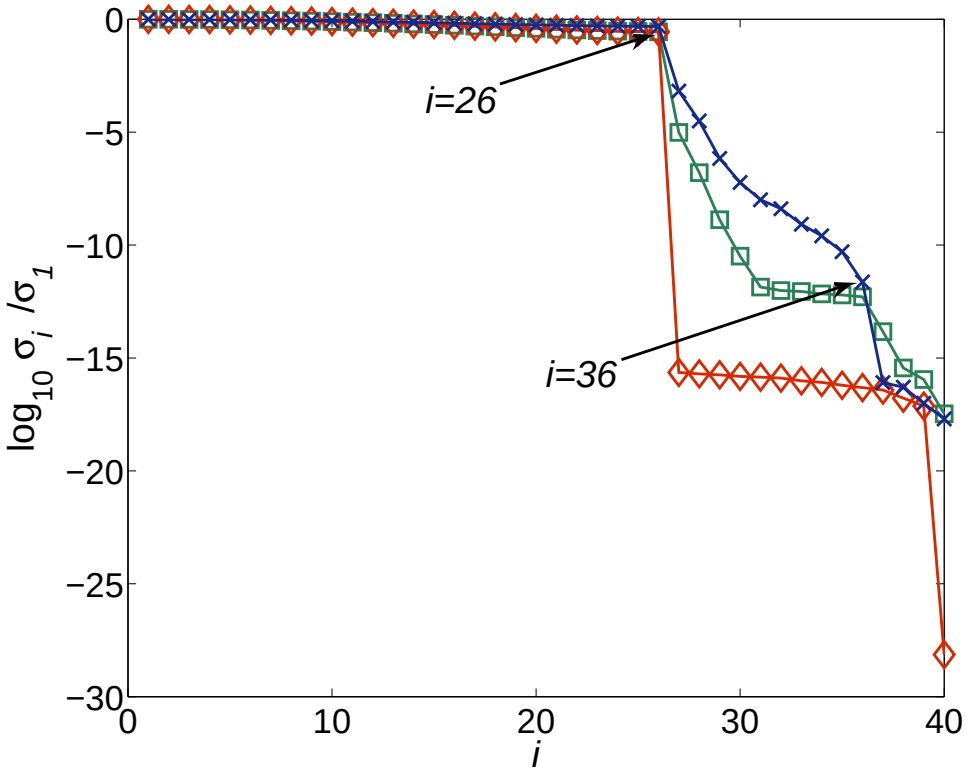



Fig. 11. The normalised singular values of the Sylvester matrices $S(\hat{g}, \hat{f})$ $\diamondsuit$; $S(g, f)$ $\square$; $S(\tilde{g}, \tilde{f}/\alpha_0)$ $\times$, for Example 6.2. The polynomials $\tilde{f}(w)$ and $\tilde{g}(w)$ are calculated using STLN.

## 7 Summary and discussion

This paper has considered the use of STLN and SNTLN for the calculation of an AGCD of two inexact polynomials. The preprocessing operations discussed in Section 3 introduce the parameters $\alpha$ and $\theta$, and the examples suggest they are important for the calculation of a structured low rank approximation of the Sylvester matrix of $f(y)$ and $g(y)$.

The examples show that the results obtained for $S(f, g)$ are not guaranteed to be equal to the results for $S(g, f)$, and in particular, STLN and/or SNTLN may compute a structured low rank approximation of one of these Sylvester matrices, but not the other Sylvester matrix. If both methods are able to compute a structured low rank approximation of $S(f, g)$ and $S(g, f)$, then the numerical rank obtained with SNTLN is more clearly defined than the numerical rank obtained with STLN, and fewer iterations are, in general, required to achieve convergence. The method of SNTLN therefore yields, in general, better results, which is expected because a wider class of perturbations is allowed by this method than is allowed by STLN. In particular, the optimal

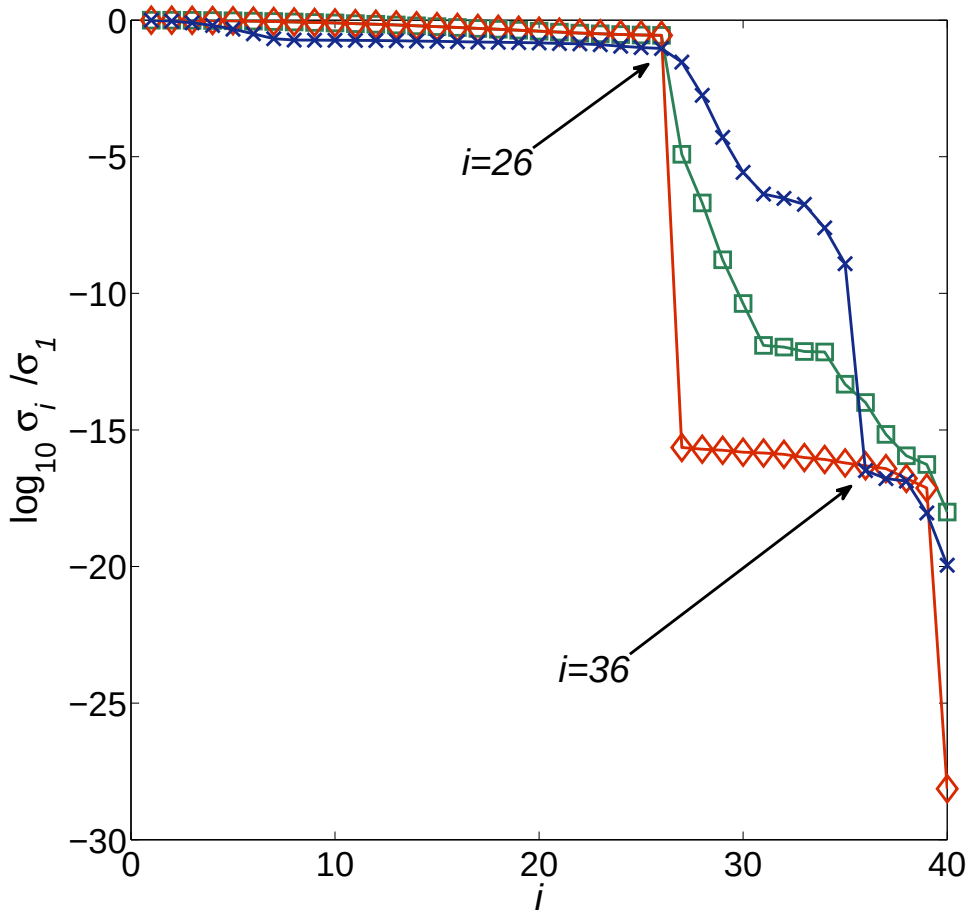
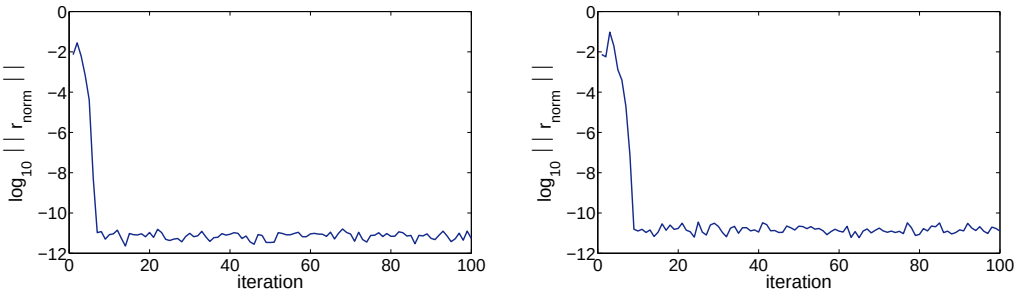



Fig. 12. The normalised singular values of the Sylvester matrices $S(\hat{g}, \hat{f})$ $\diamondsuit$; $S(g, f)$ $\square$; $S(\tilde{g}, \tilde{f}/\alpha^*)$ $\times$, for Example 6.2. The polynomials $\tilde{f}(w)$ and $\tilde{g}(w)$ are calculated using SNTLN.




(a) (b)

Fig. 13. The variation of the normalised residual with the number of iterations, for Example 6.2, using (a) STLN and (b) SNTLN, for $S(g, f)$.

values of $\alpha, \theta$ and $z$ are computed by SNTLN, but only the optimal value of $z$ is computed by STLN, and $\alpha$ and $\theta$ are constants whose values $\alpha_0$ and $\theta_0$,

respectively, are defined by the given inexact data.

The results show it is necessary to determine the best matrix, $S(f, g)$ or $S(g, f)$, to use for the computation of a structured low rank approximation of the Sylvester matrix of $f(y)$ and $g(y)$. A criterion that enables the optimal matrix, $S(f, g)$ or $S(g, f)$, to be determined will improve the quality of this structured low rank approximation.

## References


[1] D. A. Bini and P. Boito. Structured matrix-based methods for $\epsilon$-gcd: Analysis and comparisons. In *Proc. Int. Symp. Symbolic and Algebraic Computation*, pages 9–16. ACM Press, New York, 2007.

[2] R. M. Corless, P. M. Gianni, B. M. Trager, and S. M. Watt. The singular value decomposition for polynomial systems. In *Proc. Int. Symp. Symbolic and Algebraic Computation*, pages 195–207. ACM Press, New York, 1995.

[3] R. M. Corless, S. M. Watt, and L. Zhi. *QR* factoring to compute the GCD of univariate approximate polynomials. *IEEE Trans. Signal Processing*, 52(12):3394–3402, 2004.

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

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

[6] G. H. Golub and C. F. Van Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, USA, 1996.

[7] J. T. Kajiya. Ray tracing parametric patches. *Computer Graphics*, 16:245–254, 1982.

[8] E. Kaltofen, Z. Yang, and L. Zhi. Structured low rank approximation of a Sylvester matrix, 2005. Preprint.

[9] N. Karmarkar and Y. N. Lakshman. Approximate polynomial greatest common divsior and nearest singular polynomials. In *Proc. Int. Symp. Symbolic and Algebraic Computation*, pages 35–39. ACM Press, New York, 1996.

[10] B. Li, Z. Yang, and L. Zhi. Fast low rank approximation of a Sylvester matrix by structured total least norm. *J. Japan Soc. Symbolic and Algebraic Comp.*, 11:165–174, 2005.

[11] V. Y. Pan. Computation of approximate polynomial GCDs and an extension. *Information and Computation*, 167:71–85, 2001.

[12] S. Petitjean. Algebraic geometry and computer vision: Polynomial systems, real and complex roots. *Journal of Mathematical Imaging and Vision*, 10:191–220, 1999.

[13] J. Ben Rosen, H. Park, and J. Glick. Total least norm formulation and solution for structured problems. *SIAM J. Mat. Anal. Appl.*, 17(1):110–128, 1996.

[14] J. Ben Rosen, H. Park, and J. Glick. Structured total least norm for nonlinear problems. *SIAM J. Mat. Anal. Appl.*, 20(1):14–30, 1998.

[15] J. R. Winkler and J. D. Allan. Structured low rank approximations of the Sylvester resultant matrix for approximate GCDs of Bernstein polynomials. *Electronic Transactions on Numerical Analysis*, 31:141–155, 2008.

[16] J. R. Winkler and J. D. Allan. Structured total least norm and approximate GCDs of inexact polynomials. *Journal of Computational and Applied Mathematics*, 215:1–13, 2008.

[17] J. R. Winkler and X. Y. Lao. The calculation of the degree of an approximate greatest common divsior of two polynomials, 2009. Submitted.

[18] C. J. Zarowski, X. Ma, and F. W. Fairman. $QR$-factorization method for computing the greatest common divisor of polynomials with inexact coefficients. *IEEE Trans. Signal Processing*, 48(11):3042–3051, 2000.