Chapter 1

NUMERICAL AND SYMBOLICAL METHODS COMPUTING THE GREATEST COMMON DIVISOR OF SEVERAL POLYNOMIALS

Dimitrios Christou,[1] Nicos Karcanias,[1]
Marilena Mitrouli [2] and Dimitrios Triantafyllou[2]

[1] *School of Engineering and Mathematical Sciences, Control Engineering Research Centre, City University, Northampton Square, London, EC1V 0HB, UK*
dchrist@math.uoa.gr, N.Karcanias@city.ac.uk

[2] *Department of Mathematics, University of Athens, Panepistemiopolis 15784, Athens, Greece*
mmitroul@math.uoa.gr, dtriant@math.uoa.gr

Abstract

The computation of the Greatest Common Divisor (GCD) of a set of polynomials is an important issue in computational mathematics and it is linked to Control Theory very strong. In this paper we present different matrix-based methods, which are developed for the efficient computation of the GCD of several polynomials. Some of these methods are naturally developed for dealing with numerical inaccuracies in the input data and produce meaningful approximate results. Therefore, we describe and compare numerically and symbolically methods such as the ERES, the Matrix Pencil and other resultant type methods, with respect to their complexity and effectiveness. The combination of numerical and symbolic operations suggests a new approach in software mathematical computations denoted as *hybrid computations*. This combination offers great advantages, especially when we are interested in finding approximate solutions. Finally the notion of approximate GCD is discussed and a useful criterion estimating the *strength* of a given approximate GCD is also developed.

Keywords: Greatest Common Divisor, Gaussian elimination, Resultant, Matrix-Pencil, Singular Value Decomposition, symbolic-numeric computations

1. Introduction

The problem of finding the greatest common divisor (GCD) of a polynomial set has been a subject of interest for a very long time and has widespread applications. Since the existence of a nontrivial common divisor of polynomials is a property that holds for specific sets, extra care is needed in the development of efficient numerical algorithms calculating correctly the required GCD. Several numerical methods for the computation of the GCD of a set $P_{m,n}$, of m polynomials of $\Re[s]$ of maximal degree n, have been proposed, [2–4, 11, 20, 24, 27, 29, 31, 33, 37] and references therein. These methods can be classified as :

(i) Numerical methods based on Euclid's algorithm and its generalizations.

(ii) Numerical methods based on procedures involving matrices (matrix based methods).

The methods that are based on Euclid's algorithm, are designed for processing two polynomials and they are applied iteratively for sets of more than two polynomials. On the other hand, the matrix-based methods usually perform specific transformations to a matrix formed directly from the coefficients of the polynomials of the entire given set.

The GCD has a significant role in Control Theory [15, 32]. A number of important invariants for Linear Systems rely on the notion of Greatest Common Divisor (GCD) of several polynomials. In fact, it is instrumental in defining system notions such as zeros, decoupling zeros, zeros at infinity or notions of Minimality of system representations. On the other hand, Systems and Control Methods provide concepts and tools, which enable the development of new computational procedures for GCD [16].

The existence of certain types and/or values of invariants and system properties may be classified as *generic* or *nongeneric* on a family of linear models. Numerical computations dealing with the derivation of an approximate value of a property, function, which is nongeneric on a given model set, will be called nongeneric computations (NGC) [16]. Computational procedures aiming at defining the generic value of a property, function on a given model set (if such values exists), will be called generic (GC). On a set of polynomials with coefficients taking values from a certain parameter set, the existence of GCD is nongeneric [14, 36]; numerical procedures that aim to produce an approximate nontrivial value by exploring the numerical properties of the parameter set are typical examples of NGC computations and approximate GCD pro-

cedures will be considered subsequently. NG computations refer to both continuous and discrete type system invariants. The various techniques, which have been developed for the computation of approximate solutions of GCD [27, 19] and LCM (Least Common Multiple) [16, 21, 22], are based on methodologies where exact properties of these notions are relaxed and appropriate solutions are sought using a variety of numerical tests.

The development of a methodology for robust computation of nongeneric algebraic invariants, or nongeneric values of generic ones, has as prerequisites:

(a) The development of a numerical linear algebra characterization of the invariants, which may allow the measurement of degree of presence of the property on every point of the parameter set.

(b) The development of special numerical tools, which avoid the introduction of additional errors.

(c) The formulation of appropriate criteria which, allow the termination of algorithms at certain steps and the definition of meaningful approximate solutions to the algebraic computation problem.

It is clear that the formulation of the algebraic problem as an equivalent numerical linear algebra problem, is essential in transforming concepts of algebraic nature to equivalent concepts of analytic character and thus setup up the right framework for approximations.

A major challenge for the control theoretic applications of the GCD is that frequently we have to deal with a very large number of polynomials. It is this requirement that makes the pairwise type approaches for GCD [1, 24, 29, 37] not suitable for such applications [33]. However, because of the use of the entire set of polynomials, matrix-based methods tend to have better performance and quite good numerical stability, especially in the case of large sets of polynomials [2–4, 10, 20, 27].

The study of the invariance properties of the GCD [18] led to the development of the ERES method [27], which performs extensive row operations and shifting on a matrix formed directly from the coefficients of the polynomials. The ERES method has also introduced for the first time a systematic procedure for computing *approximate* GCDs [19] for a set of polynomials and extends the previously defined notion of *almost zeros* [17]. The notion of *almost zeros* is linked to the *approximate GCD* problem [19] and it is based on a relaxation of the exact notion of a zero.

Another algorithm for the GCD computation based on Systems Theory and Matrix Pencils, was introduced in 1994, [20], and a variant using generalized resultant matrices was presented in 2006 [23].

The implementation of matrix-based methods computing the GCD in a programming environment often needs a careful selection of the

proper arithmetic system. Most modern mathematical software packages use variable floating-point or exact symbolic arithmetic. If symbolic arithmetic is used, the results are always accurate, but the time of execution of the algorithms can be prohibitively high. In variable precision floating-point arithmetic the internal accuracy of the system can be determined by the user. Variable precision operations are faster and more economical in memory bytes than symbolic operations, but if we increase the number of digits of the system's accuracy, the time and memory requirements will also increase. An alternative approach is to combine symbolic with floating-point operations of enough digits of accuracy in an appropriate way. Such combination will be referred as *Hybrid Computations*. This technique often improves the performance of the matrix based algorithms.

In the following, we will be mainly concerned with the performance of the ERES, Matrix Pencil, and Resultant ERE methods in a numerical-symbolical computational environment. Also, a useful indicator for the quality of the GCD, known as the *strength of an approximate GCD* [19], is described.

2. The ERES, Resultant ERE (RERE) and Modified RERE (MRERE) Methods

In this section, we present the description of the two methods for computing the GCD of several polynomials using *Extended Row Equivalence* (ERE) [16]. Their corresponding algorithms are tested and compared thoroughly and representative examples are given in tables.

Suppose that we have a set of m polynomials :

$$\mathcal{P}_{m,n} = \Big\{ \quad a(s), b_i(s) \in \Re[s],\ i = 1, 2, \ldots, m-1 \text{ with } n = \deg\{a(s)\} \text{ and } p = \max_{1 \le i \le m-1} \big\{ \deg\{b_i(s)\} \big\} \le n \Big\}$$

with the following form :

$$a(s) = a_n s^n + a_{n-1}\, s^{n-1} + \ldots + a_1\, s + a_0$$

$$b_i(s) = b_{i,n}\, s^p + b_{i,n-1}\, s^{p-1} + \ldots + b_{i,n-p+1}\, s + b_{i,n-p}$$

$$\text{for } i = 1, 2, \ldots, m-1$$

and suppose that there is at least one i : $b_{i,n} \neq 0$ but $b_{i,j} = 0$ for $j > n - p\ \forall i$, [2].

For any $\mathcal{P}_{m,n}$ set, we define a vector representative (vr) $\underline{p}_m(s)$ and a basis matrix P_m represented as :

$$\underline{p}_m(s) = [a(s), b_1(s), \ldots, b_{m-1}(s)]^t \tag{1.1a}$$

$$= [\underline{p}_0, \underline{p}_1, \ldots, \underline{p}_{n-1}, \underline{p}_n] \cdot \underline{e}_n(s) = P_m \cdot \underline{e}_n(s) \tag{1.1b}$$

where $P_m \in \Re^{m\times(n+1)}$, $\underline{e}_n(s) = [1, s, \ldots, s^{n-1}, s^n]^t$.

The basis matrix P_m is formed directly from the coefficients of the polynomials of the set.

Additionally, for any vector of the form:

$$\underline{r}^t = [0, \ldots, 0, a_k, \ldots, a_d] \in \Re^d,\ a_k \neq 0$$

we define the *Shifting* operation

$$shf : \mathrm{shf}(\underline{r}^t) = [a_k, \ldots, a_d, 0, \ldots, 0] \in \Re^d$$

In the following, without loss of generality, we suppose that the GCD of a given set of polynomials has no zero roots.

2.1 The ERES Method

The ERES method is an iterative matrix based method, which is based on the properties of the GCD as an invariant of the original set of polynomials under extended-row-equivalence and shifting operations [18]. The algorithm of the ERES method [27], [28] is based on stable algebraic processes, such as Gaussian elimination with partial pivoting scaling, normalization and Singular Value Decomposition, which are applied iteratively on a basis matrix formed directly from the coefficients of the polynomials of the original set. Thus, the ERES algorithm works with all the polynomials of a given set simultaneously. The main target of the ERES algorithm is to reduce the number of the rows of the initial matrix and finally to end up to a unity rank matrix, which contains the coefficients of the GCD. The Singular Value Decomposition provides the ERES algorithm with a termination criterion. The performance of the algorithm is better [5] if we perform hybrid computations. The following algorithm corresponds to an implementation of the ERES method in a hybrid computational environment.

2.1.1 The ERES Algorithm.

– Form the basis matrix $P_m \in \Re^{m\times(n+1)}$.
– Convert the elements of P_m to a rational format :
$P_m^{(0)} := \mathrm{convert}(P_m, \mathrm{rational})$

– Initialize $k := -1$.

Repeat

$k := k + 1$

STEP 1: Let $r :=$ the row dimension of $P_m^{(k)}$.
Specify the degree d_i of each polynomial row.
Reorder matrix $P_m^{(k)}$: $d_{i-1} \leq d_i,\ i = 2, \ldots, r$.
If $d_1 = d_2 = \ldots = d_r$ **then**
Convert the elements of $P_m^{(k)}$ to a floating-point format:
$P_m^{(F)} :=$ convert$(P_m^{(k)}, \text{float})$
Normalize the rows of $P_m^{(F)}$ using norm-2 :
$P_m^{(N)} :=$ Normalize$(P_m^{(F)})$
Compute the Singular Value Decomposition :
$P_m^{(N)} := V\,\Sigma\,W^t$, $\Sigma = diag\{\sigma_1, \ldots, \sigma_r\}$,
$\sigma_1 > \sigma_2 \geq \ldots \geq \sigma_r$ and $W^t = [\underline{w}_1, \ldots, \underline{w}_{n+1}]^t$
If ε_t-rank$(P_m^{(N)}) = 1$ **then**
Select the GCD vector $\underline{g}$.
($\underline{g} := \underline{w}_1^t$ or $\underline{g} :=$ any row of $P_m^{(k)}$)
quit

STEP 2: Scale properly matrix $P_m^{(k)}$.
Apply Gaussian elimination with partial pivoting to $P_m^{(k)}$.

STEP 3: Apply shifting on every row of $P_m^{(k)}$.
Delete the zero rows and columns.

until $r = 1$

The ERES algorithm produces either a single row-vector or a unity rank matrix. The main advantage of this algorithm is the reduction of the size of the original matrix during the iterations, which leads to fast data processing and low memory consumption.

2.1.2 Complexity. For a set of m polynomials the amount of floating point operations performed in the k^{th} iteration of the algorithm depends on the size of the matrix $P_m^{(k)}$. If the size of $P_m^{(k)}$ is $m' \times n'$, the ERES algorithm requires $O(\frac{z^3}{3})$, $z = \min\{m' - 1, n'\}$ operations for the Gaussian elimination, $O(2m'n')$ operations for the normalization and $O(m'n'^2 + n'^3)$ for the SVD process. The first iteration is the most computationally expensive iteration since the initial matrix $P_m^{(0)}$ has larger dimensions than any $P_m^{(k)}$. Unless we know exactly the degree of the GCD of the set we cannot specify from the beginning the number

of iterations required by the algorithm. Therefore, we cannot express a general formula for calculating the total number of operations, which are required by the algorithm.

2.1.3 Behavior and stability of the ERES algorithm. The combination of rational and numerical operations aims at the improvement of the stability of the ERES algorithm and the presence of *good* approximate solutions. The main iterative procedure of the algorithm and especially the process of Gaussian elimination, is entirely performed by using rational operations. With this technique any additional errors from the Gaussian elimination are avoided. The operations during the Gaussian elimination are always performed accurately and if the input data are exactly known and a GCD exists, the output of the algorithm is produced accurately from any row of the final unity rank matrix. Obviously, rational operations do not reveal the presence of approximate solutions. In cases of sets of polynomials with inexact coefficients, the presence of an approximate solution relies on the proper determination of a numerical ε_t-rank 1 matrix for a specific accuracy ε_t. Therefore, the singular value decomposition together with the normalization process of the matrix $P_m^{(k)}$ are performed by using floating-point operations. The polynomial that comes from the right singular vector that corresponds to the unique singular value of the last unity rank matrix, can be considered as a GCD approximation and represents the numerical output of the ERES algorithm.

The normalization of the rows of any matrix $P_m^{(k)}$ (by the Euclidean norm) does not introduce significant errors and in fact the following result can be proved [27]:

PROPOSITION 1.1 *The normalization $P_m^{(N)}$ of a matrix $P_m^{(k)} \in \Re^{m' \times n'}$, computed by the method in the k^{th} iteration, using floating-point arithmetic with unit round-off u, satisfies the properties*

$$P_m^{(N)} = N \cdot P_m^{(k)} + E_N, \quad \|E_N\|_\infty \leq 3.003 \cdot n' \cdot u$$

where $N \in \Re^{m' \times m'} = diag(d_1, d_2, \ldots, d_{m'})$, $d_i = \left(\left\| P_m^{(k)}[i, 1 \ldots n'] \right\|_2 \right)^{-1}$, $i = 1, \ldots, m'$ the matrix accounting for the performed transformations and $E_N \in \Re^{m' \times n'}$ the error matrix.

It is important to notice that the SVD is actually applied to a numerical copy of the matrix $P_m^{(k)}$ and thus the performed transformations during the SVD procedure do not affect the matrix $P_m^{(k)}$ when returning to the main iterative procedure. For this reason, there is *no accumulation*

of numerical errors. The only errors appearing are from the normalization and the singular value decomposition [8, 12] of the last matrix $P_m^{(F)}$ and represent the total numerical error[1] of the ERES algorithm.

The combination of rational-symbolic and floating-point operations ensures the stability of the algorithm and gives to the ERES the characteristics of a hybrid computational method.

2.2 The Resultant ERE (RERE) and Modified ERE (MRERE) Methods

Another method to compute the GCD of several polynomials is to triangularize the generalized Sylvester matrix S [2], which has the following form:

$$S = \begin{bmatrix} S_0 \\ S_1 \\ \vdots \\ S_m \end{bmatrix}$$

where the block $S_i, i = 1, \ldots, m$ represents the Sylvester matrix of the i-th polynomial. In [6] we have modified the huge initial generalized Sylvester matrix in order to take advantage of its special form and modifying the classical procedures (such as SVD, QR and LU factorization) we reduce the required floating point operations from $O(n^4)$ to $O(n^3)$ flops making the algorithms efficient. More specifically the application of Householder or Gaussian transformations to the modified generalized Sylvester matrix requires only $O((n+p)^3(2log_2(n) - \frac{1}{3}) + (n+p)^2(2mlog_2(n) + p))$ flops and in the worse case where $m = n = p$ the required flops will be equal to $O(16log_2(n) \cdot n^3)$ or the half flops for the LU factorization respectively. In practice the flops that demand the previous methods are less because of the linear dependent rows which are zeroed and deleted during the triangularization of the matrix.

2.2.1 Numerical Stability. In [6] we proved that the final error matrix in the modified QR method is $E = \sum_{i=1}^{log_2(n)} E_i$ with

$$\| E \|_F \leq \phi(n)\mu\sqrt{n+p}(\| S^* \|_F + \| ((S^*)') \|_F) \tag{1.2}$$

where $((S^*)')$ is the last triangularized sub-matrix, ϕ a slowly growing function of n and μ the machine precision and the final error in the modified LU method is

$$E = \sum_{i=1}^{log_2(n)} E_i \tag{1.3}$$

with $\| E \|_\infty \leq (n+p)^{[log_2 n]} pu \| S^* \|_\infty + (n+p)^2 pu \| ((S^*)') \|_\infty)$, where p is the growth factor and u the unit round off.

3. The Matrix Pencil Methods

3.1 The Standard Matrix Pencil Method (SMP)

The matrix pencil method [20, 23] is a direct method, which is based on system properties of the GCD. The algorithm of the SMP method uses stable processes, such as SVD for computing the right $\mathcal{N}_r$ and left $\mathcal{N}_l$ null spaces of appropriate matrices. The SMP method requires the construction of the observability matrix $Q(\hat{A}, \hat{C}) = [\hat{C}^t, \hat{A}^t\hat{C}^t, \ldots, (\hat{A}^t)^{(d-1)}\hat{C}^t]^t$ of the companion matrix $\hat{A}$ of the polynomial of maximal degree and the left null space $\hat{C}$ of a matrix M_1 as we will see below. It is known that the computation of powers of matrices is not always stable. As it is shown in [23], because of the special form of the companion matrix A and the orthogonality of C, the powers of A and their product with C can fail only if it holds very special formulas between the coefficients of the polynomials ([23] example (8)). An alternative way is this computation to be done symbolically: there will be no rounding off errors and because only an inner product must be computed for the last column for every computation of a power of A (the other columns are the columns of the previous power of A left shifted), the increase of the computational time because of the rational representation of the coefficients and the symbolical computation of the products is not very considerable. In this manner we achieve to compute the observability matrix avoiding one of the main disadvantages of the SMP method (the computation of the powers of A) without significant surcharge of the required time.

Another stable way to compute the null space of Q is first to reduce the pair $(\hat{A}, \hat{C})$ to a block Hessenberg form without computing the observability matrix. From the staircase algorithm [9], we take an orthogonally similar pair $(H, \widetilde{C})$, such that: $Q(\hat{A}, \hat{C}) = P^T[\widetilde{B}, H\widetilde{B}, \ldots, H^{(d-1)}\widetilde{B}]$, where P is an orthogonal matrix such that $P\hat{A}P^T = H$. Because the matrix H has much more elements than the companion matrix A, the computation of the powers of H demands more flops than those of the powers of A and thus in our case it is better to use the first way for the computation of the null space of Q.

The main target of the SMP algorithm is to form the GCD pencil $Z(s)$ and specify any minor of maximal order, which gives the required GCD. This specification can be done symbolically. Let $\mathcal{P}_{m,d}$ be the set of polynomials as defined in section II.

3.1.1 The SMP Algorithm.

STEP 1 : Compute a basis matrix M for the right nullspace $\mathcal{N}_r(P_m)$ using the SVD algorithm.
STEP 2 : Construct M_1 by deleting the last row of M
STEP 3 : Compute the matrix $\hat{C}$ such that $\hat{C}M_1 = 0$
STEP 4 : Construct the observability matrix:
$Q(\hat{A}, \hat{C}) = [\hat{C}^t, \hat{A}^t\hat{C}^t, \ldots, (\hat{A}^t)^{(d-1)}\hat{C}^t]^t$
STEP 5 : Compute the right nullspace $W = \mathcal{N}_r(Q(\hat{A}, \hat{C}))$ using the SVD algorithm.
STEP 6 : Construct the pencil $Z(s) = sW - \hat{A}W$. Any minor of maximal order of $Z(s)$ defines the GCD of set of the polynomials.

3.1.2 Complexity. The computation of the right nullspace of P_m requires $O(mn^2 + \frac{11n^3}{2})$ flops, of the of the matrix C demands $O((r-1)n^3 + \frac{11n^3}{2})$ flops, where $r = \rho(P_m)$. The computation of $\hat{C}$ such that $\hat{C}M_1 = 0$ requires $O\,(4\mu d^2 + 8d^3)$ flops applying the SVD algorithm to M_1^t, where $\mu = d - r + 1, r = \rho(P_m)$ (*).
The computation of any minor of maximal order of $Z(s)$ can be done symbolically using the LU factorization.

Totally the Standard Matrix Pencil method demands: $O\,(4md^2 + 4d^3(r-1) + 4\mu d^2 + 24d^3 + \frac{3}{2}d^2r)$ flops. If the computation of Q is done symbolically the required flops are diminished by the flops in (*) but the computational time is increased slightly.

3.1.3 Numerical Stability. The Standard Matrix Pencil Method requires two SVD calls and the construction of the observability matrix. The numerical computation of the powers of A $(\hat{A})^{(k)}$ is in practise stable. Of course there are no errors in symbolically implementation of this step. Since the matrix $(\hat{A})^{(k)}$ is computed, the numerical computation of the product $(\hat{A}^t)^{(k)}\hat{C}^t = (\hat{C}(\hat{A})^{(k)})^t$ is stable because the matrix C is orthonormal. For the last matrix multiplication it holds : $fl(\hat{C}(\hat{A})^{(k)}) = \hat{C}(\hat{A})^{(k)} + E$, with $\|E\|_2 \leq d^2u_1\|C\|_2\|A^k\|_2 = d^2u_1\|A^k\|_2$, where $fl(\cdot)$ denotes the computed floating point number and u_1 is of order of unit round off. The computation of the minor of maximal order of $Z(s)$ is done symbolically and so there are no rounding off errors.

3.2 The Modified Resultant Matrix Pencil Method (MRMP)

The modified matrix pencil method [23] is a similar with the MP method, which is based to the modified Sylvester matrix S^*, it constructs

another GCD pencil $Z(s)$ and specify any minor of maximal order, which gives the required GCD. This specification can also be done symbolically.

3.2.1 The MRMP Algorithm.

STEP 1 : Define a basis $\widetilde{M}$ for the right nullspace of the modified Sylvester matrix S^* using the modified QR factorization in first phase of SVD.

STEP 2 : Define the Matrix Pencil $\widetilde{Z}(s) = s\widetilde{M_1} - \widetilde{M_2}$ for the Resultant set, where $\widetilde{M_1}$, $\widetilde{M_2}$ are the matrices obtained from $\widetilde{M}$ by deleting the last and the first row of $\widetilde{M}$ respectively.

STEP 3 : Compute any non-zero minor determinant $d(s)$ of $\widetilde{Z}(s)$ and thus obtain GCD=$d(s)$

3.2.2 Complexity. The Modified Resultant Matrix Pencil method requires $O((n+p)^3(2log_2(n) - \frac{1}{3}) + (n+p)^2(2mlog_2(n) + p) + 12k(n+p)^2)$, where k is the number of the calls of the SVD-step.

3.2.3 Numerical Stability. The computed GCD is the exact GCD of a slightly disrupted set of the initial polynomials. The final error is $E = E_1 + E_2$, with $\|E_1\|_F \leq \varphi(n)u\|S\|_F$ and $\|E\|_2 \leq (\varphi(n) + c(m,n) + c(m,n) \cdot \varphi(n) \cdot u) \cdot u \cdot \|S\|_F$ where u is the unit round off error, $\|\cdot\|_F$ the Frobenius norm and $\varphi(n)$ is a slowly growing function of n [8] and $c(m,n)$ is a constant depending on m, n.

3.3 Another Subspace-Based Method for Computing the GCD of Several Polynomials (SS)

The subspace concept is actually very common among several methods for computing the GCD of many polynomials, including those we described in the previous sections. The SVD procedure applied to a generalized Sylvester matrix is the basic tool for a subspace method. A representative and rather simple algorithm, which approaches the GCD problem from the subspace concept, is presented in [31] and we shall refer to it as the *SS algorithm*.

Given a set of univariate polynomials $\mathcal{P}_{m,n}$, the first two steps of the SS algorithm involves the construction of an $m(n+1) \times (2n+1)$ generalized Sylvester matrix Y from the input polynomials and the computation of the left null space of the transposed Y^t via SVD. If we denote by $U_0 \in \Re^{(2n+1)\times k}$ the basis matrix for the computed left null space of Y^t and C is the $(2n+1) \times (2n+1-k)$ Toeplitz matrix of a degree

K polynomial with arbitrary coefficients, then the GCD vector is actually the unique (up to a scalar) solution of the system $U_0^t C = 0$, [31]. Obviously, the degree of the GCD is $k = \text{colspan}\{U_0\}$.

For the approximate GCD problem, an equivalent and more appropriate way to compute the GCD vector with the SS algorithm is to construct k Hankel matrices $\tilde{U}_i \in \Re^{(k+1)\times(2n+1-k)}$, $i = 1, \ldots, k$ from the columns of U_0, form the matrix $\tilde{U} = [\tilde{U}_1, \ldots, \tilde{U}_k] \in \Re^{(k+1)\times k(2n+1-k)}$ and compute a basis matrix V_0 for the left null space of $\tilde{U}$ by using the SVD procedure. The last column of V_0, which corresponds to the smallest singular value (expected to be zero), contains the $k+1$ coefficients of the GCD. The yielded GCD can be considered as an approximate ε-GCD for a tolerance ε equal to the machine's numerical precision. However, for a different tolerance ε, we can select a singular value σ_j from the singular value decomposition of Y^t such that $\sigma_j > \varepsilon \cdot f(h,n)$ and $\sigma_{j+1} \leq \varepsilon$, [7], and compute an ε-GCD of degree $k' = 2n + 1 - j \neq k$.

Although it is not mentioned in [31], the computational cost of the SS algorithm is dominated by the SVD of the generalized Sylvester matrix Y^t, which requires $O(2\,m^2n^3 + 5\,m^2n^2)$ flops, [12]. However, the stability and the effectiveness of the algorithm in large sets of polynomials is not well documented in [31] and additionally there not any reference about the total numerical error of the algorithm. Practically, the performance of the SS algorithm is good when using floating-point operations of medium-high accuracy but becomes very slow in hybrid computations.

4. Approximate Solutions

It is well known that, when working with inexact data in a computational environment with limited numerical accuracy, the outcome of a numerical algorithm is usually an approximation of the expected exact solution due to the accumulation of numerical errors. In the case of GCD algorithms, the solution produced can be considered either as an approximate solution of the original set of polynomials, within a tolerance ε, or as the exact solution of a perturbed set of polynomials. The following definition is typical for the approximate GCD.

DEFINITION 1.1 *Let* $\mathcal{P}_{m,n} = \{a(s), b_i(s),\ i = 1 \ldots m-1\}$ *a set of univariate polynomials as defined in (1.0) and* $\varepsilon > 0$ *a fixed numerical accuracy. An almost common divisor (*ε*-divisor) of the polynomials of the set* $\mathcal{P}_{m,n}$ *is an exact common divisor of a perturbed set of polynomials* $\mathcal{P}'_{m,n} = \{a(s) + \Delta a(s), b_i(s) + \Delta b_i(s),\ i = 1 \ldots m-1\}$, *where the polynomial perturbations satisfy* $\deg\{\Delta a(s)\} \leq \deg\{a(s)\}$,

$\deg\{\Delta b_i(s)\} \leq \deg\{b_i(s)\}$ *and*

$$\|\Delta a(s)\|^2 + \sum_{i=1}^{m-1} \|\Delta b_i(s)\|^2 < \varepsilon \tag{1.4}$$

An approximate GCD (ε-GCD) of the set $\mathcal{P}_{m,n}$ is an ε-divisor of maximum degree.

The computation of the GCD of a set polynomials is nongeneric. Generally, in most GCD problems the degree of the GCD is unknown and thus a numerical algorithm can easily produce misleading results. An approach to this problem is to certify and fix a maximum degree according to appropriate theorems and techniques [11, 33] and proceed with the computation of a common divisor of this particular degree. The evaluation of the strength of a given approximation is another important issue here.

The definition of the *approximate GCD* as the exact GCD of a perturbed set has led to the development of a general approach for defining the approximate GCD, evaluating the strength of approximation and finally defining the notion of the optimal *approximate GCD* as a distance problem [19]. In fact, recent results on the representation of the GCD [13, 19], using Toeplitz matrices and generalized resultants (Sylvester matrices), allow the reduction of the approximate GCD computation to an equivalent *approximate factorization* of generalized resultants. Specifically, for a given set of polynomials $\mathcal{P}_{m,n}$ and its GCD of degree k :

$$g(s) = s^k + \lambda_1 s^{k-1} + \ldots + \lambda_k,\ \lambda_k \neq 0$$

it holds that [13] :

$$S_{\mathcal{P}} = [O_k | S_{\mathcal{P}^c}] \cdot \Phi_g \tag{1.5}$$

where $S_{\mathcal{P}}$ is the $(mn+p) \times (n+p)$ Sylvester matrix of the set $\mathcal{P}_{m,n}$, O_k is the $(mn+p) \times k$ zero matrix, $S_{\mathcal{P}^c}$ is the $(mn+p) \times (n+p-k)$ Sylvester matrix of the set $\mathcal{P}^c_{m,n-k}$ of coprime polynomials, obtained from the original set $\mathcal{P}_{m,n}$ after dividing its elements by the GCD $g(s)$ and, finally, Φ_g is the $(n+p) \times (n+p)$ lower triangular Toeplitz-like matrix of the polynomial $g(s)$.

We now define the strength of an r-order approximate common divisor of a polynomial set $\mathcal{P}_{m,n}$ [19] :

DEFINITION 1.2 *Let $\mathcal{P}_{m,n}$ and $v(s) \in \Re[s]$, $deg\{v(s)\} = r \leq p$. The polynomial $v(s)$ is an r-order approximate common divisor of $\mathcal{P}_{m,n}$ and its* strength *is defined as a solution of the following minimization problem :*

$$f(\mathcal{P}, \mathcal{P}^c) = \min_{\forall\, \mathcal{P}^c} \{\|S_{\mathcal{P}} - [O_r | S_{\mathcal{P}^c}] \cdot \Phi_v\|_F\} \tag{1.6}$$

Furthermore, $v(s)$ is an r-order approximate GCD of $\mathcal{P}_{m,n}$ if the minimum corresponds to a coprime set $\mathcal{P}^c_{m,n-r}$, or to a full rank $S_{\mathcal{P}^c}$.

We prefer to use as a metric the Frobenius matrix norm [8] denoted by $\|\cdot\|_F$, which relates in a direct way to the set of polynomials. However, the minimization problem in Definition 1.2 cannot be solved easily, because it may involve too many arbitrary parameters.

Let us have a polynomial $v(s) \in \Re[s]$ of degree r given as a solution by a GCD algorithm. We consider it as an exact GCD of a perturbed set of polynomials $\mathcal{P}'_{m,n}$ of the form :

$$\begin{aligned} \mathcal{P}'_{m,n} &\triangleq \mathcal{P}_{m,n} - \mathcal{Q}_{m,n} && \text{(1.7a)} \\ &= \Big\{ p'_i(s) = p_i(s) - q_i(s) : \\ &\qquad \deg\{q_i(s)\} \leq \deg\{p_i(s)\}, i = 1, \ldots, m \Big\} && \text{(1.7b)} \end{aligned}$$

where $\mathcal{Q}_{m,n}$ denotes the set of polynomial perturbations [13]. The polynomials of the set $\mathcal{Q}_{m,n}$ have arbitrary coefficients, which pass to the polynomials of the set $\mathcal{P}'_{m,n}$. If we use now the respective generalized resultants (Sylvester matrices) for each set in equation (1.7a), the following relation appears :

$$S_{\mathcal{P}'} = S_{\mathcal{P}} - S_{\mathcal{Q}} \tag{1.8}$$

It is clear that the exact GCD of a set of polynomials yields $S_{\mathcal{Q}} = O \Rightarrow \|S_{\mathcal{Q}}\|_F = 0$. Therefore, we may consider a polynomial as a good approximation of the exact GCD, if $\|S_{\mathcal{Q}}\|_F$ is close enough to zero. In the following, our intention is to find some bounds for $\|S_{\mathcal{Q}}\|_F$.

If we use the factorization of generalized resultants as described in (1.5), we will have :

$$\begin{aligned} S_{\mathcal{Q}} &= S_{\mathcal{P}} - [O_r | S_{\mathcal{P}'^c}] \cdot \Phi_v \;\Leftrightarrow \\ S_{\mathcal{Q}} \cdot \Phi_v^{-1} &= S_{\mathcal{P}} \cdot \Phi_v^{-1} - [O_r | S_{\mathcal{P}'^c}] \end{aligned} \tag{1.9}$$

where Φ_v^{-1} is the inverse of Φ_v. It is important to notice here that $\mathcal{P}'^c$ contains arbitrary parameters. We can select specific values for these parameters such as :

$$S_{\mathcal{P}} \cdot \Phi_v^{-1} - [O_r | S_{\mathcal{P}'^c}] = [S^{(r)} | O_{n+p-r}] \equiv \widehat{S} \tag{1.10}$$

Therefore, from equations (1.9) and (1.10) it follows :

$$\begin{aligned} S_{\mathcal{Q}} \cdot \Phi_v^{-1} &= \widehat{S} \\ S_{\mathcal{Q}} &= \widehat{S} \cdot \Phi_v \end{aligned}$$

and, since $\text{Cond}(\Phi_v) = \|\Phi_v\|_F \|\Phi_v^{-1}\|_F \geq n + p$, [8], we conclude with the following inequality :

$$\frac{\|\widehat{S}\|_F}{\|\Phi_v^{-1}\|_F} \leq \|S_{\mathcal{Q}}\|_F \leq \|\widehat{S}\|_F \|\Phi_v\|_F \tag{1.11}$$

If $v(s)$ has the same degree as the exact GCD of the set, the properties

$$\underline{\mathcal{S}_v} = \frac{\|\widehat{S}\|_F}{\|\Phi_v^{-1}\|_F} \quad \text{and} \quad \overline{\mathcal{S}_v} = \|\widehat{S}\|_F \|\Phi_v\|_F \tag{1.12}$$

characterizes the quality of the proximity of $v(s)$ to the exact GCD of the set $\mathcal{P}_{m,n}$ and we shall refer to them as the *minimum and maximum strength numbers* of $v(s)$ respectively.

These strength numbers are useful indicators for the evaluation of the strength of a given approximate GCD. More particularly, if $\underline{\mathcal{S}_v} \geq 1$, then the strength of the given approximation is bad and the opposite holds if $\overline{\mathcal{S}_v} < 1$. Normally, we prefer solutions with maximum strength number $\overline{\mathcal{S}_v} << 1$ or better close to the numerical software accuracy of the system. Otherwise, we have to solve the minimization problem (1.6) to find the actual strength. The advantage is that the computation of the strength numbers is straightforward and can give us information about the strength of a GCD before we go to an optimization method.

4.1 Computational Examples

The previous methods have been applied to many sets of polynomials. The final results using variable floating point, symbolic and hybrid operations are presented in tables 1 - 4. The following notation is used in the tables.

m	: the number of polynomials
n	: the maximum degree of the polynomials
p	: the second maximum degree of the polynomials
d	: the degree of the GCD
Tol	: numerical accuracy ε
Rel	: the numerical relative error
Strength	: the strength of the GCD
Dig	: the digits of software accuracy
Time	: the required time in seconds
Flops	: the required flops
Num	: numerical implementation
Sym	: symbolical implementation
Hybrid	: Hybrid implementation

In tables 1 & 2 the results of the following example sets are presented :
Example I : Two polynomials of degree 16 and 14 with integer coefficients of 8 digits and GCD degree 4, [2].
Example II : Eleven polynomials of degree 17 with integer coefficients of 2 digits and GCD degree 3, [28].
Example III : Two polynomials of degree 12 and GCD degree 6, (example 1, [37]). The roots of the polynomials spread on the circles of radius 0.5 and 1.5.

Comment : In tables 1 and 2 the tolerance (Tol) is fixed and the digits are variable. In tables 3 and 4 both tolerance and digits are fixed.

5. Numerical, Symbolical and Hybrid Behavior of the methods

All the sets of polynomials have been tested numerically and symbolically. Executing the programs symbolically, there are no floating point errors during the processes and the final result is the exact GCD of the polynomials. But the time required for symbolical computations is considerable. On the other hand, the floating-point operations and the accumulation of rounding errors force us to use accuracies on which the GCD is dependent. Different accuracies may lead to different GCDs. This is the main disadvantage of numerical methods. However, the required time is less than the corresponding time of symbolical methods. The use of the partial SVD [26, 35] can reduce the execution time of ERES, MRMP and SMP methods.

Having tested thoroughly, the ERES, RERE, MRERE, SMP methods computing the GCD of several sets of polynomials, we have reached the following conclusions about the behavior of the methods :

(a) The ERES method behaves very well in Hybrid mode, since it produces accurate results fast enough. The method becomes slower in the case of small sets of polynomials of high degree.

(b) The MRERE methods behave very well in numerical mode for various kinds of sets of polynomials and especially in large sets of polynomials of high degree, but lose their speed, when using symbolic type of operations.

(c) The RERE methods demand significantly more flops in comparison with MRERE methods and their complexity makes them inefficient methods.

(d) The MRMP method demands much more time and flops than the SMP method without producing better results.

(e) It seems that for large sets of linearly depended polynomials the Hybrid ERES and the Numerical MRERE yield better results in ac-

Table 1.1. Comparison of Algorithms : $Tol = 10^{-16}$

Example		*ERES*	*RERE (LU)*		*MRERE (LU)*		*RERE (QR)*	
		Hybrid	Num	Sym	Num	Sym	Num	Sym
I	Dig	16	32	-	32	-	35	-
	Rel	0	$0.50 \cdot 10^{-24}$	0	$0.50 \cdot 10^{-24}$	0	$0.13 \cdot 10^{-24}$	0
	Strength	0	$0.20 \cdot 10^{-22}$	0	$0.20 \cdot 10^{-22}$	0	$0.37 \cdot 10^{-23}$	0
	Time	1.842	0.020	0.120	0.081	0	0.060	0.270
	Flops	162140	14400	-	9790	-	32400	-
II	Dig	16	24	-	25	-	25	-
	Rel	0	$0.90 \cdot 10^{-21}$	0	$0.12 \cdot 10^{-19}$	0	$0.40 \cdot 10^{-21}$	0
	Strength	0	$0.21 \cdot 10^{-20}$	0	$0.18 \cdot 10^{-19}$	0	$0.24 \cdot 10^{-20}$	0
	Time	1.342	0.260	2.190	0.161	1.432	0.881	4.513
	Flops	112437	176000	-	87529	-	362667	-
III	Dig	16	18	-	18	-	19	-
	Rel	0	$0.68 \cdot 10^{-17}$	0	$0.68 \cdot 10^{-17}$	0	$0.13 \cdot 10^{-17}$	0
	Strength	0	$0.32 \cdot 10^{-16}$	0	$0.32 \cdot 10^{-16}$	0	$0.66 \cdot 10^{-17}$	0
	Time	2.794	0.010	0.340	0.007	0.234	0.030	0.793
	Flops	162140	6912	-	4759	-	16128	-

Example I : $m = 2$, $n = 16$, $p = 14$, $d = 4$
Example II : $m = 11$, $n = 17$, $p = 17$, $d = 3$
Example III : $m = 2$, $n = 12$, $p = 12$, $d = 6$

Table 1.2. Comparison of Algorithms : $Tol = 10^{-16}$

Example		*MRERE (QR)*		*MRMP*	*MP*	*SS*
		Num	Sym	Hybrid	Hybrid	Num
I	Dig	35	-	25	27	25
	Rel	$0.13 \cdot 10^{-24}$	0	$0.26 \cdot 10^{-13}$	$0.19 \cdot 10^{-13}$	$2.90 \cdot 10^{-12}$
	Strength	$0.37 \cdot 10^{-23}$	0	$0.31 \cdot 10^{-6}$	$0.96 \cdot 10^{-11}$	$0.2 \cdot 10^{-1}$
	Time	0.050	0.169	0.050	0.060	0.985
	Flops	19580	-	267080	121314	91898532
II	Dig	25	-	20	21	20
	Rel	$0.95 \cdot 10^{-21}$	0	$0.25 \cdot 10^{-17}$	$0.80 \cdot 10^{-17}$	$6.0 \cdot 10^{-19}$
	Strength	$0.15 \cdot 10^{-20}$	0	$0.56 \cdot 10^{-16}$	$0.33 \cdot 10^{-16}$	$5.51 \cdot 10^{-14}$
	Time	0.381	2.178	3.395	0.861	5.360
	Flops	175058	-	761725	56664	$4.1 \cdot 10^{9}$
III	Dig	19	-	19	21	20
	Rel	$0.13 \cdot 10^{-17}$	0	$0.65 \cdot 10^{-12}$	$0.59 \cdot 10^{-17}$	$6.9 \cdot 10^{-19}$
	Strength	$0.66 \cdot 10^{-17}$	0	$0.38 \cdot 10^{-11}$	$0.28 \cdot 10^{-16}$	$3.35 \cdot 10^{-18}$
	Time	0.020	0.468	2.835	0.290	0.579
	Flops	9517	-	126720	50832	24082500

Example I : $m = 2$, $n = 16$, $p = 14$, $d = 4$
Example II : $m = 11$, $n = 17$, $p = 17$, $d = 3$
Example III : $m = 2$, $n = 12$, $p = 12$, $d = 6$

Table 1.3. Comparison of Algorithms : Tol = 10^{-16}, Dig = 32

		ERES	*RERE(LU)*	*MRERE(LU)*	*RERE(QR)*	*MRERE(QR)*	*MRMP*	*MP*	*SS*
		Hybrid	Numerical	Numerical	Numerical	Numerical	Hybrid	Hybrid	Numerical
A	Rel	$3.31 \cdot 10^{-31}$	$3.82 \cdot 10^{-30}$	$2.94 \cdot 10^{-31}$	$1.35 \cdot 10^{-29}$	$3.36 \cdot 10^{-30}$	$2.08 \cdot 10^{-31}$	$2.49 \cdot 10^{-31}$	$2.15 \cdot 10^{-31}$
	Strength	$1.25 \cdot 10^{-27}$	$6.11 \cdot 10^{-26}$	$7.70 \cdot 10^{-27}$	$3.30 \cdot 10^{-25}$	$5.56 \cdot 10^{-26}$	$2.60 \cdot 10^{-26}$	$3.28 \cdot 10^{-26}$	$8.95 \cdot 10^{-27}$
	Time	0.094	0.016	0.016	0.031	0.015	0.125	0.063	0.203
	Flops	471	2584	1447	5167	2097	9167	2480	5445000
B	Rel	$1.30 \cdot 10^{-31}$	$4.56 \cdot 10^{-26}$	$2.01 \cdot 10^{-28}$	$5.56 \cdot 10^{-27}$	$3.80 \cdot 10^{-26}$	$1.82 \cdot 10^{-26}$	$1.53 \cdot 10^{-27}$	$6.12 \cdot 10^{-31}$
	Strength	$4.26 \cdot 10^{-27}$	$1.49 \cdot 10^{-23}$	$6.42 \cdot 10^{-24}$	$1.16 \cdot 10^{-21}$	$1.44 \cdot 10^{-21}$	$1.66 \cdot 10^{-21}$	$1.01 \cdot 10^{-22}$	$4.37 \cdot 10^{-26}$
	Time	0.141	0.047	0.031	0.172	0.063	0.937	0.156	1.0
	Flops	900	20667	11894	41334	20603	73334	2480	$1.2 \cdot 10^{9}$
C	Rel	$3.82 \cdot 10^{-32}$	$2.43 \cdot 10^{-28}$	$1.78 \cdot 10^{-29}$	$1.98 \cdot 10^{-23}$	$5.11 \cdot 10^{-23}$	$3.16 \cdot 10^{-24}$	$2.14 \cdot 10^{-25}$	$2.02 \cdot 10^{-31}$
	Strength	$3.03 \cdot 10^{-26}$	$1.99 \cdot 10^{-23}$	$9.96 \cdot 10^{-25}$	$1.47 \cdot 10^{-18}$	$3.34 \cdot 10^{-18}$	$5.59 \cdot 10^{-19}$	$1.85 \cdot 10^{-20}$	$7.07 \cdot 10^{-26}$
	Time	0.110	0.094	0.032	0.375	0.172	1.516	0.203	2.375
	Flops	1476	40896	19355	81792	30334	126720	4575	$5.7 \cdot 10^{9}$

Example A : $m = 10$, $n = 5$, $p = 5$, $d = 2$
Example B : $m = 10$, $n = 10$, $p = 10$, $d = 2$
Example C : $m = 10$, $n = 15$, $p = 10$, $d = 3$

Table 1.4. Comparison of Algorithms : Tol = 10^{-16}, Dig = 32

		ERES	*RERE(LU)*	*MRERE(LU)*	*RERE(QR)*	*MRERE(QR)*	*MRMP*	*MP*	*SS*
		Hybrid	Numerical	Numerical	Numerical	Numerical	Hybrid	Hybrid	Numerical
D	Rel	$4.37 \cdot 10^{-32}$	0.927	0.928	0.927	0.928	$3.84 \cdot 10^{-27}$	$1.51 \cdot 10^{-18}$	$1.01 \cdot 10^{-30}$
	Strength	$6.10 \cdot 10^{-17}$	$4.53 \cdot 10^{-10}$	$1.71 \cdot 10^{-9}$	$2.93 \cdot 10^{-9}$	$3.79 \cdot 10^{-10}$	$5.026 \cdot 10^{-12}$	$3.36 \cdot 10^{-12}$	$1.48 \cdot 10^{-16}$
	Time	5.203	1.328	0.562	2.719	1.578	18.048	4.078	18.063
	Flops	3993	479167	204917	958334	372389	1145834	73684	$3.1 \cdot 10^{12}$
E	Rel	$4.31 \cdot 10^{-31}$	$0.14 \cdot 10^{-26}$	$0.96 \cdot 10^{-30}$	$0.13 \cdot 10^{-28}$	$0.29 \cdot 10^{-28}$	$0.95 \cdot 10^{-28}$	$0.23 \cdot 10^{-28}$	×
	Strength	$1.51 \cdot 10^{-27}$	$0.13 \cdot 10^{-26}$	$0.16 \cdot 10^{-26}$	$0.41 \cdot 10^{-26}$	$0.42 \cdot 10^{-27}$	$0.55 \cdot 10^{-23}$	$0.33 \cdot 10^{-25}$	×
	Time	4.390	39.81	2.75	190.19	11.64	32.92	12.62	×
	Flops	37266	10125000	1125291	20331000	2250582	8933088	19551	×
F	Rel	$4.39 \cdot 10^{-32}$	$0.14 \cdot 10^{-26}$	$0.96 \cdot 10^{-30}$	$0.13 \cdot 10^{-28}$	$0.29 \cdot 10^{-28}$	$0.93 \cdot 10^{-25}$	$0.55 \cdot 10^{-25}$	×
	Strength	$4.67 \cdot 10^{-27}$	$0.13 \cdot 10^{-26}$	$0.16 \cdot 10^{-26}$	$0.41 \cdot 10^{-26}$	$0.42 \cdot 10^{-27}$	$0.72 \cdot 10^{-22}$	$0.48 \cdot 10^{-21}$	×
	Time	4.225	96.20	3.22	198.64	11.39	51.35	24.91	×
	Flops	358875	9703225	485534	19488301	971068	7433284	1125376	×

Example D : $m = 15$, $n = 25$, $p = 25$, $d = 5$
Example E : $m = 50$, $n = 40$, $p = 40$, $d = 5$
Example F : $m = 50$, $n = 40$, $p = 40$, $d = 30$

ceptable time limits, number of digits, with negligible relative error and strength number.

A subtle point in the numerical calculations is that it is not always easy to specify the exact accuracy to get numerically the correct GCD. This can be overcome in symbolical implementation but the more polynomials and the higher degrees we have, the more time we need to compute the GCD. Thus, a combination of the above implementations can lead to an improvement in the overall performance of the algorithms. Of course the hybrid implementation depends on the nature of the algorithm. Not all algorithms can be benefit from a hybrid environment like ERES [5]. The conversion of the data to an appropriate type often leads to more accurate computations and thus less numerical errors.

In the following table we compare the algorithms computing the GCD of polynomials. We propose the most suitable algorithm for the computation of the GCD of two or several polynomials according to its stability, complexity and required computational time. In the following table, m denotes the number of polynomials and n the maximum degree.

Table 1.5. Comparison of Algorithms: Computation of the GCD of polynomials.

Method	m	n	*Stability*	*Decision*
ERES	two	high	stable	not proposed
RERE	two	high	stable	proposed
MRERE	two	high	stable	proposed
SMP	two	high	stable	not proposed
MRMP	two	high	stable	not proposed
ERES	several	high	stable	proposed
RERE	several	high	stable	not proposed
MRERE	several	high	stable	proposed
SMP	several	high	stable	proposed
MRMP	several	high	stable	not proposed

6. Conclusions

According to the comparison of algorithms that we made, we conclude with the following :

1 Matrix-based methods can handle many polynomials simultaneously, without resorting to the successive two at a time computations of the Euclidean or other pairwise based approaches. Although the computation of the GCD of a pair of polynomials, by using a Euclid-based algorithm, can be stable, it is not quite clear if such method can be generalised to a set of several polynomials

and how this generalisation affects the complexity of the algorithm and the accuracy of the produced results. The sequential computation of the GCD in pairs often leads to an excessive accumulation of numerical errors especially in the case of large sets of polynomials and obviously create erroneous results. On the other hand, matrix-based methods tend to have better performance and numerical stability in the case of large sets of polynomials.

2 The development of robust computational procedures for engineering type models always has to take into account that the models have certain accuracy and that it is meaningless to continue computations beyond the accuracy of the original data set. Therefore, it is necessary to develop proper numerical termination criteria that allow the derivation of *approximate* solutions to the GCD computation problem [20, 27]. In [19] the definition of the approximate GCD is considered as a distance problem in a projective space. The new distance framework given for the approximate GCD provides the means for computing optimal solutions, as well as evaluating the strength of ad-hoc approximations derived from different algorithms.

3 The combination of symbolic-numeric operations performed effectively in a mixture of numerical and symbolical steps can increase the performance of a matrix-based GCD algorithm. Generally, symbolic processing is used to improve on the conditioning of the input data, or to handle a numerically ill-conditioned subproblem, and numeric tools are used in accelerating certain parts of an algorithm, or in computing approximate outputs. The effective combination of symbolic and numerical operations depends on the nature of an algebraic method and the proper handling of the input data either as rational or floating-point numbers. Symbolic-numeric implementation is possible in software programming environments with symbolic-numeric arithmetic capabilities such as Maple, Mathematica, Matlab and others, which involve the efficient combination of exact (rational-symbolic) and numerical (floating-point) operations. This combination gives a different perspective in the way to implement an algorithm and introduces the notion of *hybrid computations*. The nature of the ERES method allows the implementation of a programming algorithm that combines in an optimal setup the symbolical application of rows transformations and shifting, and the numerical computation of an appropriate termination criterion, which can provide the required approximate solutions. This combination highlights

the hybridity of the ERES method and makes it the most suitable method for the computation of approximate GCDs.

4 Most of the methods and algorithms, which were described in the previous section, perform singular value decomposition (svd). The necessary information that we need from the svd often has to do with the smallest or the greatest singular value (ERES, MP). Therefore a partial singular value decomposition algorithm [35] can be applied in order to speed up the whole process.

The paper is focused on the development of matrix-based algorithms for the GCD problem of sets of several real univariate polynomials, which is considered a part of the fundamental problem of computing nongeneric invariants.

References

[1] Blankiship, B., *A new version of Euclid Algorithm*, Amer. Math. Mon., **70**, (1963), 742-744.

[2] Barnett, S. *Greatest common divisor of several polynomials*, Proc. Cambridge Phil. Soc. **70** (1971), 263-268.

[3] Barnett, S. *Greatest common divisor from Generalized Sylvester Matrices*, Proc. Cambridge Phil. Soc., **8**, (1980), 271-279.

[4] Chin, Corless and Corliss, *Optimization Strategies for the Approximate GCD Problem*, Proceedings Internat. Symp. Symbolic Algebraic Comput. (ISSAC'98), (1998), 228-235.

[5] Christou, D. and Mitrouli, M., *Estimation of the Greatest Common Divisor of many polynomials using hybrid computations performed by the ERES method*, Appl. Num. Anal. and Comp. Math., Vol. 2, No. 3, (2005), 293-305.

[6] Christou, D., Karcanias, N., Mitrouli, M. and Triantafyllou, D., *Numerical and symbolical comparison of resultant type, ERES and MP methods computing the Greatest Common Divisor of several polynomials*, Proc. European Control Conference ECC'07, Kos, Greece, (2007), 504-511.

[7] Corless, R. M., Gianni, P. M., Trager B. M. and Watt, S. M. *The Singular Value Decomposition for Polynomial Systems*, Proc. ISSAC'95, Quebec, Canada, (1995), 195-207.

[8] Datta, B.N., *Numerical Linear Algebra and Applications*, Brooks/Cole Publishing Company - ITP, (1995).

[9] Datta, B.N. *Numerical Methods for Linear Control Systems*, Elsevier Academic Press, (2004).

[10] Diaz-Toca, G. M. and Gonzalez-Vega, L., *Computing greatest common divisors and squarefree decompositions through matrix methods: The parametric and approximate cases*, Linear Algebra and its Appl., **412**, (2006), 222-246.

[11] Emiris, I.Z., Galligo, A. and Lombardi, H., *Certified approximate univariate GCDs*, J. Pure and Applied Algebra, 117-118, (1997), 229-251.

[12] Golub, G.H. and Van Loan, C.F., *Matrix Computations*, Sec. Edition, The John Hopkins University Press, Baltimore, London, (1989).

[13] Fatouros, S., Karcanias, N., *Resultant properties of gcd of many polynomials and a factorization representation of gcd*, Int. Journ. Control, **76** (2003), 1666-1683.

[14] Hirsch, M. W. and Smale, S., *Differential Equations, Dynamic Systems and Linear Algebra*, Academic Press, New York, (1974).

[15] Kailath, T., *Linear Systems*, Prentice Hall, Inc, Englewood Cliffs, New Jersey, (1980).

[16] Karcanias, N., and Mitrouli, M., *Approximate algebraic computations of algebraic invariants*, Symbolic methods in control systems analysis and design, IEE Control Engin. Series, **56**, (1999), 162-168.

[17] Karcanias, N., Giannakopoulos, C., and Hubard, M., *Almost zeros of a set of polynomials of* $\Re[s]$, Int. J. of Control, **38**, (1983), 1213-1238.

[18] Karcanias, N., *Invariance properties and characterisation of the greatest common divisor of a set of polynomials*, Int. Journ. Control, **46** (1987), 1751-1760.

[19] Karcanias, N., Fatouros, S., Mitrouli, M., and Halikias, G.H., *Approximate greatest common divisor of many polynomials, generalised resultants, and strength of approximation*, Computers & Mathematics with Applications, **51**, Issue 12, (2006), 1817-1830.

[20] Karcanias, N. and Mitrouli, M., *A Matrix Pencil Based Numerical Method for the Computation of the GCD of Polynomials*, IEEE Trans. Autom. Cont., **39**, (1994), 977-981.

[21] Karcanias, N. and Mitrouli, M., *Numerical Computation of the Least Common Multiple of a Set of Solynomials*, Reliable Computing, **6**, Issue 4, (2000), 439-457.

[22] Karcanias, N. and Mitrouli, M., *System theoretic based characterisation and computation of the least common multiple of a set of polynomials*, Linear Algebra and its Applications, **381**, (2004), 1-23.

[23] Karcanias, N., Mitrouli, M. and Triantafyllou, D., *Matrix Pencil methodologies for computing the greatest common divisor of polynomials: Hybrid algorithms and their performance*, Int. Journ. of Control, **79**, No 11, (2006), 1447-1461.

[24] Karmarkar, N. and Lakshman, Y.N., *Approximate polynomial greatest common divisors and nearest singular polynomials*, ISSAC'96, ACM Press, (1996), 35-39.

[25] Li, T.Y. and Zeng, Z., *A rank-revealing method with updating, downdating and applications*, SIAM Journal on Matrix Analysis and Applications, to appear, (*www.neiu.edu/˜zzeng/*).

[26] Marco, A. and Martinez, J.J., *A new source of structured singular value decomposition problems*, Electronic Trans. on Num. Anal., **18**, (2004), 188-197.

[27] Mitrouli, M., Karcanias, N., *Computation of the GCD of polynomials using Gaussian transformations and shifting*, Int. Journ. Control, **58** (1993), 211-228.

[28] Mitrouli, M., Karcanias, N. and Koukouvinos, C., *Further numerical aspects of the ERES algorithm for the computation of the Greatest Common Divisor of polynomials and comparison with other existing methodologies*, Utilitas Mathematica, **50** (1996), 65-84.

[29] Noda, M. and Sasaki, T., *Approximate GCD and its applications to ill-conditioned algebraic equations*, Jour. of Comp. and Appl. Math., **38** (1991), 335-351

[30] Pace, I.S. and Barnett, S., *Comparison of algorithms for calculation of g.c.d of polynomials*, Int. J. Systems Sci., Vol. 4, No. 2, (1973), 211-226.

[31] Qiu, W., Hua, Y., and Abed-Meraim, K., *A subspace method for the computation of the GCD of polynomials*, Automatica, Vol. 33, No. 4, (1997), 741-743.

[32] Rosenbrock, H., *State Space and Multivariable Theory*, Nelson, London, (1970).

[33] Rupprecht, D., *An algorithm for computing certified approximate GCD of n univariate polynomials*, J. of Pure and Applied Algebra, Vol. 139, (1999), 255-284.

[34] Triantafyllou D. and Mitrouli M., *Two Resultant Based Methods Computing the Greatest Common Divisor of Two Polynomials*, Lecture Notes in Computer Science, Vol. 3401, (2005), 519-526.

[35] Van Huffel, S. and Vandewalle, J., *An efficient and reliable algorithm for computing the singular subspace of a matrix, associated with its smallest singular values*, J. of Comp. and Applied Math., **19**, (1987), 313-330.

[36] Wonham, W., *Linear Multivariable Control: A Geometric Approach*, Springer Verlag, New York Sec. Edit, (1984).

[37] Zeng, Z., *The approximate GCD of inexact polynomials, Part I: A univariate algorithm*, to appear, (*www.neiu.edu/˜zzeng/*).