THE SNAP PACKAGE FOR ARITHMETIC WITH NUMERIC POLYNOMIALS

C.-P. JEANNEROD
LIP, Ecole Normale Supérieure de Lyon,
46 Allée d'Italie, 69364 Lyon Cedex 07, France
E-mail: Claude-Pierre.Jeannerod@ens-lyon.fr

G. LABAHN
Department of Computer Science, University of Waterloo,
Ontario N2L 3G1 Canada
E-mail: glabahn@scg.math.uwaterloo.ca

In this paper we describe the SNAP package, a new MAPLE package for algebraic manipulation of univariate numeric polynomials. The package includes commands for quotient, remainder, greatest common divisors and related operations. We discuss the methods used in the package along with the key issues that were encountered during the implementation.

1 Introduction

There are a number of application areas which can be formally described in terms of algebraic operations on polynomials and matrix polynomials such as division, remainders and greatest common divisors. Areas which make use of such an approach include control theory, linear systems theory and signal processing[14]. While an algebraic polynomial formalism is extremely useful for theoretical studies of properties, it has drawbacks when one tries to take advantage for computations. The primary problem is that many applications use inexact data and finite precision computations while the formalism assumes exact information and error-free computations[16,20].

In this paper we describe the SNAP (Symbolic-Numeric Algorithms for Polynomials) package for computing with polynomials having inexact coefficients. This package is a first attempt to provide the standard functionalities for inexact polynomials that exist for exact polynomials, including the taking of quotients and remainders, determining if two polynomials are relatively prime and finding greatest common divisors (GCDs). The package is included in the coming release of the MAPLE computer algebra system.

Several algorithms exist for performing such operations, in particular for computing the various notions of the so-called approximate GCDs. These algorithms include modifications of the Euclidean algorithm[10,16,19], optimization techniques[5,12,13], subresultant-based algorithms[6,7], and polynomial root approximation[17]. For the most part, the previously mentioned algorithms resort to either infinite or adaptive precision. In our case, the software we present works under the customary model of numerical computation using fixed precision floating-point arithmetic and the underlying algorithms benefit from round-off error analysis which ensures numerical stability[9].

This approach, based on estimating the distance to the closest pair of polynomials having a common factor[3], also allows for computations that typically run in quadratic time. Also, the two approximate GCDs that we focus on in the package (quasi-GCD[19] and ϵ-GCD[3]) are certified in the sense of Emiris *et al* [7].

The remainder of the paper is organized as follows. The next section gives a description of the commands in the SNAP package. Since the numerical computation of GCDs plays such a significant role in the package, we summarize in Section 3 the Beckermann-Labahn algorithm, which is the main tool used for this computation and for a number of commands inside the package. Section 4 then gives some details of the implementation used while Section 5 gives some of the practical features of the package. The paper ends with a conclusion along with topics for future research.

2 Overview of the SNAP package

The SNAP package has been designed for univariate polynomials with real floating-point coefficients. The commands it offers are described below, where (a, b) denotes a pair of such numeric polynomials:

- `AreCoprime`: coprimeness test for (a, b) when known up to a given ϵ;
- `DistanceToCommonDivisors`: estimate the distance between (a, b) and the set of polynomial pairs with at least one nontrivial common divisor;
- `DistanceToSingularPolynomials`: estimate the distance between a and the set of polynomials with at least one multiple root;
- `EpsilonGCD`: compute a polynomial g and a quantity ϵ such that g is an ϵ-GCD for (a, b);
- `EuclideanReduction`: return a degree reduced basis that is numerically equivalent to the basis (a, b);
- `IsSingular`: decide whether a has at least one mutiple root when known up to a given ϵ;

- `QuasiGCD`: compute a polynomial g and a quantity ϵ such that g is a quasi-GCD with precision ϵ for (a,b);
- `Quotient`: compute the quotient of a divided by b;
- `Remainder`: compute the remainder of a divided by b.

A typical use of a SNAP command is illustrated through the simple example below (with 10 digits) where one investigates the relative primeness of two polynomials up to a tolerance of 0.5 and 0.1:

```
> with(SNAP):
> a := 0.1*z^2+1.5*z-0.2:
> b := 0.2*z^3+0.15:
> AreCoprime(a,b,z,0.5);
```

false

```
> AreCoprime(a,b,z,0.1);
```

true

Since for the latter tolerance the above polynomials are coprime, their Bézout coefficients for the associated linear diophantine equation can also be obtained using an optional parameter:

```
> AreCoprime(a,b,z,0.1,output='BC');
```

true,

$$[-0.871004100817765581z^2 - 0.112232907263963163z - 0.05851459268,\\ 0.435502050408882734z + 6.588647210]$$

3 Solving polynomial Diophantine equations numerically

Let $a(z)$, $b(z)$ be two univariate polynomials over $\mathbb{C}$ with degrees m, n. Except for `Quotient` and `Remainder`, the majority of the current functions in the SNAP package are based on the following approach, developed by Beckermann and Labahn[2,3]: compute polynomial solutions $u(z)$, $v(z)$, $\underline{u}(z)$, $\underline{v}(z)$ to the Diophantine equations

$$a(z)v(z) + b(z)u(z) = 1, \quad \deg u < m, \quad \deg v < n, \tag{1}$$

$$a(z)\underline{v}(z) + b(z)\underline{u}(z) = z^{m+n-1}, \quad \deg \underline{u} < m, \quad \deg \underline{v} < n, \tag{2}$$

in a fast and numerically stable way. Here, "fast" means arithmetic complexity in $O((m+n)^2)$ and "stable" means weakly stable in the sense of Bunch[4].

This approach provides a numerical coprimeness test together with Bézout coefficients. Additionally, a solution $(u, v, \underline{u}, \underline{v})$ to Equations (1), (2) yields a sharp estimation of the distance $\epsilon(a, b)$ to the set of polynomial pairs with a common root. Indeed, denoting by $|| \;||$ the 1-norm for the space of matrix polynomials over $\mathbb{C}$, one has

$$\epsilon(a,b) = \min\{||(a-a^*, b-b^*)|| : \deg \gcd(a^*, b^*) > 0,\ \deg a^* \le m, \deg b^* \le n\}.$$

If we let $S(a, b)$ be the Sylvester matrix associated with $a(z)$, $b(z)$ then it can be shown[2] that

$$\frac{1}{||S(a,b)^{-1}||} \le \frac{1}{\kappa} \le \epsilon(a,b) \quad \text{where} \quad \kappa = \left|\left| \begin{bmatrix} v & \underline{v} \\ u & \underline{u} \end{bmatrix} \right|\right|. \tag{3}$$

This improves upon the well known lower bound $1/||S(a,b)^{-1}||$ for $\epsilon(a, b)$.

Of course, a solution to Equations (1), (2) may not exist. However, the COPRIME[3] algorithm of Beckermann and Labahn always returns a reduced pair of polynomials from which one often can deduce a solution of (under additional assumptions) some approximate GCDs.

3.1 The algorithm COPRIME

We may assume[3,11] w.l.o.g. that $m > n$ and that the input numerical polynomials have been scaled to satisfy $1/2 \le ||(a, b)|| \le 1$. Starting with $(a^{(0)}, b^{(0)}, c^{(0)}) = (a(z), b(z), -z^{n-1})$, the COPRIME algorithm computes for $k \le m$ a numerical polynomial remainder sequence $(a^{(k)}, b^{(k)}, c^{(k)})$ by means of so-called **unimodular reductions** $U^{(k)}(z) \in \mathbb{C}[z]^{2\times 2}$ **of order** k together with **associated vectors** $\underline{U}^{(k)}(z) \in \mathbb{C}[z]^{2\times 1}$. More precisely,

$$(a^{(k)}, b^{(k)}) = (a, b) \cdot U^{(k)}, \quad c^{(k)} = (a, b) \cdot \underline{U}^{(k)} - z^{n+k-1} \tag{4}$$

with $U^{(k)}$ unimodular and where, for $k \ge 1$,

$$n \ge \deg a^{(k)} = m - k > \deg b^{(k)}, \quad \deg c^{(k)} < m - k - 1,$$

$$\deg U^{(k)} \le \begin{bmatrix} n-m+k-1 & n-m+k \\ k-1 & k \end{bmatrix}, \quad \deg \underline{U}^{(k)} \le \begin{bmatrix} n-m+k-1 \\ k-1 \end{bmatrix}.$$

The two main features of this Euclidean-like reduction process are summarized in the theorem below.

Theorem 3.1 *The pairs (a, b) and $(a^{(k)}, b^{(k)})$ define two different bases of the same ideal with $a^{(k)}$ of smaller degree than a. When $k = m$, a solution $(u, v, \underline{u}, \underline{v})$ to Equations (1), (2) is given by*

$$\begin{bmatrix} v \\ u \end{bmatrix} = \frac{1}{a^{(m)}(0)} U^{(m)} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} \underline{v} \\ \underline{u} \end{bmatrix} = \underline{U}^{(m)}. \tag{5}$$

In finite precision arithmetic, some remainders defined by (4) may correspond to ill-conditioned subproblems[3]. In order to ensure numerical stability, such remainders are discarded by a "look-ahead" strategy: one jumps from $(a^{(k)}, b^{(k)}, c^{(k)})$ to the first $(a^{(k+s)}, b^{(k+s)}, c^{(k+s)})$ with $s \geq 1$ such that

$$|\det U^{(k+s)}(0)| > \tau \quad \text{and} \quad ||\underline{U}^{(k+s)}|| < 1/\tau. \tag{6}$$

Here, τ is a threshold parameter of order the cubic root of the machine precision and $(a^{(k+s)}, b^{(k+s)})$ is the first basis after $(a^{(k)}, b^{(k)})$ that is **numerically well-behaved** in the sense of (6). The set $\mathcal{A}$ of the indices of all such accepted bases clearly satisfies $\mathcal{A} \subset \{0, 1, 2, \ldots, m\}$.

On the other hand, $U^{(k+s)}$ and $\underline{U}^{(k+s)}$ are determined from $a^{(k)}$, $b^{(k)}$, $c^{(k)}$, $U^{(k)}$, $\underline{U}^{(k)}$ in a numerically stable manner as follows. Consider the $2s \times 2s$ complex matrix

$$M_s^{(k)} = \left[\begin{array}{cccc|ccc}
a_{m-k}^{(k)} & & & & & 0 & \\
a_{m-k-1}^{(k)} & a_{m-k}^{(k)} & & & b_{m-k-1}^{(k)} & \ddots & \\
\vdots & & \ddots & & \vdots & \ddots & 0 \\
\vdots & & & a_{m-k}^{(k)} & \vdots & & b_{m-k-1}^{(k)} \\
\vdots & & & \vdots & \vdots & & \vdots \\
a_{m-k-2s+1}^{(k)} & \cdots & \cdots & a_{m-k-s}^{(k)} & b_{m-k-2s+1}^{(k)} & \cdots & b_{m-k-s}^{(k)}
\end{array}\right] \tag{7}$$

and solve the three linear systems $M_s^{(k)} x_i = y_i$ with

$$y_1 = (0, \ldots, 0, 1)^T, \quad y_2 = -\left(b_{m-k-j}^{(k)}\right)_{j=1,..,2s}, \quad y_3 = \left(c_{m-k-j}^{(k)}\right)_{j=1,..,2s}. \tag{8}$$

Setting up the 2×2 and 2×1 polynomial matrices

$$U^{(k,k+s)}(z) = \begin{bmatrix} z^{s-1} \cdots z^1 \; z^0 & 0 \;\cdots\cdots\; 0 \\ 0 \;\cdots\cdots\; 0 & z^{s-1} \cdots z^1 \; z^0 \end{bmatrix} \cdot \left(x_1, x_2\right) + \begin{bmatrix} 0 & 0 \\ 0 & z^s \end{bmatrix}$$

and

$$\underline{U}^{(k,k+s)}(z) = \begin{bmatrix} z^{s-1} \cdots z^1 \; z^0 & 0 \;\cdots\cdots\; 0 \\ 0 \;\cdots\cdots\; 0 & z^{s-1} \cdots z^1 \; z^0 \end{bmatrix} \cdot x_3,$$

one finally obtains[3]

$$U^{(k+s)} = U^{(k)} \cdot U^{(k,k+s)} \quad \text{and} \quad \underline{U}^{(k+s)} = z^s \underline{U}^{(k)} - U^{(k)} \cdot \underline{U}^{(k,k+s)}.$$

When $k + s \in \mathcal{A}$, additional scaling ensures that $1/2 \leq ||U^{(k+s)}|| \leq 1$. The resulting algorithm, called COPRIME, is given in Table 1.

Method:	Construct (scaled) unimodular reductions $U^{(k)}$ of (a,b) of order $k \in \mathcal{A}$ together with associated vectors $\underline{U}^{(k)}$ for $1 \leq k \leq m$.
Input:	Two polynomials a, b with $\deg a = m > \deg b$. A stability parameter τ.
Output:	If $m \in \mathcal{A}$: RETURN $1/\kappa$ given by (3) and (5). If $m \notin \mathcal{A}$: message, for $1/\kappa$ does not exist or is "too small".
Initialization:	$k = 0$, $\mathcal{A} = \{\}$, $U^{(0)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and $\underline{U}^{(0)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$.
Single Step:	For $s = 1, 2, \ldots$: Compute $U^{(k,k+s)}$, $\underline{U}^{(k,k+s)}$ by solving linear systems (7-8) (if $\det M_s^{(k)} = 0$ then increase s and restart). Rescale $U^{(k)} \cdot U^{(k,k+s)}$ to obtain $U^{(k+s)}$.
Exit s–loop:	if (6) holds. In this case $k \leftarrow k + s$ and $\mathcal{A} \leftarrow \mathcal{A} \cup \{k\}$.
Exit ALGO:	If $k + s = m$.

Table 1. The algorithm COPRIME

Theorem 3.2 *The algorithm COPRIME is weakly stable and requires in most cases $O((m+n)^2)$ flops.*

Quadratic complexity can be achieved mainly because for most dense polynomial pairs (a, b) only small jumps ($s \leq 3$) and thus only small linear systems are typically encountered. Note further that the products of (4) can be replaced with the "shorter" products

$$(a^{(k+s)}, b^{(k+s)}) = (a^{(k)}, b^{(k)}) \cdot U^{(k,k+s)}, \quad c^{(k+s)} = z^s c^{(k)} - (a^{(k)}, b^{(k)}) \cdot \underline{U}^{(k,k+s)}.$$

When $m \in \mathcal{A}$ we conclude that (a, b) are numerically coprime up to perturbations of order $\epsilon < 1/\kappa$. When $m \notin \mathcal{A}$ then the last accepted basis $(a^{(k)}, b^{(k)})$ may still provide various approximate GCDs.

3.2 Approximate GCDs from the last accepted basis

Recall that a polynomial g is a **quasi-GCD**[19] **with precision** ϵ for a, b if there exist polynomials u_1, v_1, u_2, v_2 such that

$$||(a,b) - g(u_2, v_2)|| < \epsilon, \quad ||av_1 + bu_1 - g|| < \epsilon ||g||, \quad \deg u_1 < m, \quad \deg v_1 < n.$$

Also, g is an **ϵ-GCD**[3] for a, b if there exist polynomials $\tilde{a}$, $\tilde{b}$ with degrees at most m, n such that $||(a,b)-(\tilde{a},\tilde{b})|| \leq \epsilon$ and $g = \gcd(\tilde{a},\tilde{b})$ has maximal degree.

All the computations in the COPRIME algorithm are done using finite precision arithmetic. As such there are residual error polynomials $\alpha^{(k)}$ and $\beta^{(k)}$ so that

$$(a,b) = U^{(k)} \cdot (a^{(k)}, b^{(k)}) + (\alpha^{(k)}, \beta^{(k)})$$

with $\deg a^{(k)} = m-k > \deg b^{(k)}$. Additionally, let $\rho_l(a,b)$ be the minimum of the set of all products $||(a,b)|| \cdot ||(g_a, g_b)^T||$ where the polynomial pair (g_a, g_b) is such that $\deg g_a < l$, $\deg g_b < l$ and

$$z^n a(z) g_a(z) + z^m b(z) g_b(z) = z^{m+n+l-1} + O(z^{m+n-1})_{z\to\infty}. \tag{9}$$

Beckermann and Labahn point out that when the last accepted basis has "small" errors then it can often still be used for computing either a quasi-GCD or an ϵ-GCD.

Theorem 3.3 *Let $U^{(k)}$ be the last well-behaved unimodular reduction computed by the algorithm COPRIME. Then*

(a) *If $||b^{(k)}|| + ||(\alpha^{(k)}, \beta^{(k)})|| < \epsilon\,|\det U^{(k)}(0)|/12$ with $0 < \epsilon < 1/6$ then $a^{(k)}$ is a quasi–GCD with precision ϵ.*

(b) *If $2||b^{(k)}|| + (2 + 4\rho_{n-m+2k}(a,b)) \cdot ||(\alpha^{(k)}, \beta^{(k)})|| \leq \epsilon\,|\det U^{(k)}(0)|$ and $|\det U^{(k)}(0)| > 4\rho_{n-m+2k}(a,b) \cdot \epsilon$ then $a^{(k)}$ is an ϵ-GCD.*

4 Implementation details

4.1 Fast and stable update of QR factorizations

It may happen that the look-ahead procedure has several jumps of order $s = O(m)$. In this case, successively computing QR decompositions of $M_1^{(k)}, \ldots, M_s^{(k)}$ by the classical method costs $\sum_{i=1}^{s} i^3$ flops, resulting in an $O((m+n)^4)$ algorithm. However, by taking advantage of the recursive structure of the matrix $M_s^{(k)}$ of (7) it is possible to reduce the above complexity to $O((m+n)^3)$. Indeed, we can determine the ith QR decomposition from the $(i-1)$th one in quadratic rather than cubic time[8]. This is precisely why such intermediate linear systems are solved via QR decomposition rather than say Gaussian elimination[1]. The details of such QR updates can be found in the SNAP user's guide[11].

4.2 Using MAPLE hardware floats

MAPLE has two floating point systems, hardware float and software float[15]. Software floats are used for extended precision arithmetic while hardware float makes use of double precision arithmetic available on all computers. Hardware floating point is much faster than software float, but is limited in its usage. Our implementation of COPRIME was done is such a way that it could run in either setting, making use of hardware floats where valid. We remark that the structures allowed inside procedures that can work with hardware floating point are limited (for example sets, lists and sequences are not allowed).

4.3 Determining $\rho_l(a, b)$ via least squares

In order to check the conditions of Theorem 3.3, we need to estimate $\rho_l(a, b)$ (see Equation (9)). Computing a 2-norm equivalent of $\rho_l(a, b)$ allows to reduce to the following least squares problem.

When $l \geq 0$, consider the $l \times 2l$ complex matrix

$$M = \left[\begin{array}{ccc|ccc} a_m & & & b_n & & \\ \vdots & \ddots & & \vdots & \ddots & \\ a_{m-l+1} & \cdots & a_m & b_{n-l+1} & \cdots & b_n \end{array}\right].$$

In the case where $l < 0$, it suffices to replace l with $-l$ and $a(z)$, $b(z)$ with $z^m a(z^{-1})$, $z^n b(z^{-1})$ respectively. We then compute the Moore-Penrose inverse M^+ of M by singular value decomposition[8]. We then take for $\rho_l(a, b)$ the product $||M^+(., 1)||_2 \cdot ||(a, b)||_2$ where $M^+(., 1)$ denotes the first column of M^+.

The SNAP commands `EpsilonGCD` and `QuasiGCD` are based on this approach.

5 Practical features of SNAP

5.1 Efficient coprimeness test

The SNAP package allows for numerical coprimeness to be detected efficiently for most inputs. Dense polynomials of degree of order 1000 can typically be handled by our implementation of COPRIME within a few minutes (on a Pentium III 800 MHz with 512MB of RAM, under Linux). This is illustrated in the table below. For comparison, we also give the time (in seconds) required to estimate $\epsilon(a, b)$ via setting up the Sylvester matrix $S(a, b)$ and solving the associated linear systems by LU decomposition (with the MAPLE command `LinearAlgebra[LinearSolve]`):

$m = n + 1$	Sylvester	COPRIME
50	0.33	0.34
100	1.30	1.34
250	8.6	8.7
500	43	33
10^3	307	126
$2 \cdot 10^3$	3360	515

5.2 Certified approximate GCDs

Another important feature of the package is the ability to compute approximate GCDs at the same cost as a coprimeness test. Such GCDs are certified by Theorem 3.3. As an illustration, we consider the following example due to Rupprecht[18]. Let

$$a(z) = (z^4 - 1.000001)(z^3 - 3.000001z + 0.99999999)$$

and

$$b(z) = (z^4 - 0.9999999)(z^4 - 3.0000003z - 2.9999999).$$

Calling `SNAP[EpsilonGCD](a,b,z)` with 10 digits (the default in MAPLE) yields the 0.0006481143605-GCD

$$\begin{aligned} z^4 + 5.198691325\,10^{-8}z^3 + 5.804634175\,10^{-7}z^2 \\ +9.873862873\,10^{-7}z - 1.000000584 \end{aligned}$$

after normalization of the leading coefficient.

5.3 Satisfactory answers from Euclidean reduction

When the conditions of Theorem 3.3 are not satisfied, the SNAP command `Euclideanreduction` proves in general to be very useful for two reasons: in most cases, the returned polynomial pair has much smaller degrees that the input; also, the first component can be seen as a satisfactory candidate for an approximate GCD. For example, let

$$a(z) = z^6 - 12.4z^5 + 62.53z^4 - 163.542z^3 + 232.9776z^2 - 170.69184z + 50.18112$$

and

$$b(z) = z^5 - 17.6z^4 + 118.26z^3 - 372.992z^2 + 538.3333z - 274.09272$$

Calling `SNAP[EuclideanReduction](a,b,z,tau=1e-8)` with 10 digits then yields the reduced pair

$$0.2500000000z^2 - 0.8750003765z + 0.6600005057,$$
$$-0.90726410^{-7}z + 0.133529610^{-6}.$$

Degrees have decreased from $(6,5)$ to $(2,1)$ but the second component has 1-norm of order $O(10^{-7})$ and should be interpreted as a zero remainder.

6 Concluding remarks

In this paper we have given a description of the SNAP package, a new MAPLE package for arithmetic with univariate numeric polynomials. This package is a first version and it is by no means complete with its primary focus being on the operations of quotient, remainder and GCDs. As such there are a number of additional commands that need to be investigated and implemented for future versions of the package. These include commands for numerical division of univariate polynomials, reduction of numeric rational expressions and others. Furthermore, commands that allow for numerically correct arithmetic with multivariate polynomials will also be needed.

In this paper, we have focused only on two types of approximate GCDs. The package would thus gain by the inclusion of the algorithms which follow an adaptive precision model, for example as found in Corless *et al* [5] and Karmarkar and Lakshman[13]. It would also be interesting to compare our available software with implementations of the methods of Noda and Sasaki[16], Pan[17] and Zarowski *et al* [21]. Because of these various definitions and algorithms for approximate GCDs, it may be of interest to provide a user with an automatic selection facility, depending on the problem to be solved.

Finally, the package would also gain significant use if it included algebraic operations for matrix polynomials, particularly in such application areas as control theory and linear systems theory. The delay in this aspect of the package has to do with the lack of numerically stable algorithms for computing such operations as numeric matrix greatest common divisor or even numeric matrix normal forms.

Acknowledgements

We would like to thank Bernhard Beckermann for providing some related MATLAB code, and Robert Corless and Lihong Zhi for their early helpful comments on the SNAP package.

References

1. B. Beckermann, The stable computation of formal orthogonal polynomials, *Numerical Algorithms* **11** (1996) 1-23.
2. B. Beckermann and G. Labahn, When are two numerical polynomials relatively prime? *Journal of Symbolic Computation* **26** (1998) 677-689.
3. B. Beckermann and G. Labahn, A fast and numerically stable Euclidean-like algorithm for detecting relatively prime numerical polynomials, *Journal of Symbolic Computation* **26** (1998) 691-714.
4. J.R. Bunch, The weak and strong stability of algorithms in numerical linear algebra, *Lin. Alg. Appl.* **88-89** (1987) 49-66.
5. R.M. Corless, P.M. Gianni, B.M. Trager and S.M. Watt, The singular value decomposition for polynomial systems, Proceedings ISSAC'95, ACM Press (1995) 195-207.
6. I. Z. Emiris, Symbolic-numeric algebra for polynomials. Survey available at `http://www-sop.inria.fr/galaad/emiris` (1997).
7. I. Z. Emiris, A. Galligo and H. Lombardi, Certified approximate univariate GCDs, *J. Pure and Applied Algebra* **117** (1997) 229-251.
8. G.H. Golub and C. Van Loan, Matrix computations, Johns Hopkins University Press (1989).
9. N.J. Higham, *Accuracy and Stability of Numerical Algorithms* (SIAM, Philadelphia, 1996).
10. V. Hribernig and H.J. Stetter, Detection and validation of clusters of polynomial zeros, J. Symbol. Comput. 24 (1997) 667-681.
11. C.P. Jeannerod and G. Labahn, SNAP User's Guide (2002).
12. N. Karmarkar and Y.N. Laksmann, Approximate polynomial greatest common divisors and nearest singular polynomials, Proceedings ISSAC'96, ACM Press (1996) 35-43.
13. N. Karmarkar and Y.N. Laksmann, On approximate GCDs of univariate polynomials, *Journal of Symbolic Computation* **26** (1998) 653-666.
14. T. Kailath, Linear systems, Prentice–Hall (1980).
15. M.B. Monagan, K.O. Geddes, K.M. Heal, G. Labahn, S.M. Vorkoetter, J. McCarron and P. DeMarco, *Maple 7 Programming Guide,* Toronto: Waterloo Maple Inc., 2001.
16. M.-T. Noda & T. Sasaki, Approximate GCD and its applications to ill–conditioned algebraic equations, *J. Comput. Appl. Math.* **38** (1991) 335-351.
17. V.Y. Pan, Numerical computations of a polynomial GCD and extensions. Technical Report 2969, INRIA, Sophia-Antipolis (1996).
18. D. Rupprecht, Eléments de géométrie algébrique approchée: étude du PGCD et de la factorisation, Thèse de l'Université de Nice-Sophia Antipolis (2000).
19. A. Schönhage, Quasi–GCD computations, *J. Complexity* **1** (1985) 118-137.
20. H. Stetter, Numerical Polynomial Algebra: Concepts and Algorithms, ATCM 2000, Proceed. 5th Asian Technology Conf. in Math. (Eds.: W.-Ch. Yang, S.-Ch. Chu, J.-Ch. Chuan), 22-36, ATCM Inc. USA (2000).
21. 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)** (2000) 3042-3051.