QR Factoring to Compute the GCD of Univariate Approximate Polynomials

Robert M. Corless, Stephen M. Watt, and Lihong Zhi

***Abstract*—We present a stable and practical algorithm that uses QR factors of the Sylvester matrix to compute the greatest common divisor (GCD) of univariate approximate polynomials over $\mathbb{R}[x]$ or $\mathbb{C}[x]$. An approximate polynomial is a polynomial with coefficients that are not known with certainty. The algorithm of this paper improves over previously published algorithms by handling the case when common roots are near to or outside the unit circle, by splitting and reversal if necessary. The algorithm has been tested on thousands of examples, including pairs of polynomials of up to degree 1000, and is now distributed as the program QRGCD in the SNAP package of Maple 9.**

***Index Terms*—Greatest common divisor, QR-factoring, Sylvester matrix.**

I. Introduction

For an introduction to and motivation for the problem studied in this paper, and a brief survey of recent results and methods, see [1]. In short, the problem of finding a polynomial greatest common divisor (GCD), when the coefficients of the polynomials are not known exactly, is of great practical importance (for example in avoiding spurious near pole-zero combinations in certain adaptive control applications) and is of some mathematical difficulty, owing to the potential discontinuity (of the degree of the GCD) as the coefficients are varied. Discontinuity is difficult to deal with, both symbolically (with parameters) and numerically, where problems that are near to points of discontinuity are ill-conditioned [2]. Some form of regularization must therefore be used, and most of the work in this area can be considered to be examining the effects of different regularizations. The most successful regularization seems to be to phrase the problem as an optimization problem, as in [3]–[5]. For an introduction to the mathematical problems in this area, see [6].

A. Notation: Approximate Polynomials and Approximate GCD

Several papers use distinct wording and notation for the objects under study here. We follow [3] and say that an *approximate polynomial* is a polynomial with coefficients that are not known exactly. We say that $d(x)$ is an *approximate* GCD of approximate polynomials $f(x)$ and $g(x)$ if there exist perturbations Δf and Δg, which are small in a sense to be specified later, such that $d(x)$ is a (true) GCD of $f+\Delta f$ and $g+\Delta g$. This can be contrasted with the notion of "quasi-GCD" of [7], in which the input polynomials f and g are known at any time only to a finite accuracy, but by some "oracle" more digits of accuracy for any coefficient can be obtained on demand. The notion of "quasi-GCD" thus fits in with mathematical and computational studies of computable real numbers but does not fit in with engineering or empirical models, where the input polynomials are known only to a limited accuracy once and for all. The paper [8] uses (differently) the terms "quasi-GCD" and "ε-GCD" to distinguish two technical notions of approximate GCD.

The works by Pan (see, e.g., [9]) show that it is also possible to compute approximate GCD by first numerically finding the roots of each polynomial, and then matching nearest approximate roots using a graph-theoretic technique. The algorithm of this paper, in contrast, works directly on the coefficients.

B. QR Factoring of the Sylvester Matrix to Find an Approximate GCD

In [1], an efficient method to use QR factoring to compute an approximate GCD is described, and the paper contains several important ideas and advances. Like the contemporary paper [10], it uses the QR factoring for stability, and also like [10], it uses Gauss elimination adapted to the structure of the Sylvester matrix in order to speed up computation, lowering the cost from $O((n+m)^3)$ to $O(n^3)$.

The paper [10] is perhaps not as easily available to the audience of this paper as the paper [1], and so, we summarize it briefly here.

It is well known that Householder transformations and Givens rotations give stable methods to compute the QR factoring of a matrix. Householder transformations are powerful tools for introducing zeros into vectors, whereas Givens rotations introduce zeros into a vector one at a time. Therefore, Givens rotations are useful for operating on structured matrices. The Sylvester matrix (2) is a block Toeplitz matrix ($\mathbb{R}^{m\times(n+m)}, \mathbb{R}^{n\times(n+m)}$) formed by the coefficient vectors from f and g in (1) below.[1] If we apply Givens rotations to the first row and $m+1$st row to eliminate g_m, then the rows i and $m+i$ for i from 2 to m can be changed in the same way. The near Toeplitz structure will not be changed until we obtain the following block matrix:

$$\begin{bmatrix} U & V \\ 0 & W \end{bmatrix}$$

where U is an $m\times m$ upper triangular matrix, 0 is a matrix with all elements zero, and W is an $n\times n$ matrix. Since U is now an

[1] All the results of this paper go through immediately in the case of complex coefficients, i.e., polynomials in $\mathbb{C}[x]$, if we replace orthogonal matrices by unitary matrices.

Manuscript received October 20, 2002; revised October 17, 2003. The associate editor coordinating the review of this paper and approving it for publication was Prof. Abdelhak M. Zoubir.

The authors are with The Ontario Research Centre for Computer Algebra, University of Western Ontario, London, ON N6A 5B7 Canada (e-mail: rob.corless@uwo.ca; watt@csd.uwo.ca; lzhi@mmrc.iss.ac.cn).

Digital Object Identifier 10.1109/TSP.2004.837413

upper triangular matrix, Householder transformations can then be applied to the submatrix W. The complexity advantage of combining Givens rotations with Householder transformations can be seen clearly. The cost of a general QR decomposition is $(4/3)(m+n)^3$ flops. Using the above special strategy, taking advantage of the structure, the flop count drops to $6nm+(4/3)n^3$.

This approach is similar in complexity to the nonorthogonal methods used in [1] to condense the Sylvester matrix into a smaller matrix, but [10] uses (as this present paper does) orthogonal reductions at all stages for stability reasons.

The paper [10] then goes on to use this method as a base method for multivariate GCD computations.

II. QR Factoring for a Sylvester Matrix

Let given polynomials f, g have degree $n \geq m$, respectively, where

$$\begin{aligned} f &= f_n x^n + f_{n-1}x^{n-1} + \cdots + f_1 x + f_0 \\ g &= g_m x^m + g_{m-1}x^{m-1} + \cdots + g_1 x + g_0. \end{aligned} \tag{1}$$

The Sylvester matrix of f and g is

$$S(f,g) = \begin{bmatrix} f_n & f_{n-1} & \cdots & f_1 & f_0 & & & \\ & f_n & f_{n-1} & \cdots & \ddots & \ddots & & \\ & & \ddots & \ddots & & \ddots & \ddots & \\ & & & f_n & f_{n-1} & \cdots & f_1 & f_0 \\ g_m & g_{m-1} & \cdots & g_1 & g_0 & & & \\ & g_m & g_{m-1} & & \ddots & \ddots & & \\ & & \ddots & \ddots & & \ddots & \ddots & \\ & & & g_m & g_{m-1} & \cdots & g_1 & g_0 \end{bmatrix}. \tag{2}$$

Row Equilibration: We will henceforth assume that the input polynomials f and g have been scaled to have unit 2-norm, and thus, the rows of $S(f,g)$ will also have unit 2-norm. This is known as row-equilibration, and to have beneficial effects on the conditioning of the matrix in certain circumstances. Here, it will simplify our error analysis somewhat, and increases the stability of the numerical computations, in essence replacing the condition numbers that come up in the analysis with an equivalent componentwise condition number [11]. This also makes the unit circle special, i.e., that $\int_C f^*(z)f(z)dz = 1$, where C is the unit circle.

Remark: Unless otherwise specified, $\|\cdot\|$ denotes the vector 2-norm. The specific notation $\|\cdot\|_2$ will sometimes be used for emphasis. Other notations include $\|\cdot\|_F$ for the Frobenius norm.

Theorem 1: [12] Suppose the QR factoring of (2) is $S(f,g) = QR$, where $Q \in \mathbb{R}^{(m+n)\times(m+n)}$ is orthogonal,[2] and R is upper triangular. Then, the last nonzero row of R gives the coefficients of a GCD of f and g.

Proof: This theorem is proved in many places. See, for example, [12]. We include the following proof here because it helps motivate the proof for the approximate polynomial case.

From the construction of the Sylvester matrix, we have

$$\begin{bmatrix} x^{m-1}f \\ \vdots \\ f \\ x^{n-1}g \\ \vdots \\ g \end{bmatrix} = Q \cdot R \begin{bmatrix} x^{n+m-1} \\ \vdots \\ x^{n-1} \\ \vdots \\ x \\ 1 \end{bmatrix} = Q \begin{bmatrix} r_{n+m-1}(x) \\ \vdots \\ r_d(x) \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{3}$$

The polynomial r_i of degree i is formed from the $(n+m-i)$th nonzero row of R for $i = d, d+1, \ldots, n+m-1$.

Suppose that x_k is a common root of multiplicity e_k of $f(x)$ and $g(x)$. Then, one can easily verify that $S\Lambda$ is the $(m+n-1) \times e_k$ zero matrix, where Λ is the $(m+n-1) \times e_k$ matrix parameterized by x_k as in (4), shown at the bottom of the page. We denote $a(a-1)\cdots(a-b+1)$ by $a^{\underline{b}}$, following [13].

From this, it is obvious that $r_d(x)$ and all its derivatives up to order $e_k - 1$ are zero at x_k. Conversely, if $r_d(x)$ and all its derivatives up to order $e_k - 1$ are zero at x_k, then by using the upper triangular structure of R we may see that $R\Lambda$ is zero. ♮

The GCD computation of f and g is equivalent to finding the null space of $S(f,g)$, i.e.,

$$\begin{aligned} f^{(\ell)}(x_k) = g^{(\ell)}(x_k) = 0 \Longleftrightarrow S \cdot \mathbf{x}_\mathbf{k}^{(\ell)} \\ = 0, \mathbf{x}_\mathbf{k}^{(\ell)} = \frac{d^\ell}{dx^\ell}[x^{n+m-1}, \ldots, x, 1]^T \Big|_{x=x_k}. \end{aligned} \tag{5}$$

Corollary 1: x_k is a common zero of f and g if and only if x_k is a zero of r_d, which is the polynomial formed by multiplying the last nonzero row of R by $\mathbf{x_k}$.

It is well known (see e.g., [11]) that QR factoring using Givens rotations or Householder transformations is numerically

[2] Q^* is the Hermitian transpose or transpose if Q is real.

$$\begin{bmatrix} x_k^{n+m-1} & (n+m-1)x_k^{n+m-2} & \cdots & (n+m-1)^{\underline{e_k-1}} x_k^{n+m-e_k} \\ x_k^{n+m-2} & (n+m-2)x_k^{n+m-3} & \cdots & \vdots \\ \vdots & \vdots & & (e_k-1)! \\ & & \iddots & \\ x_k^2 & 2x_k & & \\ x_k & 1 & & \\ 1 & & & \end{bmatrix} \tag{4}$$

stable, in the following sense. Let $\hat{R}$ be a computed upper triangular factor of S obtained via Givens rotations or Householder transformations. Then there exists an orthogonal $\hat{Q}$ such that

$$S + \Delta S = \hat{Q}\hat{R} \tag{6}$$

with $\|\Delta S\|_F \leq \eta \|S\|_F, \eta = O(\mu), \mu$ is the unit roundoff, and $\|\cdot\|_F$ is the Frobenius norm. Nevertheless, the small residual $\Delta S = \hat{Q}\hat{R} - S$ for the factoring does not guarantee a small forward error $\Delta R = \hat{R} - R$. Consider the following example:

$$\begin{aligned} f &= (x-5)(x-1/2) \\ &\quad \times (56x^8 + 83x^7 + 91x^4 - 92x^2 + 93x - 91) \quad (7) \\ g &= (x-5)(x-1/2) \\ &\quad \times (32x^8 - 37x^6 + 93x^5 + 58x^4 + 90x^2 + 53). \quad (8) \end{aligned}$$

Computing the QR factoring of $S(f/\|f\|_2, g/\|g\|_2)$ numerically for Digits $= 10$ in Maple 7, and comparing with the exact solution (being careful about the possible nonunique orderings of factors), we obtain that

$$\begin{aligned} \|\Delta S\|_F &= 0.106 \cdot 10^{-8} \\ \|\Delta R\|_F &\geq 0.11. \end{aligned}$$

This shows that the forward error (ΔR) may be many orders of magnitude larger than the backward error (ΔS). Due to the sensitivity of $\hat{R}$, Theorem 1 and Corollary 1 seem useless for numerically computing the GCD. For Example 1, the symbolic (exact) QR factoring gives us $r_{20} = r_{19} = 0$ and the GCD $(x-5)(x-1/2)$ can be discovered from the polynomial r_{18}. On the other hand, the numerical QR factoring gives us

$$\begin{aligned} &\vdots \\ r_{19} &= -0.113\,364\,838\,1x + 0.056\,682\,419\,03 \\ r_{20} &= -0.262 \cdot 10^{-10}. \end{aligned}$$

We see that the size of r_{19} is too large to be neglected. From the equation for r_{19}, the common factor $x - 1/2$ can easily be found, but the other common factor $x - 5$ is lost. The reason is shown by the following analysis.

Theorem 2: Let f and g be given univariate approximate polynomials, with common roots $x_i, 1 \leq i \leq k$, all lying inside the unit circle $|x_i| \leq \rho_1 < 1$. There may be other common roots not inside the unit circle. Then, the QR factoring of the Sylvester matrix reveals (in the last nonzero row of $\hat{R}$) a factor of the approximate GCD of f and g that contains the zeros $x_i, 1 \leq i \leq k$.

Proof: If $S = QR$ and $S + \Delta S = \hat{Q}\hat{R}$, and S is the Sylvester matrix of $f + \Delta f$ and $g + \Delta g$ such that the null space N of S is parameterized by the zeros x_i of the GCD of $f + \Delta f$ and $g + \Delta g$, then we have that

$$(S + \Delta S)N = \Delta S\, N = \hat{Q}\hat{R}N$$

and so

$$\hat{R}N = \hat{Q}^* \Delta S\, N.$$

Interpreting this matrix equation as polynomial evaluation at the common zeros of $f + \Delta f$ and $g + \Delta g$, then we see that in particular, the polynomial $r_d(x)$ arising from the last nonzero row of $\hat{R}$, when evaluated at the common zeros, will be bounded in value by

$$|r_d(x_i)| \leq \|\Delta S\| \|N\|.$$

If the roots x_i are less than 1 in magnitude, then the corresponding columns of N form a subspace N_k that also satisfies the above equation. Therefore $\|N_k\| \leq c_m$, a constant that depends on the dimension of the problem and on the multiplicity of the zeros, and we thus see that each x_i is a pseudozero of $r_d(x)$. By the results of [14], this polynomial is therefore close (in a dual norm) to the common divisor $R_d(x)$. ♮

Remark: A short calculation using the known structure of the null space shows that c_m may be taken as $(n+m)^{\underline{e}}$, where e is the maximum order of multiplicity of any zero inside the unit circle. If ρ_1 is very close to 1, then this bound may be nearly attained in practice; and if the root is of maximum possible multiplicity, namely m, then we see that this constant may grow exponentially with m, in these special circumstances. Thus, the only problem here is with a highly multiple root x^* close to the unit circle (close to zero is not a problem). If there is only one such root x^*, expanding the polynomial in the basis $1, x - x^*, (x - x^*)^2, \ldots$ may improve stability.

Theorem 3: If f and g are given as in Theorem 2, then it occurs in practice that if any x_j is outside the unit circle, it might not be detected by the QR factoring algorithm.

Proof: The numerical QR factoring gives us

$$\begin{bmatrix} x^{m-1}f \\ \vdots \\ f \\ x^{n-1}g \\ \vdots \\ g \end{bmatrix} + \Delta S \begin{bmatrix} x^{n+m-1} \\ \vdots \\ x^{n-1} \\ \vdots \\ x \\ 1 \end{bmatrix} = \hat{Q} \cdot \hat{R} \begin{bmatrix} x^{n+m-1} \\ \vdots \\ x^{n-1} \\ \vdots \\ x \\ 1 \end{bmatrix}. \tag{9}$$

With high probability, $\|\Delta S \cdot \mathbf{x}\| \approx \|\Delta S\| \|\mathbf{x}\|$, because usually (with probability 1), ΔS is not itself a Sylvester matrix, and hence, $\mathbf{x}$ is not nearly in its null space. The common roots of f and g will still correspond to the null space of $\hat{Q}\hat{R}$ if and only if the perturbation term $\Delta S \cdot \mathbf{x}$ can be neglected. Supposing $\|f\|_2 = \|g\|_2 = 1$, then $\|S\|_F = \sqrt{n+m}$. If $|x| > 1$, $\|\Delta S \cdot \mathbf{x}\|_2$ may increase quickly with $m + n$.♮

Let us check Example 1 again. For the common root $x = 1/2$,

$$\|\Delta S \cdot \mathbf{x}\|_2 \approx 0.122 \cdot 10^{-8}.$$

In contrast, for the common factor $x - 5$

$$\|\Delta S \cdot \mathbf{x}\|_2 \approx 0.206 \cdot 10^{5}.$$

The perturbation is large enough to disrupt the null space. Therefore, it is not a surprise that the root $x = 1/2$ can be recovered from r_{19}, whereas the other common root $x = 5$ is missing.

If we compute the QR factoring for Digits $=$ 20 in Maple, $\|\Delta S\|_F = 0.1526 \cdot 10^{-18}$, and

$$\begin{aligned}&\vdots\\ r_{18} &= 0.625\,43 - 0.137\,59x + 0.025\,017x^2\\ r_{19} &= 0.251\,95 \cdot 10^{-11} - 0.5039 \cdot 10^{-11}x\\ r_{20} &= -0.129\,70 \cdot 10^{-20}.\end{aligned}$$

Both common roots can be recovered from r_{18} since $\|\Delta S \cdot \mathbf{x}\|_2 \le 10^{-10}$ for $x = 1/2$ and less than 10^{-5} for $x = 5$ because we worked to higher precision here (and the input was in fact exact).

In the special case where all common roots of f and g lie inside the unit disc, the last "nonzero" row of R as in Theorem 1 will give us a good candidate for the GCD.

A. Reversals

Similarly, if all common roots of f, g lie outside the unit disc, the QR factoring of $S(\underline{f}, \underline{g})$, where $\underline{f} = x^{\deg f} f(1/x), \underline{g} = x^{\deg g} g(1/x)$, are the reversals (reciprocals) of f and g, will provide us the reversal (reciprocal) of the GCD of f and g. Consequently, we can detect relatively prime numerical polynomials from the QR factors of $S(f, g)$ and $S(\underline{f}, \underline{g})$.

B. Relative Primality

It has been proved in [8] that a lower bound for perturbations Δf and Δg such that $f + \Delta f$ and $g + \Delta g$ have a common root is $(1/\kappa)$, where

$$\kappa = \left\| \begin{bmatrix} v & \underline{v} \\ u & \underline{u} \end{bmatrix} \right\| \tag{10}$$

can be found from $u, v, \underline{u}, \underline{v}$, which are polynomials solving the Diophantine equations:

$$f \cdot v + g \cdot u = 1, \deg u < \deg f, \deg v < \deg g \tag{11}$$
$$f \cdot \underline{v} + g \cdot \underline{u} = x^{n+m-1}, \deg \underline{u} < \deg f, \deg \underline{v} < \deg g \tag{12}$$

which arise from imposing relative primality on f with g and $\underline{f}$ with $\underline{g}$. We can find this bound from the QR factoring used here, as follows. Suppose $S(f, g) = Q \cdot R$ and $S(\underline{f}, \underline{g}) = \underline{Q} \cdot \underline{R}, u, \underline{u}, v, \underline{v}$ are obtained from the last rows of $Q^T, R, \underline{Q}^T, \underline{R}$. Since $Q, \underline{Q}$ are orthogonal, κ is determined by the last rows of R and $\underline{R}$.

We note that the complexity of [8] is typically $O((m+n)^2)$, which is therefore "fast." Here, since the Sylvester matrix consists of two Toeplitz blocks, we can apply selected Givens rotations to take advantage of the special structure of S and obtain a more efficient QR factoring, as in [10]. The complexity is $O(n^3)$. We use orthogonal transformations in an effort to delay the accumulation of rounding errors.

Now that we have a stable and practical method, we may look for ways to make it as fast as the weakly stable methods.

C. Common Roots Outside the Unit Circle

Since the common roots of f and g inside the unit circle are easily identified by using QR factoring, we can find an approximate common factor d_1 of f and g by QR factoring of $S(f, g)$ and another common factor d_2 by applying the QR factoring to $S(f^*, g^*)$, where $f^* = \underline{(f/d_1)}, g^* = \underline{(g/d_1)}$ are the reversals of f and g, after having divided out the common factor already found. For Example 1, after dividing out the factor $x - 1/2$, the QR factoring of f^*, g^* for Digits $= 10$ in Maple 7 returns:

$$\begin{aligned}&\vdots\\ r_{17} &= 0.005\,438 - 0.027\,19x\\ r_{18} &= -0.267 \cdot 10^{-11}.\end{aligned}$$

The common root $x = 5$ can be easily identified from r_{17}.

From our experiments with thousands of examples, of degrees up to approximately 1000, we find that about 90% of all problems can be solved in the above way. The algorithm even works for polynomials of high degree. See the last several examples in Table I. This can be explained by the rapid increase of $\|\Delta S \cdot \mathbf{x}\|$ for $|x| > 1$ when the degrees of f and g are large. Therefore, there will be a clear separation of common roots inside the unit circle from common roots outside the unit circle.

Example 2—Random Polynomials of Large Degree:

$$\begin{aligned} f &= -0.011\,637 + 0.011\,604x + \cdots \\ &\quad + 0.003\,553\,9x^{1019} + 0.004\,498\,0x^{1020} \\ g &= 0.006\,016\,3 - 0.002\,343\,2 - \cdots \\ &\quad - 0.006\,734\,6x^{1019} + 0.012\,149x^{1020}. \end{aligned}$$

Suppose $S(f, g) = QR$. We observe that the norms of the right-bottommost submatrices of R have a big jump in norm between the last 15th and 16th rows

$$0.622 \cdot 10^{-11}, 0.146 \cdot 10^{-10}, \ldots, 0.646 \cdot 10^{-9}$$
$$0.182 \cdot 10^{-8}, 0.376 \cdot 10^{-8}, 0.006\,77 \ldots.$$

The last 15th row of R gives a common factor d_1 of f, g with backward error of the order 10^{-7}. The roots of d_1 are all inside the unit circle.

The QR factoring of $S(\underline{f/d_1}, \underline{g/d_1})$ gives us another common factor d_2 of degree 6 as the norm of R also has a big jump between the last sixth and seventh rows:

$$0.165 \cdot 10^{-13}, 0.439 \cdot 10^{-12}, 0.133 \cdot 10^{-11}, 0.749 \cdot 10^{-11}$$
$$0.148 \cdot 10^{-10}, 0.320 \cdot 10^{-10}, 0.05 \ldots.$$

d_2 has all its roots outside the unit circle. The details of the backward errors are given in the second last row in Table I.

D. Graeffe's Root-Squaring to Improve Separation from the Unit Circle

Graeffe's root-squaring technique is a classical technique to transform one polynomial problem to another, hopefully simpler, problem. We give a brief overview of this process here, but for details see any older numerical analysis text, e.g., [15]. The basic idea is this: Suppose $f(x) = f_n(x - \alpha_1)(x - \alpha_2) \ldots (x - \alpha_n)$ is the polynomial whose zeros α_k we wish to approximate.

TABLE I

BACKWARD ERRORS FOR ALGORITHM GCD. d_1 IS THE COMMON FACTOR OF f AND g FOUND BY SUBROUTINE GCDAux1, WHICH COMPUTES THE QR FACTORS OF THE SYLVESTER MATRIX OF f AND g. d_2 IS THE COMMON FACTOR FOUND BY SUBROUTINE GCDAux2, WHICH COMPUTES THE QR FACTORS OF THE SYLVESTER MATRIX OF THE REVERSALS OF f/d_1, AND g/d_1. $d = d_1 \cdot d_2$ IS THE APPROXIMATE GCD OF f AND g. AN (*) DENOTES A DIFFICULT CASE, WHERE THE SPLITTING ALGORITHM IS NEEDED

$(\deg(f), \deg(g))$	$\deg(d_1)$	$\deg(d_2)$	$\|u \cdot f + v \cdot g - d\|$	$\|f - d \cdot f_1\|$	$\|g - d \cdot g_1\|$
(30, 25)	8	2	0.326e-6	0.718e-7	0.926e-7
(42, 32)	11	1	0.174e-6	0.981e-7	0.780e-7
(55, 50)	18	2	0.553e-5	0.404e-5	0.356e-5
(55, 50)*	10	10	0.154e-7	0.651e-7	0.336e-7
(65, 45)	3	12	0.208e-6	0.151e-6	0.523e-6
(65, 55)*	14	11	0.156e-5	0.470e-6	0.467e-6
(65, 55)	21	4	0.820e-8	0.136e-7	0.129e-7
(75, 65)	14	1	0.890e-6	0.389e-6	0.168e-6
(80, 60)	19	1	0.371e-4	0.642e-5	0.290e-5
(80, 60)*	10	10	0.349e-7	0.719e-7	0.862e-7
(100, 80)	36	4	0.216e-3	0.210e-4	0.193e-4
(100, 80)*	18	22	0.123e-7	0.388e-7	0.409e-7
(110, 90)	5	5	0.270e-7	0.321e-8	0.684e-8
(155, 155)	97	6	0.628e-4	0.367e-4	0.478e-4
(159, 159)	99	5	0.203e-4	0.389e-6	0.714e-6
(168, 101)*	33	26	0.632e-7	0.233e-7	0.198e-7
(166, 79)*	28	33	0.411e-7	0.217e-6	0.213e-6
(156, 78)*	28	26	0.702e-5	0.631e-6	0.740e-6
(138, 83)*	30	20	0.7335e-6	0.213e-6	0.131e-6
(132, 128)*	19	29	0.193e-5	0.259e-6	0.239e-6
(208, 108)	4	4	0.192e-4	0.293e-5	0.208e-5
(220, 120)	9	11	0.277e-4	0.258e-4	0.156e-4
(320, 220)	12	8	0.381e-6	0.266e-7	0.214e-7
(510, 210)	4	6	0.322e-6	0.108e-8	0.912e-8
(530, 230)	18	12	0.327e-5	0.112e-5	0.127e-5
(1020,1020)	14	6	0.192e-6	0.183e-7	0.187e-7
(1024,1022)	10	10	0.282e-5	0.421e-8	0.227e-8
(1020,1020)	8	12	0.121e-3	0.247e-5	0.877e-6

Consider

$$\begin{aligned} f_2(x) &= (-1)^n f(-\sqrt{x}) f(\sqrt{x}) \\ &= (-1)^n f_n^2 (x - \alpha_1^2)(x - \alpha_2^2) \cdots (x - \alpha_n^2). \end{aligned} \quad (13)$$

The first equation gives us a rational means to compute the coefficients of f_2, while the second equation shows that the roots of f_2 are the squares of the roots of f. The coefficients of f_2 can be computed using the fast Fourier transform (FFT) in time $O(n \log n)$.

Roots less than 1 in magnitude become smaller, therefore, while roots larger than 1 become larger. Thus, each step of the root-squaring process taken improves the separation of the roots from the unit circle. One drawback is that initially close complex zeros may become more separated (in angle and in magnitude) by this process, so we do not wish to use too many root-squaring operations.

In practice, we find that only a few root-squaring steps are needed to give useful improvements to the QR factoring. However, it is not a panacea, and for difficult cases, a further refinement is needed.

III. SPLITTING POLYNOMIALS OVER THE UNIT CIRCLE

In our experiments, we noticed some difficult cases where $\|\Delta S \cdot \mathbf{x}\|$ is of moderate size because there were common roots very close to the unit circle. Without a refinement of the above technique, it is hard to compute the GCD correctly in such a case. We now present one such refinement.

It is well known that, for high degree polynomials with coefficients randomly chosen from a normal or uniform distribution, the roots cluster about the unit circle [16]. However, for many cases occurring in practice, the common roots of f and g are distributed randomly inside or outside the unit circle. If the common roots are too close to the unit circle, the numerical QR factoring may not give us correct information about the GCD.

One possible solution is to split (factor) f or g over the unit circle. For example, $f(x) = f_1(x) \cdot A(x) \cdot f_2(x)$ where all zeros of f_1 lie inside the circle of radius $\rho_1 < 1$, all zeros of f_2 lie outside the circle of radius $\rho_2 > 1$, and $A(x)$ has all its zeros in the "ambiguous" annulus $\rho_1 \leq |z| \leq \rho_2$. Then, $\text{GCD}(f_1, g), \text{GCD}(\underline{f_2}, g)$ can be obtained correctly by the QR factoring. We will discuss this approach in more detail in a subsequent paper. For the rest of this paper, we assume that $A(x)$ can always be taken to be 1, i.e., that there are no common roots in the ambiguous annulus. This can be made more nearly true always by using Graeffe's root-squaring process. The splitting helps the stability of the algorithm considerably.

Example 3—A Difficult Example: Let $f = f_1 \cdot h_1, g = g_1 \cdot h_1, f_1, g_1$ be relatively prime, and

$$\begin{aligned} f_1 &= -26 + 31x + 18x^2 + 56x^3 + \cdots + 65x^{22} \\ &\quad - 48x^{23} + 3x^{24} + 64x^{25} \\ g_1 &= 37 - 31x - 45x^2 - 80x^3 + \cdots + 85x^{25} \\ &\quad - 54x^{26} + 49x^{27} + 79x^{28} \\ h_1 &= 1 + 14x + 90x^2 - 4x^3 + \cdots - 8x^{11} \\ &\quad - 55x^{12} - 62x^{13} - 10x^{14}. \end{aligned}$$

The norms of the right bottommost submatrices of R increase steadily as

$$\begin{aligned} &.164 \cdot 10^{-14}, \ldots, .703 \cdot 10^{-8}, .168 \cdot 10^{-7}, .156 \cdot 10^{-5} \\ &\qquad .157 \cdot 10^{-5}, .758 \cdot 10^{-4}, .139 \cdot 10^{-3}, .0111, \ldots. \end{aligned}$$

It is therefore hard to obtain a good approximation to h_1.

However, if we split the polynomial $f = f_1 \cdot f_2$ over the unit circle and $S(f_1, g) = Q_1 \cdot R_1, S(\overline{f_2}, \overline{g}) = Q_2 \cdot R_2$, the norms of the right-bottom submatrices of R_1, R_2 have a big jump in norm:

$$\begin{aligned} &0.669 \cdot 10^{-10}, 0.645 \cdot 10^{-9}, 0.140 \cdot 10^{-8}, 0.225 \cdot 10^{-8} \\ &0.413 \cdot 10^{-8}, 0.001\,52, \ldots \\ &0.580 \cdot 10^{-13}, 0.601 \cdot 10^{-12}, 0.411 \cdot 10^{-11}, \ldots \\ &0.336 \cdot 10^{-9}, 0.479 \cdot 10^{-9}, 0.002\,01, \ldots \end{aligned}$$

It follows that h_1 can be retrieved from the sixth last row of R_1 and the 10th last row of R_2. In this case, there are no roots too close to the unit circle for this refinement, and h_1 is correctly recovered by this refined technique.

The splitting can be performed in a classical way, using contour integrals. This method has been discussed by many authors [17]–[20] as a tool for finding all roots of polynomials. The main steps are given in the following algorithm. It is time-consuming to evaluate the contour integrals to high accuracy. Therefore, the algorithm first splits the polynomial to a relatively low accuracy and then refines the factoring by an iterative method. The first step, root-squaring, is used to push the roots away from the unit circle. The FFT is used to accelerate the computation in steps 1 and 2.1. Unlike the algorithm in [19], which uses the fast methods available for Padé approximation in Step 3, we recursively make use of QR factoring for GCD computation. The reason is that polynomials $p_{k-j}(x)$ and $G_{k-j+1}(x^2)$ only have common zeros outside the unit circle since the all roots of G_{k-j+1} lie outside the unit circle. Step 2.2.1 can also use QR factoring for the same reason because the roots of F_k and G_k are well separated by the unit circle.

Step 2.2.2 needs a good algorithm for approximate division. See [21] for a description of the algorithm that we have used.

Algorithm Split

Input: One monic univariate polynomial $p(x)$ with degree n, radius $r = 1$, tolerance ε.

Output: Univariate polynomials F, G, such that $\|p - F \cdot G\| < \varepsilon$, split by the unit circle and center $(0, 0)$.

Step 1) [recursive lifting] Apply k root-squaring Graeffe's steps (usually k is 1, 2 or 3)

$$p_{i+1}(x) = (-1)^n p_i(-\sqrt{x}) p_i(\sqrt{x}), \quad i = 0, 1, \ldots, k-1.$$

Step 2) [splitting p_k]

2.1 [rough approximation of F and G]

2.1.1 Compute

$$s_N = \frac{1}{2\pi i} \int_C x^N \frac{p_k'(x)}{p_k(x)} dx = \sum_{i=1}^{v} z_i^N$$

where C is the unit circle; z_i are all the roots of p_k inside C, i.e. ,the roots of F_k; v is the integer closest to s_0, i.e., the number of zeros of p_k inside C.

2.1.2 From $s_1, s_2, \ldots, s_v$, compute the coefficients of the polynomial F_k.

2.1.3 Compute $G_k = \text{quo}(p_k, F_k)$.

2.2 [Newton's iteration, one step]

2.2.1 Compute u and v such that

$$u \cdot F_k + v \cdot G_k = 1.$$

2.2.2 Compute ΔF_k and ΔG_k in higher precision as

$$\begin{aligned} \Delta F_k &= \text{polyrem}\ (p_k \cdot v, F_k) \\ \Delta G_k &= \text{polydiv}\ (p_k - F_k \cdot G_k - \Delta F \cdot G_k, F_k). \end{aligned}$$

2.2.3 Set $F_k = F_k + \Delta F_k, G_k = G_k + \Delta G_k$.

Step 3) [recursive descending] For i from 1 to k, do

$$\begin{aligned} G_{k-j} &= \text{gcd}\ (p_{k-j}(x), G_{k-j+1}(x^2)) \\ F_{k-j} &= \text{polydiv}\ (p_{k-j}, G_{k-j}). \end{aligned}$$

Here, `polyrem(a, b)` is the remainder on approximate division of a by b, while `polydiv` is the best fit quotient on division of a by b, that is, it has the smallest remainder in the 2-norm sense. The algorithm is expected to improve in performance if this is replaced by approximate division using total least squares.

IV. Algorithm for GCD Computation

Algorithm GCD

Input: Two univariate polynomials $f(x)$ and $g(x)$, tolerance ε.

Output: Univariate polynomials u, v, d such that $\|f - d \cdot f_1\| < \varepsilon, \|g - d \cdot g_1\| < \varepsilon, \|uf + vg - d\| < \varepsilon$ and $\deg(u) < \deg(g) - \deg(d), \deg(v) < \deg(f) - \deg(d)$.

Note that $\|uf + vg - d\| < \varepsilon$ can be rewritten as $\|u\Delta f + v\Delta g\| < \varepsilon$, where $\Delta f = f - d \cdot f_1$ and $\Delta g = g - d \cdot g_1$. This represents an extra constraint on u and v and, thus, disallows them from growing to be too large.

Step 1) [Initialization]

1.1 Make the input f and g to be unit 2-norm with positive leading coefficients.

Step 2) [QR-factoring]

2.1 Form the Sylvester matrix S of f and g.

2.2 Compute the QR-factoring for $S = Q \cdot R$.

2.3 Suppose $R_{22}^{(k)}$ are the last $(k+1) \times (k+1)$ submatrices of R such that $\|R_{22}^{(k)}\| > \varepsilon$ but $\|R_{22}^{(k-1)}\| < \varepsilon$.

Case 1) $[\|R_{22}^{0}\| > \varepsilon] : d_1 = 1, u$ and v are formed by the last row of Q^T.

Case 2) $[(\|R_{22}^{(k)}\|)/(\|R_{22}^{(k-1)}\|) > 0.1/\varepsilon] : d_1$'s coefficients are given by the first row of $R_{22}^{(k)}$.

Case 3) $[\exists k_1(\text{biggest})$ such that $(\|R_{22}^{(k_1)}\|)/(\|R_{22}^{(k_1-1)}\|) > 0.1/\varepsilon] : d_1$'s coefficients are given by the first row of $R_{22}^{(k_1)}$.

Case 4) [Difficult case]: Use the algorithm **Split** to find the common roots of f and g inside the unit circle and form the divisor d_1.

Step 3) [Coprime check]

3.1 Compute cofactors f_1 and g_1:

$$f_1 = \text{polydiv}\,(f, d_1)$$
$$g_1 = \text{polydiv}\,(g, d_1).$$

3.2 Apply Step 2 to $x^{\deg(f_1)} \cdot f_1(x^{-1}), x^{\deg(g_1)} \cdot g_1(x^{-1})$ to obtain d_2.

3.3 Apply Step 2 to cofactors of f, g w.r.t. $d = d_1 \cdot d_2$ to obtain u, v (case 0).

Step 4) Return u, v, d.

V. Multiple Common Roots

The method given in this paper has no difficulty finding accurate common factors of problems that have multiple approximate common roots; however, it is the coefficients of the factors with multiple roots that are recovered and not the multiple roots themselves. To accurately find the multiple roots from these approximate common factors requires a separate analysis.

VI. Theoretical Complexity Analysis

Suppose that the degree of f is n, the degree of g is m, and $n \geq m$. The complexity of the main steps of Algorithm GCD (page 16) are as follows.

1) Step 2.2. $O(n^3)$ (see Section I-B)
2) Step 2.3. See below (Algorithm Split)
3) Step 3.1. $O(n^2)$ because the matrix involved in the polynomial division is a Toeplitz matrix. The complexity can be achieved by using the algorithm in [22].

Supposing that the degree of the polynomial to be split is n, the complexity of the main steps in Algorithm Split is as follows.

1) Step 1. $O(kn \log n)$, k is usually 1, 2, or 3.
2) Step 2.1. $O(n \log^2 n)$.
3) Step 2.2.1. $O(n^2)$. Since the roots of two factors are well separated (with respect to the unit circle), the Sylvester matrix is well conditioned. Moreover, since a Sylvester matrix is a quasi-Toeplitz matrix, the method in [23] gives a fast stable way to find u and v.
4) Step 2.2.2. $O(n^2)$.
5) Step 3. $O(n^3)$. We apply QR factoring to the reciprocal of the two polynomials since the two polynomials only have common roots outside the unit circle.

VII. Test Results

A. Comparison with [1, Ex. 2]

The following is [1, Ex. 2] i. Let $A(z) = d(z)A_1(z)$ and $B(z) = d(z)B_1(z)$, where

$$d(z) = z^5 - 0.6z^4 - 0.05z^3 - 0.05z^2 - 1.05z + 0.55 \quad (14)$$

$$B_1(z) = z^9 + 1.95z^8 + 0.6699z^7 + 0.1978z^6 + 0.2271z^5 - 1.5652z^4 - 1.991\,18z^3 - 0.7413z^2 - 0.0801z + 0.0634 \quad (15)$$

$$A_1(z) = z^{10} - 1.6z^9 + 2.43z^8 - 1.148z^7 + 1.2248z^6 + 1.3875z^5 - 0.9895z^4 + 0.9751z^3 - 0.7813z^2 - 0.623z + 0.0692. \quad (16)$$

We note that $d(z)$ has one root inside the unit circle, and four outside. Using the technique of this paper, the root inside the unit circle is easily found by a QR factoring of the Sylvester matrix of A and B, and the four roots outside are found by a QR factoring of the reversals of A and B.

The paper [1] reported a failure of the condition estimator of the method of that paper. We believe that this failure was, in essence, caused by the fact that some of the common roots were inside and some were outside the unit circle. The improvement of this present paper is sufficient to allow this example to be solved in a straightforward way, even without the contour integral splitting refinement.

Assume $A(z), B(z)$ are perturbed by noise uniformly distributed over the interval $[-10^{-4}, 10^{-4}]$, for example

$$p_1 = -0.000\,045\,z - 0.000\,026\,z^{14} + 0.000\,027\,z^{13} + 0.000\,007\,z^7 + 0.000\,031\,z^2 + 0.000\,097\,z^5$$
$$p_2 = 0.000\,067\,z^{10} + 0.000\,077\,z^{11} + 0.000\,009\,z^9 - 0.000\,09\,z^8 + 0.000\,017\,z^7 - 0.000\,007\,z^2.$$

What follows is the output of our prototype[3] Maple implementation of this algorithm, with a "verbose" flag set to display diagnostics.

```
u, v, G := GCD(A, B, z, 10^(−4));
GCDAux1:   "the norm of last row"   .284 656 437 560 687 029e-7
GCDAux1:   "the norm of row i"   .831 627 567 1e-5
GCDAux1:   "the norm of row i"   .148 122 525 4e-4
GCDAux1:   "the norm of row i"   .705 676 781 1e-4
GCDAux1:   "the norm of row i"   .270 490 103 4e-3
GCDAux1:   "difficult case"   3.833 059 421
evalpower:   "the number of evaluation points"   128
newtoncorr:   "backward error before Newton correction"
.354 778 554 787 317 338 90e-8
newtoncorr:   "backward error after Newton correction"
.494 576 642 755 856 003 18e-14
liftsplit:   "the recursive lifting"   2
gcd:   "the norm of last row"   .229 531 605 543 399 416 00e-19
gcd:   "the norm of row i"   .466 052 253 477 438 643 45e-19
gcd:   "the norm of row i"   .337 929 733 532 576 706 83e-18
gcd:   "the norm of row i"   .975 512 977 864 364 667 87e-17
gcd:   "the norm of row i"   .379 995 406 168 670 813 04
liftsplit:   "the recursive lifting"   1
gcd:   "the norm of last row"   .106 642 541 366 968 712 63e-18
gcd:   "the norm of row i"   .267 301 840 882 935 266 29e-18
gcd:   "the norm of row i"   .470 710 960 065 758 619 55e-18
gcd:   "the norm of row i"   .725 944 217 953 661 048 27e-18
gcd:   "the norm of row i"   .496 357 078 317 834 075 65
GCDAux1:   "Degree of GCD and backward error for f,g"   1
.277 581 160 4e-5   .552 008 035 6e-5
GCDAux2:   "the norm of last row"   .687 906 778 348 371 770e-6
GCDAux2:   "the norm of row i"   .144 656 424 3e-5
GCDAux2:   "the norm of row i"   .328 231 424 1e-5
GCDAux2:   "the norm of row i"   .116 765 445 1e-4
GCDAux2:   "the norm of row i"   .125 189 140 6
GCDAux2:   "quick decrease"   10 721.420 24
GCDAux2:   "Degree of GCD and backward error for f,g"   4
.112 814 847 2e-4   .829 114 748 7e-5
```

$$
\begin{aligned}
u = {} & -2.535\,456\,495 - 13.239\,344\,93z - 25.830\,168\,28z^2 \\
& - 17.924\,652\,45z^3 + 3.008\,400\,870z^4 \\
& + 4.855\,307\,551z^5 + 11.934\,682\,74z^6 \\
& + 24.995\,214\,76z^7 + 12.246\,681\,03z^8 \\
v = {} & 10.776\,571\,20 + 3.167\,861\,608z - 3.131\,239\,168z^2 \\
& - .368\,745\,126\,8z^3 - 9.197\,080\,343z^4 \\
& - 21.268\,446\,93z^5 + 15.196\,575\,43z^6 \\
& - 29.535\,337\,51z^7 + 18.481\,366\,19z^8 - 12.247\,267\,91z^9 \\
G = {} & .279\,076\,256 - .760\,829\,618\,6z - .025\,350\,494\,3z^2 \\
& - .025\,328\,610\,52z^3 - .304\,433\,627\,2z^4 + .507\,242\,283\,2z^5.
\end{aligned}
$$

If we start with the QR-factorization of reciprocal of A and B, then no splitting is needed. Since the final results are quite similar to the u, v, G above, we omit them here but just show the diagnostics.

```
GCDAux2:   "the norm of last row"   .827 778 337 414 056 300e-6
GCDAux2:   "the norm of row i"   .171 812 930 8e-5
GCDAux2:   "the norm of row i"   .288 318 074 2e-5
GCDAux2:   "the norm of row i"   .132 082 912 5e-4
GCDAux2:   "the norm of row i"   .119 536 924 8
GCDAux2:   "quick decrease"   9050.143 015
GCDAux2:   "Degree of GCD and backward error for f,g"   4
.734 782 691 0e-5   .798 072 620 3e-5
GCDAux1:   "the norm of last row"   .102 517 511 356 116 598e-6
GCDAux1:   "the norm of row i"   .177 417 393 2e-2
GCDAux1:   "quick decrease"   17 306.057 35
GCDAux1:   "Degree of GCD and backward error for f,g"   1
.798 796 977 1e-5   .156 137 594 6e-4
```

[3]At the time this paper was written, only a prototype was available. Now, by the efforts of L. Zhi and H. Kai, this algorithm has been incorporated into Maple 9 for public distribution.

B. *Summary of Tests with High Degree Random Polynomials*

See Table I.

VIII. Concluding Remarks

This paper identifies a difficulty with previous attempts at practical methods for the computation of approximate GCD and presents an improved alternative together with an error analysis, theoretical complexity analysis, and experimental results on several thousand examples. The method used in this paper seems to be of potential use in practice, for polynomials of moderately large degree (up to about 1000). One open problem of theoretical interest is what to do about common roots in the (narrow) ambiguous annulus $\rho_1 \leq |z| \leq \rho_2$, and we will pursue this in a future paper. Another open problem is whether fast $O(n^2)$ QR factoring [23] can be stably used in this context.

Acknowledgment

L. Zhi would like to thank Prof. M.-T. Noda for useful suggestions. The authors thank H. Kai for polishing and delivering the QRGCD package for distribution with Maple 9.

References

[1] C. J. Zarowski, X. Ma, and F. W. Fairman, "A QR-factorization method for computing the greatest common divisor of polynomials with real-valued coefficients," *IEEE Trans. Signal Processing*, vol. 48, pp. 3042–3051, Dec. 2000.

[2] J. W. Demmel, "On condition numbers and the distance to the nearest ill-posed problem," *Numer. Math.*, vol. 51, pp. 251–289, 1987.

[3] 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 Algebraic Computation*, A. H. M. Levelt, Ed.. Montréal, QC, Canada, pp. 195–207.

[4] N. K. Karmarkar and Y. N. Lakshman, "Approximate polynomial greatest common divisors and nearest singular polynomials," in *Proc. Int. Symp. Symbolic Algebraic Comput.*, Zürich, Switzerland, 1996, pp. 35–42.

[5] ——, "On approximate GCD's of univariate polynomials," *J. Symbolic Comput.*, vol. 26, no. 6, pp. 653–666, 1998. Special issue of the JSC on Symbolic Numeric Algebra for Polynomials.

[6] *Handbook of Computer Algebra*, V. Weispfennig, J. Grabmeier, and E. Kaltofen, Eds., Springer Verlag, Heidelberg, 2002, pp. 113–127. E. L. Kaltofen, R. M. Corless, S. M. Watt, Symbolic/numeric algorithms.

[7] A. Schönhage, "Quasi-GCD computations," *J. Complexity*, vol. 1, pp. 118–137, 1985.

[8] B. Beckermann and G. Labahn, "When are two polynomials relatively prime?," *J. Symbolic Comput.*, vol. 26, no. 6, pp. 677–689, 1998. Special issue of the JSC on Symbolic Numeric Algebra for Polynomials.

[9] V. Y. Pan, "Solving a polynomial equation: Some history and recent progress," *SIAM Rev.*, vol. 39, no. 2, pp. 187–220, 1997.

[10] L. Zhi and M.-T. Noda, "Approximate gcd of multivariate polynomials," in *Proc. ASCM*, 2000, pp. 9–18.

[11] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*. Philadelphia, PA: SIAM, 1996.

[12] M. A. Laidacker, "Another theorem relating Sylvester's matrix and the greatest common divisor," *Math. Mag.*, vol. 42, pp. 126–128, 1969.

[13] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics*. Reading, MA: Addison-Wesley, 1994.

[14] H. J. Stetter, "The nearest polynomial with a given zero, and similar problems," *SIGSAM Bulletin: Commun. Comput. Algebra*, vol. 33, no. 4, pp. 2–4, 1999.

[15] B. Carnahan, H. A. Luther, and J. O. Wilkes, *Applied Numerical Methods*. New York: Wiley, 1969.

[16] L. A. Shepp and R. J. Vanderbei, "The complex zeros of random polynomials," *Trans. AMS*, vol. 347, no. 11, pp. 4365–4384, Nov. 1995.

[17] L. M. Delves and J. N. Lyness, "A numerical method for locating the zeros of an analytic function," *Math. Comput.*, vol. 21, pp. 543–560, 1967.

[18] P. Kirrinnis, "Partial fraction decomposition in $\mathbb{C}[\mathcal{F}]$ and simultaneous Newton iteration for factorization in $[\mathcal{F}]$," *J. Complexity*, vol. 14, pp. 378–444, 1998.

[19] V. Y. Pan, "Univariate polynomials: Nearly optimal algorithms for factorization and rootfinding," in *Int. Symp. Symbolic and Algebraic Computation*, B. Mourrain, Ed.. London, ON, Canada, 2001, pp. 253–267.

[20] A. Schönhage, "The Fundamental Theorem of Algebra in Terms of Computational Complexity," Math. Dept., Univ. Tübingen, Tübingen, Germany, Tech. Rep., 1982.

[21] R. M. Corless, C.-P. Jennerod, G. Labahn, S. M. Watt, and L. Zhi, "A Report on the SNAP Package in Maple," ORCCA, London, ON, Canada, Tech. Rep., 2002.

[22] A. W. Bojanczyk, R. P. Brent, and F. R. de Hoog, "Stability analysis of a general Toeplitz systems solver," *Num. Algorithms*, vol. 10, pp. 225–244, 1995.

[23] S. Chandrasekaran and A. H. Sayed, "A fast stable solver for nonsymmetric Toeplitz and quasi-Toeplitz systems of linear equations," in *SIMAX*, vol. 19, 1998, pp. 107–139.

Robert M. Corless received the Ph.D. degree from the Mechanical Engineering Department, University of British Columbia, Vancouver, Canada. in 1987.

Since then, he has been a professor in the Applied Mathematics Department at the University of Western Ontario, London, Canada, and was appointed Chair of department in July 2002. He is the author of more than 80 papers in the areas of flow-induced vibration, nonlinear dynamical systems, numerical analysis, symbolic computation, and the effective use of technology in mathematics education. His principal interests are now hybrid symbolic–numeric computation, especially for reliable and efficient methods to solve polynomial problems. He was the Editor-in-Chief for the *ACM Sigsam Bulletin* from 1996 to 2000, Chair of ACM Sigsam from 2000 to 2003, and is now Past Chair. He is the author of *Essential Maple* (New York: Springer-Verlag).

Dr. Corless received the University of Western Ontario Faculty of Science Distinguished Research Professorship in 1999.

Stephen M. Watt received the Ph.D. degree in 1986 from the University of Waterloo, Waterloo, ON, Canada.

He has held positions as research staff member at the IBM T.J. Watson Research Center, Yorktown Heights, NY; professor at the University of Nice, NIce, France; and professor at the University of Western Ontario, London, where he served as chair of the Department of Computer Science from 1997 to 2002. His research interests lie primarily in the areas of computer algebra, programming languages, and compiler implementation. He has been one of the principals in the creation of the Maple and Axiom computer algebra systems, the Aldor programming language and compiler, and the MathML standard for mathematics in XML documents.

Dr. Watt is the Director of the Ontario Research Centre for Computer Algebra and received the 2002 Faculty of Science Distinguished Research Professorship and the Canarie IWAY award for work in mathematical data communication.

Lihong Zhi received the B.S. degree in mathematics from Peking University, Beijing, China, and the Dr.Sci. degree in computer algebra from the Institute of Systems Science, Academia Sinica, Beijing, China.

She is an associate professor with the Institute of Systems Science, Academia Sinica. From 1998 to 2001, she was an assistant professor with the Department of Computer Science, Ehime University, Matsuyama, Japan. From 2001 to 2002, she did post-doctoral research at the Ontario Research Center for Computer Algebra, University of Western Ontario, London, ON, Canada. Her research interests include numeric/symbolic hybrid algorithms, parallelism in symbolic mathematical computations, polynomial factorization over algebraic extension fields, and Ritt-Wu's characteristic set method.