Symbolic-Numeric Computation
D. Wang and L. Zhi, Eds.
Trends in Mathematics, 17–45

Ill-conditioned Properties and Hybrid Computations

Matu-Tarow Noda

Abstract. Approximate algebraic computation (AAC) has been one of the most important research areas in algebraic computation. The basis of AAC is an algorithm of computing approximate greatest common divisors (AppGCD) proposed by T. Sasaki and the author. AppGCD and its applications work well, especially, for obtaining accurate results of ill-conditioned problems. Algorithms and implementation methods of AppGCD are briefly surveyed and its applications such as hybrid integral, hybrid rational function approximation (HRFA), data smoothing by using HRFA and new hybrid method for computing Cauchy principal value integral are described. Further, a pathological feature of HRFA and relations of HRFA and ill-conditioned problems, and their applications are discussed.

Mathematics Subject Classification (2000). Primary 68W30; Secondary 33F10.

Keywords. Hybrid computation, ill-conditioned property, algebraic equation, rational interpolation, quadrature.

1. Introduction

Traditionally, the word **scientific computation** means computation done numerically, i.e., numerical computation. The successes of numerical computation have brought about today's development of the *Computer World*. Numerical computations have a very wide application area, and a number of research works on applied mathematics have also been proposed. Further, this research has supported current developments of the so-called engineering society. However, as is well known, these developments have some defects. They include:

1. the results are always in danger of numeric error,
2. since a number of algorithms are proposed, problems related to algorithm selection occur, and
3. algebraic or symbolic computations are impossible.

On the other hand, rapid progress of computer hardware and software has led to the use of symbolic computation. Symbolic computation wastes a significant amount of computer memory and computing time as well. Computations done symbolically give exact results. Thus, if numerical computation is effectively combined with symbolic computation, some problems in numerical computation may be solved more quickly and accurately.

Many approaches have been used for combining numerical computation with symbolic computation. Some of them are shown by Kaltofen *et al.* [13, 4]. One of the candidates for such type of computation that may be used effectively is the problem of solving ill-conditioned polynomial equations. A polynomial is said to be ill-conditioned if small changes in its coefficients result in large changes in its zeros. An ill-conditioned polynomial equation has at least one of the following properties:

1. the existence of several roots having ratios close to unity,
2. the existence of multiple roots.

The first property means the existence of close zeros in a polynomial equation. For multiple roots, let the polynomial equation be

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n = 0\,, \quad a_n \neq 0,$$

and let m_j be the multiplicity of a solution x_j of $P_n(x) = 0$. If a coefficient a_k is perturbed slightly to $a_k + \Delta a_k$, then the perturbation in x_j is

$$x_j = \left\{ -\frac{m_j! x_j^{n-k} \Delta a_k}{(d/dx)^{m_j} P_n(x_j)} \right\}^{1/m_j},$$

where $\Delta a_k \ll a_k$ [2]. Many numerical algorithms have been proposed to obtain zeros effectively. However, most of them are not effective for ill-conditioned problems.

If the polynomial $P_n(x)$ has integer coefficients and its roots are integer, then multiple roots are easily separated by symbolic computation. For s-fold multiple roots, $P_n(x)$ is divided by $d^{(s)}P_n(x)/dx^s$, with residual equal to zero. It follows that the greatest common divisor (GCD) of $P_n(x)$ and its $(s-1)$-times differentiation by x are not primitive. The GCD of two univariate polynomials P_1 and P_2 with exact coefficients, $\mathrm{GCD}(P_1, P_2)$, is obtained by algebraic Euclidean algorithm. Sasaki and the author applied the Euclidean algorithm to polynomials whose coefficients are inexact, i.e., with limited accuracy or perturbed within small tolerance. In this case, GCD is replaced by approximate GCD (AppGCD) with accuracy ε, $\mathrm{AppGCD}(P_1, P_2; \varepsilon)$ [26, 21]. There are two simple approaches to computing AppGCD for polynomials with inexact coefficients. One is the method known as AppGCD (with accuracy ε) and the other is the method using interval arithmetic [20]. These two methods will be briefly described in Sect. 2. AppGCD has been applied to several kinds of scientific computations. The first is an integral of a given ill-behaved function which is obtained in the symbolic-numeric combined environment. AppGCD plays an important role in the integration procedure. The

method is called *hybrid integral* and will be described in Sect. 3. Section 4 is devoted to hybrid rational function approximation, which is abbreviated simply as HRFA. In HRFA, AppGCD is effectively used. If a given function or set of data is approximated by a rational function, there often occur approximate common factors in numerator and denominator polynomials of the rational function. These approximate common factors are eliminated by the AppGCD of the numerator and denominator polynomials. Hybrid algorithms are applied to some practical problems such as data smoothing and Cauchy principal value integral. These applications will be shown in Sect. 5.

2. Approximate GCD Computations

Two implementation methods for AppGCD are discussed. One is known as AppGCD with accuracy ε, which is the basis of every kind of AppGCD proposal. The other is AppGCD using interval arithmetic. After the successful application of AppGCD for solving an ill-conditioned algebraic equation, other AppGCDs have been proposed by several researchers from different viewpoints. They are also summarized briefly.

2.1. Approximate GCD with Accuracy ε

The AppGCD of two polynomials P_1 and P_2 with accuracy ε, $\mathrm{AppGCD}(P_1, P_2; \varepsilon)$, is a natural extension of the usual GCD computation by Euclidean algorithm. Here, coefficients of both polynomials P_1 and P_2 are inexact and are represented by floating point numbers. The polynomial remainder sequence (PRS) is obtained by the Euclidean algorithm as follows:

$$P_{i-1} = P_i Q_i + P_{i+1}, \qquad i = 2, \ldots,$$

where Q_i is a quotient polynomial. Because coefficients are inexact, all coefficients of the polynomials in the PRS contain error by division operations. A cutoff operation which regards the coefficients of polynomials in the PRS smaller than a tolerance ε as zero is introduced. Thus the PRS is written as

$$P_1, P_2, P_3, \ldots, P_k \neq 0, P_{k+1} = 0 \text{ (cutoff } \varepsilon).$$

The approximate GCD with accuracy ε is defined as

$$\mathrm{AppGCD}(P_1, P_2; \varepsilon) = P_k.$$

The above procedures are shown in an algorithm below:

Algorithm 1: Approximate GCD Algorithm

Input: Univariate regular polynomials $P_1(x)$ and $P_2(x)$ with $\deg(P_1) > \deg(P_2)$ and a small positive number ε.

Output: Approximate GCD of $P_1(x)$ and $P_2(x)$ with accuracy less than ε, $\mathrm{AppGCD}(P_1, P_2, \varepsilon)$.

Algorithm:

1. Calculate a PRS

$$P_1,\ P_2, \ldots, P_k \neq 0 \text{ (cutoff } \varepsilon),\ P_{k+1} = 0 \text{ (cutoff } \varepsilon)$$

by the iteration formula

$$\begin{aligned} Q_i &= \text{quotient}(P_{i-1}, P_i)\,; \\ P_{i-1} &= Q_i P_i + \max\{1, \text{mmc}(Q_i)\} \times P_{i+1}, \quad i = 2, \ldots, k\,, \end{aligned}$$

where mmc denotes the maximum magnitude coefficient of the polynomial.

2. Return the primitive part (pp) of P_k as

$$\text{AppGCD}(P_1(x), P_2(x); \varepsilon) = \text{pp}(P_k).$$

The univariate AppGCD has been used to solve ill-conditioned polynomial equations (as shown in detail in [26]) and extended to obtain the AppGCD of multivariate polynomials [22, 21].

2.2. Approximate GCD by Interval Arithmetic

The PRS is computed by using the circular interval arithmetic. An inexact floating-point number is considered to be a pair of its center, M, and an error radius, $r(M)$, which corresponds to a perturbation with small tolerance in a circular interval number. Thus the circular interval number is represented as $< M, r(M) >$. The circular interval arithmetic is defined as arithmetic between circular interval numbers. The PRS computation is similar to that of AppGCD with accuracy ε. The difference is on a stopping criterion of the PRS. In AppGCD with ε, the cutoff operation is introduced to regard small coefficients as zero. However, in the interval arithmetic, the condition that the circle contains zero is used as the criterion. There arises a difficulty on division by an interval number. The fact that the denominator must not include zero in the interval arithmetic. In the circular interval arithmetic case, if the condition

$$\mid M \mid < \mid r(M) \mid$$

shows the circular interval contains zero. Thus the PRS is stopped at a polynomial whose coefficient satisfies the above condition. Later, Shirayanagi and Sweedler discussed the condition in detail with rigorous mathematical proofs by using the rectangular interval arithmetic and established a theory to stabilize algebraic algorithms [30].

2.3. Computation Examples of Both AppGCDs

Consider the polynomial equation

$$P(x) = x^4 - 10.4x^3 - 70.96x^2 + 29.6x - 3 = 0.$$

It has roots $x = -5, 15$ and a double root at $x = 0.2$. The polynomial equation is ill-conditioned because of its double root. The two methods mentioned above may

be applied to solve the equation. The Eucledian algorithm with inexact coefficients gives the following PRS:

$$
\begin{aligned}
P_1 &= x^4 - 10.4x^3 - 70.96x^2 + 29.6x - 3,\\
P_2 &= 4x^3 - 31.2x^2 - 141.92x + 29.6 \quad (= dP_1/dx),\\
P_3 &= -85.78x^2 - 107.76614385x + 24.98461538,\\
P_4 &= -0.46563934x + 0.09312787,\\
P_5 &= 2.77555756 \times 10^{-17}.
\end{aligned}
$$

The maximal value of P_5 is less than 10^{-16} times than that of P_4. Thus, the cutoff operation regards P_5 as zero. After a normalization of the head term of P_4, the AppGCD with accuracy $\varepsilon = 10^{-16}$ is obtained and shown as

$$\mathrm{AppGCD}(P_1, P_2; 10^{-16}) = x - 0.2.$$

On the other hand, the PRS generated by the interval arithmetic shows

$$
\begin{aligned}
P_1 =& < 1.0, 2.2 \times 10^{-16} > x^4 + < -10.4, 1.8 \times 10^{-15} > x^3\\
&+ <-70.96, 1.4\times 10^{-14}> x^2 + <29.6, 3.6\times 10^{-15}> x + <-3.0, 4.4\times 10^{-16}>,\\
P_2 =& < 4.0, 1.8 \times 10^{-15} > x^3 + < -31.2, 8.9 \times 10^{-15} > x^2\\
&+ < -141.92, 5.7 \times 10^{-14} > x + < 29.6, 7.1 \times 10^{-15} >,\\
P_3 =& < -8.9 \times 10^2, 3.8 \times 10^{-12} > x^2 + < -1.1 \times 10^3, 9.3 \times 10^{-12} > x\\
&+ < 2.6 \times 10^2, 1.8 \times 10^{-12} >,\\
P_4 =& < -4.7 \times 10^6, 1.1 \times 10^{-7} > x + < 9.5 \times 10^5, 2.2 \times 10^{-8} >,\\
P_5 =& < 1.3 \times 10^{-6}, 7.8 \times 10^{-4} > \ni 0.
\end{aligned}
$$

PRS computations terminate because P_5 contains zero in its interval. By the algorithm stabilization technique [30], P_5 should be rewritten to zero (zero rewriting). Then, we obtain

$$\mathrm{GCD}(P_1, P_2) = \mathrm{GCD}(P(x), dP(x)/dx) = x - < 0.2, 4.7 \times 10^{-10} > .$$

Both GCDs give approximate double root of $P(x)$ that exists very close to $x = 0.2$. Many kinds of algorithms to obtain approximate GCDs have been proposed. It follows that approximate GCD and its applications become one of the most interesting research subjects of computer algebra.

2.4. Solving Ill-conditioned Algebraic Equation

The first success of applications of AppGCD is to solve a univariate ill-conditioned algebraic equation. A polynomial equation, $P(x) = 0$, is called *ill-conditioned* when the equation has multiple and/or close roots as mentioned in Sect. 1. If the coefficients of $P(x)$ are integers or rational numbers, say *exact*, multiple roots are easily found and separated by the traditional GCD operation based on the Euclidean

algorithm. However, if coefficients are *inexact* and represented as floating-point numbers, it becomes impossible to use traditional GCD and then, the equation should be solved numerically. But, in numerical computation, close roots reduce the accuracy of all roots of the equation. Then, we should make an algorithm which separates not only multiple but also close roots effectively for the algebraic equation with *inexact* coefficients. This was the first motivation to propose the AppGCD algorithm [26].

The given polynomial $P(x)$ is, first, decomposed to a square-free form by using AppGCD with accuracy ε. The square-free decomposition of $P(x)$ is written as

$$P(x) = Q_1(x)Q_2^2(x)\cdots Q_m^m(x), \tag{2.1}$$

where $Q_m^m(x)$ contains all the m-multiple factors of $P(x)$; hence each Q_i has no multiple factor. To obtain (2.1), the following method is used:

$$\mathrm{AppGCD}\left(P(x), \frac{dP(x)}{dx}, \varepsilon\right) = Q_2(x)Q_3^2(x)\cdots Q_m^{m-1}(x). \tag{2.2}$$

Dividing (2.1) by (2.2), the product of square-free factors are obtained as

$$Q_1(x)Q_2(x)\cdots Q_m(x).$$

With repeated use of the above procedure, $P(x)$ can be separated to $Q_1, Q_2, \ldots, Q_l$. The procedure by using AppGCD may be called *approximate square-free decomposition*.

After the square-free decomposition, for each $Q_l(x),\ l = 1, \ldots, m$, $Q_l(x) = 0$ is solved by rough, numerical computation. Let the roots of the computation be $u_{l_1}, u_{l_2}, \ldots$. For each u in $u_{l_1}, u_{l_2}, \ldots$, expand $P(u+v)$ up to v^l terms by Taylor expansion as

$$P(u) + P^u\frac{v}{1!} + \cdots + P^l(u)\frac{v^l}{l!} + O(v^{l+1}) = 0, \tag{2.3}$$

where

$$P^{(l)}(u) = \frac{d^l P(x)}{dx^l}\ |_{x=u}\ .$$

The expansion (2.3) gives an equation on v, $p(v) = 0$. Since the degree of the equation is l, l solutions, $v_1, \ldots, v_l$, may be obtained. By adding these l solutions to the solution of $Q_l(x) = 0$, all roots of $P(x) = 0$ are obtained with high accuracy. We show an example of how to solve the univariate ill-conditioned algebraic equation by the method mentioned above [21]. Let an algebraic equation be

$$\begin{aligned} P(x) &= x^7 - 3.504x^6 + 0.762003x^5 + 6.87799x^4 - 4.02601x^3 \\ &\quad - 2.62198x^2 + 2.51201x - 0.504006 \\ &= (x+1)^2(x-2)^2(x-0.5)(x-0.501)(x-0.503) = 0. \end{aligned} \tag{2.4}$$

The equation (2.4) has two double roots at $x = -1$ and $x = 2$, and also three close roots around $x \simeq 0.5$. Our method is divided into three steps.

1. Obtain the AppGCD with accuracy ε
PRS is obtained as

$$P_1 = P(x),$$
$$P_2 = \frac{dP(x)}{dx} = 7x^6 - 21.024x^5 + 3.81001x^4 + 27.512x^3 - 12.078x^2 - 5.24397x + 2.51201,$$
$$P_3 = -1.28572x^5 + 3.22017x^4 - 0.333188x^3 - 2.73655x^2 + 1.77815x - 0.324372,$$
$$P_4 = 1.23979x^4 - 2.48289x^3 - 0.924883x^2 + 2.17459x - 0.623204,$$
$$P_5 = -3.37499 \times 10^{-6}x^3 + 5.06537 \times 10^{-6}x^2 + 5.505959 \times 10^{-6} - 3.38077 \times 10^{6}$$
$$\simeq 0 \text{ cutoff } 10^{-4}.$$

Thus the AppGCD with $\varepsilon = 10^{-4}$ is obtained as

$$\begin{aligned}\mathrm{AppGCD}(P, P', 10^{-4}) &= \mathrm{pp}(P_4) \\ &= x^4 - 2.0026667x^3 - 0.74599899x^2 + 1.7539990x - 0.50266867,\end{aligned}$$

where P' denotes $dP(x)/dx$.

2. Approximate square-free decomposition
By using the AppGCD obtained above, the following square-free factors are computed:

$$Q_3(x) = x - 0.50133334, \quad Q_2(x) = x^2 - x - 2.0.$$

Then $P(x)$ is decomposed as

$$P(x) = Q_2^2(x)Q_3^3(x) = (x^2 - x - 2.0)^2(x - 0.50133334)^3. \tag{2.5}$$

Roots of Q_2 are easily obtained as $x = u_{2_1} = -1$ and $x = u_{2_2} = 2$. On the other hand, a common factor Q_3 gives $x = u_3 = 0.50133334$. These solutions may correspond to two double roots and close roots of $P(x) = 0$.

3. Expand procedure of close roots
The remaining problem is to obtain the detailed behavior of $P(x)$ at $x \simeq -1, 2$ and 0.50133334. The first, a root $x = -1$, is considered. $P(x + v)$ is expanded up to v^2 at $x = -1$ and one obtains the result

$$v^2 - 1.16651 \times 10^{-16}v + \text{small term } (\ll 10^{-16}) = 0.$$

Thus, $v = 0$ is obtained with a sufficient accuracy. Thus, the root $x = -1$ is a double root of $P(x) = 0$. Similar procedure may be applied to the root $x = 2$. Then, the root $x = 2$ is also a double root of $P(x) = 0$. The next, the approximate triple factor $Q_3(x)$, is considered. Taylor expansion of $P(x + v)$ at $x = u_3$ up to v^3 gives

$$5.0625v^3 + 1.32583 \times 10^{-7}v^2 - 1.18125 \times 10^{-5} - 3.75007 \times 10^{-9} = 0.$$

Roots of this equation are

$$v_1 = -0.00133334, \quad v_2 = -0.00033334 \quad \text{and} \quad v_3 = 0.001666666.$$

By adding these values to u_3, the following results are obtained:

$$x = 0.5, \quad x = 0.501 \quad \text{and} \quad x = 0.503.$$

They are the three close roots of $P(x) = 0$ with high accuracy.

The method above can be straightly applied to solve a system of multivariate algebraic equations. In this case, coefficients of equations are also *inexact* and the system is ill-conditioned. Detailed discussions are described in the papers [22, 21].

2.5. Other Approximate GCDs

Before the proposal of the paper by Sasaki and the author [26], similar approaches have been discussed by Schönhage [28], Auzinger and Stetter[1]. However, the approach in the former study is considerably different from ours, and the discussions in the paper are devoted mostly to the computation time complexity of the algorithm. Further, coefficients of input polynomials are assumed to be arbitrarily precise, i.e., belonging to *exact*. The authors of the latter paper describe a method of solving a system of algebraic equations by using a numerical resultant algorithm.

Thus, in this paper, six different approaches for the approximate GCD are briefly reviewed.

Approximate GCD proposed by Sederberg and Chang [29]**:** The algorithm is considered and applied as a numeric-symbolic algorithm to a problem of computer-aided design. Let $P_1(x), P_2(x), \ldots, P_n(x)$ be polynomials. The set of polynomials is perturbed so as to induce a linear common factor. That is, for a set of the perturbation polynomials $\varepsilon(x) = \varepsilon_1(x), \ldots, \varepsilon_n(x)$, $\gcd(P_1(x) + \varepsilon_1(x), P_2(x) + \varepsilon_2(x), \ldots, P_n(x) + \varepsilon_n(x))$ induce a linear polynomial, common factor. A norm of perturbation polynomial is written as

$$\|\varepsilon\|_{[a,b]} = \max_{a \le x \le b} \sqrt{\sum_{i=1}^{n} \varepsilon_i^2(x)}$$

over a prescribed parameter interval $[a, b]$. If we represent $\varepsilon_i(x)$ by the Chebyshev polynomials, we can determine ε which gives the minimum norm, uniquely. The result is applied to the problem of approximating high degree curves by small degree ones.

Approximate GCD proposed by Corless, Gianni, Trager and Watt [3]**:** For two given univariate polynomials $P_1(x)$ and $P_2(x)$, write the Sylvester matrix of the coefficients of $P_1(x)$ and $P_2(x)$. The Singular Value Decomposition (SVD) technique is applied to the matrix to compute an upper bound on the degree of the approximate GCD. By SVD, if the matrix becomes rank deficient by one, then there exists a common factor of both polynomials. Further, the number of rank deficient rows shows the degree of the GCD. The method is applied to a multivariate polynomial case. In the paper [3], an optimization problem is first used on the problem of obtaining approximate GCD. The Euclidean norm (2-norm) is used and this selection of the norm is in the paper.

Approximate GCD proposed by Karmarkar and Lakshman [14]**:** The approach to find the approximate GCD is similar to the above-mentioned method proposed by Corless *et al.* [3]. Two monic polynomials $P_1(x), P_2(x) \in C[x]$ with $\deg(P_1) = m, \deg(P_2) = n$, where $C[x]$ denotes a polynomial with complex coefficients, are considered. The problem is stated as to find polynomials $\hat{P}_1, \hat{P}_2 \in C[x]$ with $\deg(\hat{P}_1) < m, \deg(\hat{P}_2) < n$ such that $P_1 + \hat{P}_1$ and $P_2 + \hat{P}_2$ have a non-trivial GCD, and $\|\hat{P}_1\| + \|\hat{P}_2\|$ is minimized. The norm used here is also the Euclidean norm. The running time of the algorithm is a polynomial of the degrees m, n. Further, a concept of the nearest singular polynomial to P_1, that is a polynomial $h \in C[x]$ with $\deg(h) = m$ such that h has a double root and $\|\hat{P}_1\|$ is minimized, where $\hat{P}_1 = h - P_1$, is proposed. Many works on the nearest singular polynomial have been considered.

Approximate GCD proposed by Hribernig and Stetter [7]**:** The approximate GCD is here called the *near-GCD*. For two given polynomials $P_1, P_2 \in C[x]$, at the accuracy level α, the polynomials possesss a near-GCD $\tilde{g}$ if there exist polynomials $P_1^*, P_2^* \in C[x]$ satisfying

$$\mathrm{GCD}(P_1^*, P_2^*) = \tilde{g}, \quad \text{and} \quad \|\tilde{P}_i - P_i^*\| \leq \alpha, \quad i = 1, 2.$$

Equivalently, a near-GCD $\tilde{g}$ of $\tilde{P}_1$ and $\tilde{P}_2$ at accuracy level α is denoted as α-$\mathrm{GCD}(\tilde{P}_1, \tilde{P}_2)$ and computed via the Euclidean algorithm. Here, as the norm of the polynomial, the 1-norm is used. The algorithm gives a lower bound on the degree of the approximate GCD.

Approximate GCD proposed by Emiris, Galligo and Lombardi [5]**:** Several Approximate GCDs of two univariate polynomials obtained by the above algorithms depend on the rounding mode or the accuracy ε selected for floating-point computations. Further, as mentioned above, algorithms based on Euclidean algorithm give the lower bound of the degree of the approximate GCD. The purpose of the algorithm, developed here, is to obtain the maximum-degree approximate GCD depending on the given tolerance ε. For two given polynomials $P_1, P_2 \in C[x]$, whose degree is n and m $(n \geq m)$ respectively, and torelance $\varepsilon \in (0, 1]$, upper bound on the degree of the approximate GCD, ε-GCD of P_1, P_2, are obtained. The degree of the ε-GCD is defined to be the maximum integer r such that there exist $\hat{P}_1, \hat{P}_2 \in C[x]$ of degree bounded by n and m, respectively, with $\mid P_1 - \hat{P}_1 \mid, \mid P_2 - \hat{(P_2)} \mid \leq \varepsilon$ and $\deg(\mathrm{GCD}(\hat{P}_1, \hat{P}_2)) = r$. To guarantee the maximum-degree approximate GCD, SVD computations on subresultant matrices and a gap theorem are effectively used. The norm used here is the Euclidean norm (2-norm).

Approximate GCD proposed by Pan [23]**:** Several methods for computing the approximate GCD have been proposed by Pan. Here, for $P_1(x)$ and $P_2(x)$, polynomials P_1^* and P_2^* are considered. The approximate GCD of P_1^* and P_2^*, $\mathrm{GCD}(P_1^*, P_2^*)$, satisfies the following relations for a real value b:

$$\deg(P_1^*) \leq \deg(P_1(x)), \qquad \deg(P_2^*) \leq \deg(P_2(x)), \tag{2.6}$$

$$\|P_1^* - P_1(x)\| \leq 2^{-b}\|P_1(x) \qquad \|, \|P_2^* - P_2(x)\| \leq 2^{-b}\|P_2(x)\|.$$

The maximum δ-GCD satisfies the relations in (2.6). Let the two given polynomials be

$$P_1(x) = u \times \prod_{i=1}^{n}(x - y_i), \quad \text{and} \quad P_2(x) = v \times \prod_{j=1}^{m}(x - z_j),$$

where u and v are the leading coefficients of $P_1(x)$ and $P_2(x)$, respectively. Thus, y_i and z_j are zeros of $P_1(x)$ and $P_2(x)$. Further, a small constant δ is given. The maximum δ-GCD $\tilde{g}_\delta(x)$ of $P_1(x)$ and $P_2(x)$ is defined as

$$\tilde{g}_\delta(x) = \prod_{k=1}^{r}(x - x_k), \qquad x_k = \frac{y_{i_k} + z_{j_k}}{2}, \quad k = 1, \ldots, r,$$

where the set of pairs $(y_{i_1}, z_{j_1}), \ldots, (y_{i_r}, z_{j_r})$ is a maximum matching which maximizes r and satisfies $\mid y_{i_k} - z_{j_k} \mid \leq 2\delta$. If δ is bounded by $\delta \leq (1 + 2^{-b})^{1/n} - 1$, then the polynomials $P_1^* = P_1(x)\tilde{g}_\delta(x)$ and $P_2^* = P_2(x)\tilde{g}_\delta(x)$ satisfy (2.6). The maximal norm is used to define the maximum δ-GCD.

After successful introduction of approximate GCDs as described above, many algorithms for computing approximate GCDs have been proposed. There are also new developments on approximate algebraic computation such as computing nearest singular polynomials and approximate factorization. Most of the algorithms are based on AppGCD. They have been published in recent proceedings of international conferences on computer algebra and are summarized in [13, 4].

3. Approximate GCD and Hybrid Integral

Obtaining an indefinite integral of a given function is one of the most important operations in scientific computation. If the function is given by a rational function with exact coefficients, it can be integrated symbolically. There remains, however, several cases such as (1) even if the function is given, a closed form solution is not obtainable, (2) the function is defined by a table or a set of discrete data, and (3) the function has inexact floating-point coefficients with limited accuracy. Numerical algorithms give only numerical results for definite integrals. Here we show an algorithm of hybrid integral in which numerical methods are effectively combined with an algorithm of symbolic (algebraic) integral. The hybrid integral algorithm gives symbolic results, a kind of *indefinite integral*, for given functions. Accurate numerical results of definite integrals are easily obtained by simple substitutions of upper and lower bounds of integrals into symbolic results. The algorithm proposed here is an extension of an algebraic algorithm of integration for rational function by Horowitz [6] to the *inexact* coefficients case. A rough sketch of the algorithm of the hybrid integral is given below.

Algorithm 2: Hybrid Integral Algorithm

Input: Univariate rational function $f(x) = P(x)/Q(x)$ whose coefficients are known with limited accuracy and represented by floating-point

numbers. Here, the approximate GCD of the two input polynomials $P(x)$ and $Q(x)$ is reduced to 1 and $\deg(P) < \deg(Q)$.

Output: Approximate indefinite integral of $f(x)$.

Algorithm:

1. Decompose $f(x)$ into rational and transcendental parts:

$$\int f(x)dx = \frac{s(x)}{t(x)} + \int \frac{p(x)}{q(x)}dx.$$

In this step, AppGCD is used. If $\mathrm{AppGCD}(Q(x), dQ(x)/dx; \varepsilon) = 1$, then $s(x)/t(x) = 0$.

2. Integrate the transcendental part $p(x)/q(x)$.
   a) Determine all zeros of $q(x)$ by numerical Durand-Kerner method. Since the coefficients of $q(x)$ are real, zeros are limited as real or complex conjugate pairs and written for $m + 2n = \deg(q)$ as real zeros $a_1, a_2, \ldots, a_m$, or complex conjugate zeros $b_1 \pm ic_1, \ldots, b_n \pm ic_n$.
   b) Decompose into a partial fraction

$$\frac{p}{q} = \sum_{k=1}^{m} \frac{e_k}{x - a_k} + \sum_{k=1}^{n} \frac{2(f_k x - b_k f_k - c_k g_k)}{x^2 - 2b_k x + b_k^2 + c_k^2},$$

where e_k, f_k and g_k are determined by the residue theory as follows:
   - Let $r(x) = p/q'$, where $q' = dq(x)/dx$,
   - for real zeros, $e_k = r(a_k)$;
   - for complex conjugate zeros,

$$f_k = \Re\{r(b_k + ic_k)\} \quad \text{and} \quad g_k = \Im\{r(b_k + ic_k)\},$$

where $\Re$ and $\Im$ represent real part and imaginary part, respectively.

   c) Substitute two well-known formulas of logarithmic integrals

$$\int \frac{p}{q}dx = \sum_{k=1}^{m} e_k \log \mid x - a_k \mid + \sum_{k=1}^{n} f_k \log \mid x^2 - 2b_k x + b_k^2 + c_k^2 \mid -2g_k \tan^{-1}\left(\frac{x}{c_k} - \frac{b_k}{c_k}\right).$$

Some properties of the algorithm are:

- An *indefinite integral* for a given function with floating-point coefficients is obtained.
- Accurate value of a definite integral is obtained only by substitutions of upper and lower bounds of integral.
- Errors caused by the algorithm are reduced to errors contained in numerical root finding process (Durand-Kerner method) and can be estimated by the Smith theorem.

Results obtained by the hybrid integral algorithm are compared with well-known and widely used numerical integration methods, such as Gaussian quadrature of 32 points (Gauss32), Double-Exponential formula (DE), the Romberg method (Romb) and adaptable Newton-Cotes method (NC). Comparisons are done for three ill-conditioned rational functions:

1. A singularity outside the integral region but close to both ends,
$$I_1 = \int_0^1 \frac{dx}{1000x(x-1) - 0.001}.$$
2. The integrand has a sharp peak in the integral region,
$$I_2 = \int_0^1 \frac{dx}{1000(x-0.5)^2 + 0.001}.$$
3. The integrand has both properties I_1 and I_2,
$$I_3 = \int_0^1 \frac{dx}{x^5 - x^4 - 0.75x^3 + x^2 - 0.25x - 10^{-6}}.$$

Results are shown in Table 1. To obtain accurate value of each integral, I_1, I_2 and I_3, Noda *et al.* [17, 18, 19] used a quadrature by parts. Each integrand which changes rapidly is carefully divided into small but smoothly changing parts and integrated numerically. These values of small integrals are added and accurate result for the integrand is obtained.[1] Results are the same as the results obtained by the hybrid integral algorithm. Thus, it can be said that the hybrid integral algorithm gives better results than well-known numerical methods. Especially it works well for ill-conditioned integrals.

TABLE 1. Comparisons of results of the hybrid integral algorithm and numerical methods

	Hybrid	Numerical Integration			
		G32	DE	Romb	NC
I_1	-0.02763097	-0.01622972	-0.02763097	-0.68482819	5.40966656
I_2	3.13759266	0.19994034	24.6736125	2.98225113	3.13759258
I_3	-5195.24497	-580.39410	-24965.007	-5759.10716	230.75544

The key strategy of the hybrid integral algorithm is to decompose into a partial fraction by computing all zeros of a denominator polynomial of a rational part of the given integrand numerically. For an integral with a parameter or a double integral, the algorithm described above may be easily applied. Here, a denominator of a rational part of an integrand with the bivariate case may be decomposed into a partial fraction by using an approximate factorization algorithm such as [27, 12]. Thus, a kind of *indefinite integral* of the bivariate integral may be obtained.

[1]Current powerful CAS such as Maple 9 or its successor gives the same results and assures the correctness of the hybrid integral algorithm.

4. Hybrid Rational Function Approximation

For rational approximation of a given function or a set of discrete data, the rational interpolation, especially the so-called *naïve rational interpolation*, may be one of the simplest methods. The function is first evaluated at several data points in an interval and changed to the set of discrete data. The set of discrete data is interpolated to a rational function. If the interpolation is done with floating-point computation, pathological features have been observed by Noda *et al.* [18]. To avoid the feature, the AppGCD algorithm is effectively used. Then the rational interpolation approximates a given function or a set of discrete data accurately. We refer to the algorithm as the Hybrid Rational Function Approximation (HRFA). Hereafter, it is simply called HRFA. We introduce HRFA briefly and discuss its theoretical considerations of pathological features.

4.1. Rational Interpolation and Pathological Feature

A rational interpolation is defined as a ratio of numerator and denominator polynomials as

$$r_{m,n}(x) = \frac{p_m(x)}{q_n(x)} = \frac{\sum_{i=0}^{m} a_i x^i}{1 + \sum_{i=1}^{n} b_i x^i} = \frac{a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m}{1 + b_0 x + b_1 x^2 + \cdots + b_n x^n}. \quad (4.1)$$

It interpolates a function $f(x)$ or a set of discrete data on a segment $[x_0, x_{m+n}]$. The rational interpolant (4.1) is called (m, n) *rational interpolant*. Here, the naïve rational interpolation is considered. For $m + n + 1$ $(= N)$ discrete points, $x_0 < x_1 < \cdots < x_{m+n}$, values $f(x_k) = f_k$ $(k = 0, \ldots, m + n)$ are evaluated. Then $m + n + 1$ $(= N)$ simultaneous linear equations

$$\sum_{i=0}^{m} a_i x_k^i - f_k \sum_{i=i}^{n} b_i k_k^i = f_k, \quad k = 0, 1, \ldots, m + n, \quad (4.2)$$

are obtained. The inexact, floating-point coefficients a_i, b_i of the rational interpolant are then determined by solving the system by Gaussian elimination. Noda *et al.* [18] discussed the problem and found that

1. even if a continuous function is interpolated, the denominator of the rational interpolant may have a zero, and the zero causes an *undesired pole*,
2. except for the above zero and pole, the rational interpolant gives accurate approximation of $f(x)$.

Further, the fact that the zero of the numerator polynomial may arise which is very close to the undesired pole has been shown by Kai *et al.* [11] through numerical experiments. The zero and pole, mentioned above, are referred to as *undesired zero and pole*. They also describe the appearance of approximate common factors in the numerator and denominator polynomials. These factors are caused by *undesired zero and pole* and eliminated from the rational interpolant by using HRFA.

The approximately common factor which causes a pathological feature of the rational interpolant is eliminated by using AppGCD. The procedure HRFA

ensures high-quality approximation without *undesired zero and pole* for any precision computations. The algorithm of HRFA is shown below. The details of the method and its accuracy are discussed by Noda and Kai [18, 8, 11].

Algorithm 3: HRFA Algorithm

Input: Rational interpolant (naïve rational interpolation)

$$r_{m,n}(x) = \frac{p_m(x)}{q_n(x)}.$$

Output: Reduced rational interpolant without singularities

$$\tilde{r}(x) = \frac{\tilde{p}(x)}{\tilde{q}(x)}.$$

Algorithm:

1. $\mathrm{AppGCD}(p_m(x), q_n(x)) = g(x)$.
2. $\tilde{r}(x) = \dfrac{p_m(x)/g(x)}{q_n(x)/g(x)}$

We show the pathological feature of the rational interpolant and how Algorithm 3 works well using an example. Suppose that $r_{4,4}(x)$ is a rational interpolant of the function $f(x) = \log(x+2)$; we obtained the following rational interpolant with a single precision floating-point computation:

$$\begin{aligned} r_{4,4}(x) &\simeq \frac{0.6931 + 214.8599x + 318.6295x^2 + 113.5897x^3 + 9.1269x^4}{1 + 309.2559x + 236.7844x^2 + 48.7819x^3 + 2.1131x^4} \\ &\simeq 4.3195\,\frac{(x+8.77124)(x+2.6710)(x+1)(x+0.0032415)}{(x+16.999)(x+3.8532)(x+2.2286)(x+0.0032416)}\,. \end{aligned}$$

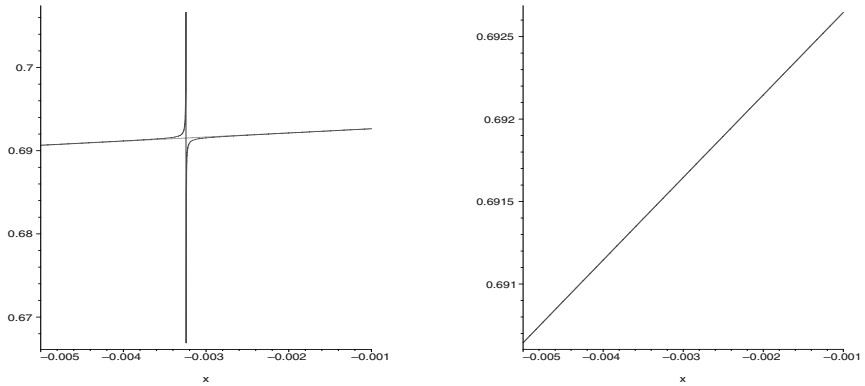
The last terms of the numerator and denominator polynomials, $x + 0.0032415$ and $x + 0.0032416$ respectively, show an approximate common factor. The rational interpolant $r_{4,4}(x)$ causes *undesired zero and pole* by these terms and shown in Fig. 1(a). However, except for a small interval containing *undesired zero and pole*, the rational interpolant constructs an accurate approximation of the function $f(x)$. The HRFA successfully removes the *undesired zero and pole* by taking the AppGCD of $p_m(x)$ and $q_n(x)$. The AppGCD $g(x)$ of the numerator and denominator polynomials of $r_{4,4}(x)$ is computed as

$$g(x) \simeq x + 0.0032423543\,.$$

Then, after dividing $r_{4,4}(x)$ by $g(x)$, a reduced rational function approximation, $R_{\mathrm{HRFA}}(x)$, is obtained as follows:

$$\begin{aligned} R_{\mathrm{HRFA}}(x) &\simeq \frac{9.1269605x^3 + 113.56010x^2 + 318.26127x + 213.82796}{2.1131867x^3 + 48.775051x^2 + 236.62620x + 308.48869} \\ &= 4.3190507\,\frac{(x+8.7712482)(x+2.6710225)(x+0.99999856)}{(x+16.999383)(x+3.8532500)(x+2.2286457)}\,. \end{aligned}$$

The reduced rational function $R_{\mathrm{HRFA}}(x)$ does not have *undesired zero and pole*, and is shown in Fig. 1(b).



(a) $r_{4,4}(x)$ (b) $R_{\mathrm{HRFA}}(x)$

FIGURE 1. Rational function approximation of $\log(x+2)$ by $r_{4,4}(x)$

4.2. Rational Interpolation and Ill-conditioned Property

Rational functions discussed in the literature [24, 25] are restricted to be irreducible, i.e., the numerator and denominator polynomials have no common factors other than a constant. Litvinov [15] discussed an interpolation of given functions by a rational function and mentioned that the system of linear equations for the rational interpolant turns out to be ill-conditioned in many cases. Although the system itself is ill-conditioned, the fact that the rational interpolant ensures accurate approximation by means of the best approximation is also described. However, Litvinov does not refer to the importance of the appearance of *undesired zero and pole*. On the pathological feature of the rational interpolation, the following facts are shown through numerical experiments by Kai *et al.* [8, 11].

1. An *undesired zero and pole* always appear very close and as a pair.
2. The position of the *undesired zero and pole* changes by degree of the rational function $r_{m,n}(x)$ and by digits of computations.
3. Except for a small interval where the *undesired zero and pole* exist, the rational interpolant gives accurate approximation of a given function or set of data.

Among the above facts, the second refers to the ill-conditioned property of the system of linear equations constructed by the rational interpolation method as mentioned in [15]. The system turns out to be highly ill-conditioned by error propagation during computation and the result in losses in accuracy, i.e., the system (4.2) is sensitive to the precision of computation. Thus, the position of *undesired zero and pole* changes by digits of computation without definite rules. A reason for facts 1 and 3 is discussed below.

From (4.1), we write a system of simultaneous linear equations which determines the coefficients of the rational interpolant, a_i, b_j $(i = 0, \ldots, m; j = 1, \ldots, n)$ as

$$A\mathbf{y} \;=\; B, \quad \mathbf{y} \;=\; (a_0, a_1, \ldots, a_m, b_1, b_2, \ldots, b_n)^T\,, \tag{4.3}$$

where A is a matrix whose size is $N \times N$ for $N = m + n + 1$. A triangular system, $\hat{A}\mathbf{y} \;=\; \hat{B}$, is obtained from (4.3) by Gaussian elimination. Solutions of the system, $\mathbf{y} \;=\; (y_1, y_2, \ldots, y_N)$, are obtained by back substitutions as

$$y_k \;=\; \frac{1}{\hat{A}_{k,k}}\left(\hat{B}_k - \hat{A}_{k,k+1}y_{k+1} - \hat{A}_{k,k+2}y_{k+2}\cdots - \hat{A}_{k,N}y_N\right), \quad k = N-1, \ldots, 1,$$

where $\hat{A}_{i,j}$ and $\hat{B}_j$ denotes the (i,j)-element of $\hat{A}$ and the jth-element of $\hat{B}$, respectively. Since the rational interpolant is ill-conditioned, the resulting $\hat{A}$ may have *bad row*s and become a rank deficient matrix, in many cases. Here, the *bad row* is defined as a row whose elements all take zeros or negligible small values. Thus, in usual computations, it is impossible to obtain $Y_N, Y_{N-1}, \ldots, Y_{N-\gamma+1}$ for $\hat{A}$ whose γ rows are deficient. Then we substitute undetermined symbols $t_1, \ldots, t_\gamma$ as

$$Y_N = t_\gamma, Y_{N-1} = t_{\gamma-1}, \ldots, Y_{N-\gamma+1} = t_1.$$

All elements of $\mathbf{y}$, i.e., the coefficients of the rational interpolant, are represented with undermined symbols. The rational interpolant for the rank deficient case is written as [16]

$$r_{m,n}(x) \;=\; \frac{p_0(x) + t_1p_1(x) + t_2p_2(x) + \cdots + t_\gamma p_\gamma(x)}{q_0(x) + t_1q_1(x) + \cdots + t_\gamma q_\gamma(x)}\,, \tag{4.4}$$

which approximates the given function or a set of discrete data, $f(x)$, accurately.

For the case when system (4.1) is not ill-conditioned, $\hat{A}$ becomes a full rank matrix. In this case, we should take $\gamma = 0$. Then (4.4) is written as

$$r_{m,n}(x) = \frac{p_0(x)}{q_0(x)} \;\simeq f(x) \text{ with high accuracy.}$$

It is evident that there are no *undesired zero and pole* for the $\gamma = 0$ case. Thus, the appearance of *undesired zero and pole* may be strongly related to $\gamma \geq 1$ cases. That is, the terms depend on the t's in both the numerator and denominator polynomials and may be considered to be a cause of the *undesired zero and pole*. On the other hand, Litvinov discussed a similar problem from a somewhat different viewpoint. In Litvinov's work [15], perturbations of coefficients a_i and b_j are considered. Suppose that a_i and b_j are perturbed and become $a_i + \Delta a_i$ and $b_j + \Delta b_j$. The rational interpolant $r_{m,n}(x) = p_m(x)/q_n(x)$ is perturbed to $\tilde{r}_{m,n}(x) = p_m(x) + \Delta p_m(x)/q_n(x) + \Delta q_n(x)$. Although the rational interpolant has some redundant coefficients, the following relation exists:

$$\frac{p_m(x_i)}{q_n(x_i)} \;\simeq\; \frac{\Delta p_m(x_i)}{\Delta q_n(x_i)} \;=\; f(x_i) \tag{4.5}$$

holds for arbitrary i. The function $\Delta p_m(x)/\Delta q_n(x)$ is called *error approximants* [15]. The function, *error approximants*, can also approximate $f(x)$ accurately. The fact suggests that the errors in the numerator and denominator polynomials of the rational interpolant compensate each other. Our $\gamma = 0$ case seems to agree with the $\Delta a_i = \Delta b_j = 0$ case of Litovinov's results.

For understanding the facts more clearly, we show practical examples of a rational approximation of a rational function, i.e., $f(x)$ is taken as a rational function.

4.3. Practical Example for Approximating a Rational Function

Suppose that the rational function $f(x) = 1/(x-3)$ is interpolated by a rational interpolant $r_{m,n}(x)$ in the interval $-1 \le x \le 1$. Here, the results of two types of computations, symbolic computation and numerical computation, are compared.

The case of exact computation

Case $m = 1, n = 1$ Since $r_{1,1} = (a_0 + a_1 x)/(1 + b_1 x)$, the coefficients a_0, a_1 and b_1 should be obtained by solving a system of linear equations (4.2) for a set of discrete data, $x_0 = -1, x_1 = 0, x_2 = 1$. We obtain

$$a_0 = -\frac{1}{3},\ a_1 = 0,\ b_1 = -\frac{1}{3}.$$

Then, $r_{1,1}$ gives an exact interpolation for $f(x)$ as

$$r_{1,1} = \frac{-\frac{1}{3}}{1 - \frac{1}{3}x} = \frac{1}{x-3}.$$

Case $m = 2, n = 2$ A set of discrete data

$$(x_0, x_1, x_2, x_3, x_4) := (-1, -\frac{1}{2}, 0, \frac{1}{2}, 1)$$

is used. After Gaussian elimination, $\hat{A}y = \hat{B}$ shows

$$\begin{bmatrix} 1 & -1 & 1 & -\frac{1}{4} & \frac{1}{4} \\ 0 & 1 & -\frac{2}{3} & \frac{3}{14} & -\frac{5}{14} \\ 0 & 0 & 1 & \frac{1}{14} & \frac{3}{14} \\ 0 & 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} -\frac{1}{4} \\ -\frac{1}{14} \\ -\frac{1}{42} \\ -\frac{1}{3} \\ 0 \end{bmatrix}.$$

Because the fifth row of $\hat{A}$ and the element $\hat{B}_5$ are zero (*bad row*), we should substitute $b_2 = \hat{B}_5/\hat{A}_{5,5} = 0/0 \Rightarrow t_1$ by an undetermined symbol t_1. The rational interpolant corresponds to the case of $\gamma = 1$. The coefficients of $r_{2,2}$ are then obtained as

$$a_0 = -\frac{1}{3},\ a_1 = t_1,\ a_2 = 0,\ b_1 = -\frac{1}{3} - 3t_1,\ b_2 = t_1.$$

Thus $r_{2,2}$ also interpolates $f(x)$ exactly. A common factor which depends on t_1 which appears in its numerator and denominator. $r_{2,2}$ is written as

$$r_{2,2} = \frac{-\frac{1}{3} + t_1 x}{1 - \frac{1}{3}x - 3t_1 x(1 - \frac{1}{3}x)} = \frac{1 - 3t_1 x}{(x-3)(1-3t_1 x)} = \frac{1}{x-3}.$$

Further, the following facts are shown in $r_{2,2}$:

- terms independent of t_1 give exact expression to $f(x)$,
- terms dependent on t_1 also give exact expression to $f(x)$.

Case $m = 3, n = 3$ The coefficients of $r_{3,3}$ are obtained from a set of data for $(x_0, x_1, x_2, x_3, x_4, x_5, x_6) := (-1, -\frac{2}{3}, -\frac{1}{3}, 0, \frac{1}{3}, \frac{2}{3}, 1)$. The resulting system $\hat{A}y = \hat{B}$ gives two *bad rows*. Thus the rational interpolant corresponds to the case of $\gamma = 2$ and two undetermined symbols, t_1 and t_2, should be introduced. The coefficients are written as

$$a_0 = -\frac{1}{3}, a_1 = 3t_1 + t_2, a_2 = t_1, a_3 = 0, b_1 = 9t_1 - \frac{1}{3} - 3t_2, b_2 = t_2, b_3 = t_1.$$

After substitutions of these coefficients to $r_{3,3}$, we obtain

$$r_{3,3} = \frac{-\frac{1}{3} + 3t_1 x + t_2 x + t_1 x^2}{(1 - \frac{1}{3}x)(1 - 9t_1 x(1 + \frac{1}{3}) - 3t_2 x)} = \frac{1}{x-3}.$$

The rational interpolant $r_{3,3}$ gives a satisfactory result. A second degree common factor appears in its numerator and denominator. Further, similar to the case of $r_{2,2}$, the following facts are shown:

- terms independent of t_1 and t_2 give exact expression to $f(x)$,
- terms dependent on t_1 also give exact expression to $f(x)$,
- terms dependent on t_1 also give exact expression to $f(x)$.

The case of numerical computation

The same computations as in the exact cases are done by floating-point operations. Given a set of discrete data which is the same as above, a function value $f(x_i)$ is evaluated as a floating-point number.

Case $m = 1, n = 1$ Gaussian elimination gives a triangular matrix $\hat{A}$ without *bad row* and we obtain

$$a_0 = -0.33333,\ a_1 = 1.0 \times 10^{-8},\ b_1 = -0.33334$$

as coefficients. Then, the rational interpolant becomes

$$r_{1,1} = \frac{-0.33333}{1 - 0.33334x} \simeq \frac{1}{x-3}\,.$$

The result approximates a given function $f(x)$ accurately. The small coefficient a_1 should be ignored.

Case $m = 2, n = 2$ The results $\hat{A}$ and $\hat{B}$ are obtained after Gaussian elimination as

$$\hat{A} = \begin{bmatrix} 1.0 & -1.0 & 1.0 & -0.25 & 0.25 \\ 0 & 1.0 & -0.66666 & 0.21428 & -0.35714 \\ 0 & 0 & 1.0 & 0.071428 & 0.21428 \\ 0 & 0 & 0 & 1.0 & 3.0 \\ 0 & 0 & 0 & 0 & -0.7 \times 10^{-9} \end{bmatrix}, \ \hat{B} = \begin{bmatrix} -0.25 \\ -0.071428 \\ -0.023809 \\ -0.33333 \\ -0.3 \times 10^{-9} \end{bmatrix}.$$

Different from the exact computation case, the last row of $\hat{A}$ and $\hat{B}$ contain very small values. They may be caused by errors of floating-point computations. Straightforward numerical computations give the coefficients of $r_{2,2}$ as

$$a_0 = -0.33333, a_1 = 0.43611, a_2 = -2.0 \times 10^{-9}, b_1 = -0.975, b_2 = -0.43611.$$

Although all coefficients are not reliable, the above coefficients give the rational interpolant

$$r_{2,2} = \frac{-0.33333 + 0.43611x}{1 - 0.975x - 0.43611x^2} \simeq \frac{-0.43611(x + 0.76433)}{0.43611(x - 3)(x + 0.76433)} \simeq \frac{1}{x - 3}.$$

The result is satisfactory. Approximate common factors appear in the numerator and denominator of $r_{2,2}$. If we apply HRFA to the above $r_{2,2}$, then approximate common factors are reduced.

Case $m = 3, n = 3$ Similar computations with the above elements in two rows of $\hat{A}$ and $\hat{B}$ become very small and depend on the environments of computers. However, the results are always meaningful and similar to the case of $m = n = 2$. A set of coefficients obtained with one computation is

$$a_0 = -0.33333, a_1 = -0.72881, a_2 = 0.17352, a_3 = 0.29 \times 10^{-7},$$
$$b_1 = 1.8530, b_2 = -1.2493, b_3 = 0.17352.$$

The resulting rational interpolant is obtained as

$$r_{3,3} \simeq \frac{0.17352(x^2 - 4.2x - 1.9207)}{0.17352(x - 3)(x^2 - 4.2x - 1.9209)} \simeq \frac{1}{x - 3}.$$

Approximate common factors which are the second order polynomial appear in the numerator and denominator of $r_{3,3}$. They may cause an *undesired zero and pole* of the rational interpolant but this is reduced by HRFA.

4.4. Facts Obtained from Practical Examples

If we wish to approximate a given function or a set of data accurately, we should increase the number of evaluation points in an interval. If the function can be written as a lower degree rational function, then there are many redundant points which could occur. They may cause *bad row*s in the triangular matrix. If undetermined symbols are substituted, then we can obtain all coefficients of the rational interpolant as shown in (4.4). However, in the approximate floating-point computation, the above computations for $f(x) = 1/(x - 3)$ show some interesting results

in terms of the appearance of pathological features of the rational interpolant. They are summarized as follows.

Exact computation

- In the case $\gamma = 0$, p_0/q_0 represents $f(x)$ exactly.
- In the case $\gamma = 1$, there is a bad row in the triangular system. An undetermined symbol t_1 should be introduced. The rational interpolant shows
$$\frac{p_0}{q_0} = \frac{p_1}{q_1} = f(x). \tag{4.6}$$
- In the case $\gamma = 2$, two undetermined symbols t_1 and t_2 should be introduced. The rational interpolant shows
$$\frac{p_0}{q_0} = \frac{p_1}{q_1} = \frac{p_2}{q_2} = f(x). \tag{4.7}$$

Approximate floating-point computation

- Instead of the introduction of undetermined symbols, negligible small values appear in the triangular system. Coefficients of the rational interpolant are obtained by straightforward numerical computations. Results of the interpolant contain approximate common factors in its numerator and denominator.
- The *undesired zero and pole* are caused by the above approximate common factors and are removed by the HRFA algorithm.

The above facts, especially (4.6) and (4.7), suggest an interesting relation

$$\frac{p_0}{q_0} = \frac{p_1}{q_1} = \frac{p_2}{q_2} = \cdots = \frac{p_\gamma}{q_\gamma} = f(x)$$

for the case of exact computation. It also agrees with Litvinov's result (4.5) for floating-point computation.

Next we consider the case of applications of HRFA to functions or a set of data which is not a rational function but a general continuous function such as a logarithmic function. As already discussed in this section, when we approximate $f(x) = \log(x + 2)$ by a rational function, we know the appearance of an approximate common factor in the numerator and denominator polynomials in the rational interpolant. By the approximate common factor, the pathological feature, *undesired zero and pole*, of the rational function appears. As future theoretical work, it is important to discuss much more the *undesired zero and pole* of the HRFA algorithm. In any event, in almost all cases, HRFA gives accurate rational approximation for a given function or a set of data.

5. Practical Applications of Hybrid Algorithms

Practical applications and developments of the HRFA and AppGCD hybrid algorithms are considered here. They are (1) smoothing data containing small and/or

large errors and (2) obtaining the Cauchy principal value integral. In the former, we can show the *robustness* of our method by comparisons with traditional methods such as the least square method. In the latter, we use the hybrid integral algorithm in which the AppGCD is effectively used [18].

5.1. Data Smoothing by HRFA

We show how HRFA works well for data including small errors and also large errors such as input errors. A set of data which includes some types of error is approximated by a naïve rational interpolation. Here, a set of data with large errors is considered as an *unattainable* point and causes an approximate GCD of the numerator and denominator polynomials of the rational function. Thus the error is removed from the reduced rational function. An early work of our method is given in [8].

5.1.1. Method for Data Smoothing. Two cases, (1) small error (noise) and (2) large error (experimental error), are considered. Let the input be a set of discrete data

$$D = \{(x_i, f_i) \mid i = 1, \ldots, m+n+1\}.$$

Small error case: Almost all the (x_i, f_i)'s are approximated by a rational function but with a small error. If we compute the approximate GCD of numerator and denominator polynomials with relatively large ε, we obtain a lower degree rational function by the approximate GCD. The resulting rational function gives a smooth function which approximates the given data D.
Large error case: We show how HRFA is effective for smoothing a set of data including a large error. The rational interpolation is shown as $r_{m,n}(x) = p_m(x)/q_n(x)$ for D. The rational function is written as

$$q_n(x_i)f_i = p_m(x_i), \quad i = 1, \ldots, m+n+1.$$

Thus, if $q_n(x_i) \neq 0$ for $i = 1, \ldots, m+n+1$, then $p_m(x)$ is uniquely determined. On the other hand, if $q_n(x_k) = 0$ is satisfied for a $k \in [1, m+n+1]$, then $p_m(x_k)$ vanishes also. Further, in this case, the following expression may be satisfied:

$$\lim_{x \to x_k} \frac{p_m(x)}{q_n(x)} \neq f_k.$$

The rational function $r_{m,n}$ is not through a point (x_k, f_k) in D. We call this point an *unattainable* point. Further, it is easy to say that if (x_k, f_k) is the *unattainable* point, then $p_m(x)$ and $q_n(x)$ have a common factor at $x = x_k$. With detailed considerations, we write the above facts as the following theorem.

Theorem. *Let s and t be positive integers satisfying $s+t < m+n$. Further let $r_{s,t} = p_s(x)/q_t(x)$ where p_s and q_t are polynomials with degrees s and t, respectively. A data set D is divided into two groups*

$$D_1 : \quad \{(x_{j_i}, f_{j_i}) \mid i = 1, \ldots, max(m+t, n+s)+1\},$$
$$D_2 : \quad \{(x_{k_i}, f_{k_i}) \mid i = 1, \ldots, \min(m-s, n-t)\},$$

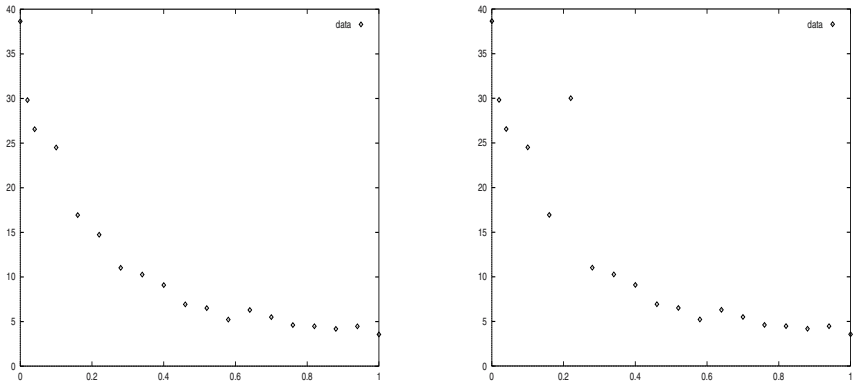
where $\{x_1, \ldots, x_{m+n+1}\} = \{x_{j_1}, x_{j_2}, \ldots, x_{k_1}, x_{k_2}, \ldots\}$. If the relations

$$f_{j_i} = r_{s,t}(x_{j_i}), \quad i = 1, \ldots, max(m+t, n+s)+1,$$
$$f_{k_i} \neq r_{s,t}(x_{k_i}), \quad i = 1, \ldots, \min(m-s, n-t),$$

hold, then the points in D_2 are unattainable points.

The theorem may be easily proved by *contradiction*. Suppose that $r_{m,n}(x) = p_m(x)/q_n(x)$ is a rational function that interpolates the given set of discrete data and is not the same as the rational function $r_{s,t} = p_s(x)/q_t(x), s < m, t < n$ described in the theorem above. If a relation $r_{m,n}(x) = r_{s,t}(x)$ holds, then the numbers of solutions of the relation are at most $\max(m+n, n+s)$. However, there are $\max(m+t, n+s)+1$ points which should be on $r_{s,t}$. Thus, the assumption that $r_{m,n}(x) \neq r_{s,t}(x)$ leads to a contradiction. This shows that $r_{m,n}(x) = r_{s,t}(x)$.

5.1.2. Examples of Data Smoothing. Next, we show with some examples how our method works well for a given set of data with small and large errors. We consider here two given types of input experimental data with small error as shown in Fig. 2(a) and large error as shown in Fig. 2(b).



(a) Including small error (b) Including large error

FIGURE 2. Given an experimental set of 19 discrete data

Small error case: Since the given set of discrete data consists of 19 points $(m+n+1 = 1)$, it is interpolated by a rational function $r_{9,9}(x) = p_9(x)/q_9(x)$.

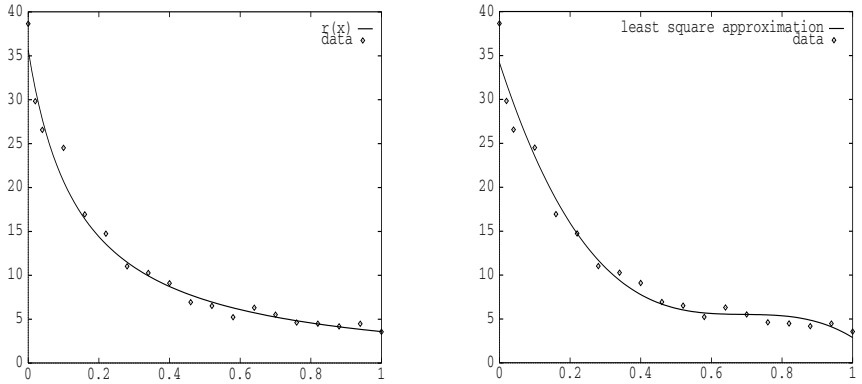
Following the above method, the AppGCD of both polynomials is computed as

$$\begin{aligned} g_8(x) &= \mathrm{AppGCD}(p_9(x), q_9(x), 0.1) \\ &= -1.991 \times 10^{-5} + 6.532 \times 10^{-3}x - 0.1549x^2 + 1.400x^3 - 6.367x^4 \\ &\quad +15.99x^5 - 22.43x^6 + 16.41x^7 - 4.862x^8. \end{aligned}$$

The result of our method is obtained in dividing $r_{9,9}(x)$ by the AppGCD as

$$\tilde{r}_{1,1}(x) = \frac{35.66 - 7.560x}{1.000 + 6.871x}.$$

The result $\tilde{r}_{1,1}(x)$ is shown in Fig. 3(a) and compared with the result of the least square method for the same set of discrete data shown in Fig. 3(b).



(a) $\tilde{r}_{1,1}(x)$ (b) Least square approximation

FIGURE 3. Results of our method and the least square method

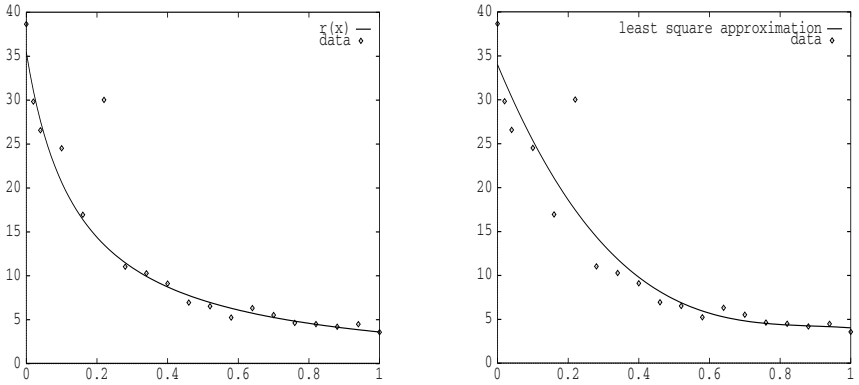
Large error case: The main parts of the given set of discrete data are the same as in the small error case. The data smoothing method is the same as above as well. We also obtain the AppGCD whose degree is 8 from a rational function $r_{9,9} = p_9(x)/q_9(x)$ as

$$\begin{aligned} g_8(x) &= \mathrm{AppGCD}(p_9(x), q_9(x), 0.1) \\ &= -1.558 \times 10^{-5} + 6.067 \times 10^{-3}x - 0.1460x^2 + 1.339x^3 - 6.167x^4 \\ &\quad +15.65x^5 - 22.13x^6 + 16.29x^7 - 4.850x^8. \end{aligned}$$

The result of our method is obtained in dividing $r_{9,9}(x)$ by the AppGCD as

$$\tilde{r}_{1,1}(x) = \frac{35.43 - 7.532x}{1.000 + 6.815x}.$$

The result $\tilde{r}_{1,1}(x)$ is shown in Fig. 4(a) and compared with the result of the least square method for the same set of discrete data shown in Fig. 4(b).



(a) $\tilde{r}_{1,1}(x)$ (b) Least square approximation

FIGURE 4. The results of our method and the least square method

The above results of data smoothing for experimental data with small/large error show the possibility of our method based on HRFA as one of the powerful methods for data smoothing.

5.2. Cauchy Principal Value Integral by Hybrid Methods

We apply HRFA to the Cauchy Principal Value integral (CPV) and a kind of the Cauchy-type singular integral equation. Through examples, we show how computations by HRFA give more accurate and stable results than usual numerical computation. Detailed discussions are in [9, 10].

The CPV is defined as

$$\wp \int_a^b \frac{f(x)}{x-\lambda} dx = \lim_{\varepsilon \to 0+} \left(\int_a^{\lambda-\varepsilon} \frac{f(x)}{x-\lambda} dx + \int_{\lambda+\varepsilon}^b \frac{f(x)}{x-\lambda} dx \right).$$

The difficulty on the numerical evaluation of such integral is that the integrand $\frac{f(x)}{x-\lambda}$ is unbounded in the neighborhood of $x = \lambda$. The Hilbert transform of $f(x)$ is a well-known method of subtracting the singularity as

$$\wp \int_a^b \frac{f(x)}{x-\lambda} dx = \int_a^b \frac{f(x)-f(\lambda)}{x-\lambda} dx + \wp \int_a^b \frac{f(\lambda)}{x-\lambda} dx. \tag{5.1}$$

A lot of methods have been proposed for numerical evaluation of the CPV, when the singular point λ is known before the evaluation. Most of the methods are based on function approximations such as polynomial approximation and spline approximation. However, it becomes difficult to evaluate the CPV by traditional numerical methods for the case when the integrand is complex or given by experimental data. We propose an efficient approach to evaluate the CPV by a straightforward application of HRFA as follows:

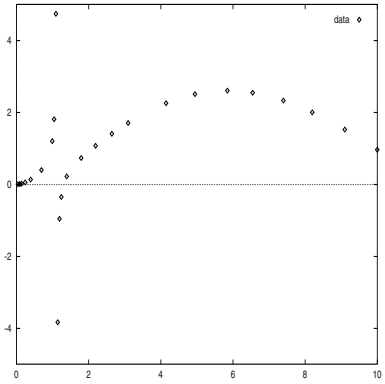


FIGURE 5. Given data from the analysis of optical waveguides

1. Obtain a set of data $D = (x_i, f(x_i)/(x_i - \lambda))$, $i = 0, \ldots, m+n$, where x_i and $f(x_i)/(x_i - \lambda)$ show the control variable and the associated value of the integrand, respectively.
2. Approximate given points and values to a rational function $r(x)$ with HRFA.
3. Obtain approximate indefinite integral of $r(x)$ by the hybrid integral algorithm.

We consider an example of using this approach. The experimental data from a problem on the analysis of optical wavewguides are shown in Fig. 5. In this case, the location of the pole is not known a priori. Thus numerical methods are difficult to apply. Naïve rational interpolation $r_{12,12}(x)$ which has *undesired zero and pole* in the interval $[0, 10]$ is first obtained. Then the HRFA algorithm reduces it to $\hat{r}_{9,9}(x)$ whithout these *zero and pole* for the accuracy $\varepsilon = 10^{-4}$ [8]. We can obtain the approximate indefinite integral, I, of $\hat{r}_{9,9}(x)$ in $[0, 10]$ by the hybrid integral algorithm. The value of the CPV is obtained by estimations of I at $x = 10$ and $x = 0$ as

$$I(x = 10) - I(x = 0) \simeq 16.8932.$$

The result is consistent with the experimental results concerning optical waveguides. Many physical/engineering problems are described by equations involving the CPV and given by discrete experimental data. Traditional numerical methods and methods using Hilbert transform cannot be applied to thse problems. Our method, the hybrid integral algorithm using HRFA, can give approximate but stable results for such problems.

The method is naturally extended to solve a Cauchy-type singular integral equation. The equation is written as

$$ag(x) + \frac{b}{\pi} \int_{-1}^{1} \frac{g(t)dt}{t - x} + \lambda \int_{-1}^{1} k(x, t)g(t)dt = f(x), \quad -1 < x < 1. \tag{5.2}$$

The so-called dominant equation, a special case of (5.2),

$$ag(x) + \frac{b}{\pi}\int_{-1}^{1} \frac{g(t)dt}{t-x} = f(x), \quad -1 < x < 1,$$

is considered in [10]. The result is compared with that obtained by other numerical methods, such as the Lobatto-Chebyshev scheme. The resulting value of our method agrees but requires smaller interpolation points than other methods.

6. Conclusion and Future Work

In this paper, hybrid symbolic-numeric computation and its applications are described. We show how hybrid computation which first began with the introduction of the notion of approximate GCD (AppGCD) works well for obtaining accurate results for many kinds of ill-conditioned problems. It can be applied to most problems which appear in scientific/engineering computations. The variables treated in them are inexact and are represented by floating-point numbers. The hybrid rational function approximation (HRFA) algorithm in which AppGCD is effectively used has theoretical implications and many applications. Regarding its theoretical implications, a pathological feature, the appearance of *undesired zero and pola*, is considered through practical examples. The fact that this may be caused by the ill-conditioned property of a system of simultaneous linear equations which determines all coefficients of the rational interpolant is mentioned. On the other hand, as applications of HRFA, several problems such as smoothing a set of data which contains small and/or large errors, integrating ill-conditioned functions or a set of data (hybrid integral) and also applying the Cauchy principal value integral are discussed.

The discussions herein are summarized as follows.

- AppGCD forms the basis of today's developments of hybrid symbolic-numeric computations.
- A hybrid integral algorithm, which is an extension of Horowitz' algorithm to an integrand with limited accuracy or perturbed within small tolerance, i.e., in the inexact case, works well for obtaining definite/indefinite integral of ill-conditioned rational functions.
- HRFA gives accurate rational approximations for a given function or a set of discrete data. An ill-conditioned property of the system of simultaneous linear equations which determines the coefficients of the rational interpolant causes the pathological feature, *undesired zero and pole*. However, the feature is eliminated by the AppGCD of numerator and denominator polynomials.
- A set of discrete data including small and/or large error is well smoothed by HRFA. Here, errors containing input data correspond to *unattainable* points and are removed by HRFA. The results of data smoothing show the *robustness* of our method.

- HRFA is used for computing the Cauchy principal value integral (CPV). Different from many proposed numeric methods for CPV, it is not necessary to have a priori knowledge of the position of singularity.

The following problems are important for further developing research related to hybrid computations.

1. Apply hybrid computation to practical engineering problems in a wider area, such as control theory.
2. Make clear the relation between hybrid computation and ill-conditioned problems mathematically such as the appearance of the pathological feature of the HRFA algorithm.
3. Apply the hybrid integral algorithm to obtain a kind of "indefinite" integral of a rational function with a parameter or a double integral.

Finally, the author wishes to acknowledge his collaborator Prof. Tateaki Sasaki of the University of Tsukuba and many graduate students who worked hard to establish the basis of hybrid computation during their studies at Ehime University. Especially, among them, Mr. Masaaki Ochi, Mr. Ei-ichi Miyahiro and Ms. Yumi Takashima, formerly Murakami, developed studies and obtained important results of Sects. 2, 3 and 4 of this paper, respectively. Further, Dr. Hiroshi Kai who is also one of them has been involved with almost all parts of the research.

References

[1] W. Auzinger and H. J. Stetter: An elimination algorithm for the computation of all zeros of a system of multivariate polynomial equations, *Numerical Mathematics*, **86**, ISNM, edited by R. P. Agarwal *et al.*, Birkhäuser, pp. 11–30, 1988.

[2] E. H. Bareiss: The numerical solution of polynomial equations and the resultant procedure, *Mathematical Method for Digital Computers*, vol. 2, John Wiley, pp. 185–214, 1967.

[3] R. M. Corless, P. M. Gianni, B. M. Trager and S. M. Watt: The singular value decomposition for polynomial systems, *Proc. ISSAC '95*, ACM Press, pp. 195–207, 1995.

[4] J. Grabmeier, E. Kaltofen and V. Weispfenning: *Computer Algebra Handbook*, Springer, pp. 112–125, 2003.

[5] I. Z. Emiris, A. Galligo and H. Lombardi: Certified approximate univariate GCDs, *J. Pure Appl. Algebra*, **117**, pp. 229–251, 1997.

[6] E. Horowitz: Algorithms for partial fraction decomposition and rational function integration, *Proc. 2nd ACM Symp. Symbolic and Algebraic Manipulation*, pp. 441–457, 1971.

[7] V. Hribernig and H. J. Stetter: Detection and validation of clusters of polynomial zeros, *J. Symb. Comp.*, **24**, pp. 667–681, 1997.

[8] H. Kai and M.-T. Noda: Hybrid rational function approximation and data smmoothing, *J. Jap. Appl. Math.*, **3**, pp. 323–336, 1993 (in Japanese).

[9] H. Kai and M.-T. Noda: Cauchy principal value integral using hybrid integral, *SIGSAM BULLETIN*, **31**, pp. 37–38, 1997.

[10] H. Kai and M.-T. Noda: Hybrid computation of Cauchy-type singular integral, *SIGSAM BULLETIN*, **32**, pp. 59–60, 1998.

[11] H. Kai and M.-T. Noda: Hybrid rational function approximation and its accuracy analysis, *Reliable Computing*, **6**, pp. 429–438, 2000.

[12] E. Kaltofen: Fast parallel absolute irreducibility testing, *J. Symb. Comp.*, **1**, pp. 57–67, 1985.

[13] E. Kaltofen: http://www.math.ncsu.edu/~kaltofen/bibliography/kaltofen.html#GKW02, pp. 112–129, 2005.

[14] N. Karmarkar and Y. N. Lakshman: Approximate polynomial greatest common divisors and nearest singular polynomials, *Proc. ISSAC '96*, ACM Press, pp. 35–39, 1996.

[15] G. Litvinov: Approximate construction of rational approximations and the effect of error autocorrection: Applications, *Russian Journal of Mathematical Physics*, vol. 1, no. 3, pp. 1–45, 1994.

[16] Y. Nakajima, H. Kai and M.-T. Noda: Hybrid rational function approximation and ill-conditioned problem, *J. JSSAC*, **11**, pp. 141–152, 2005.

[17] M.-T. Noda and E. Miyahiro: On the symbolic/numeric hybrid integration, *Proc. ISSAC '90*, ACM Press, p. 304, 1990.

[18] M.-T. Noda, E. Miyahiro and H. Kai: Hybrid rational function approximation and its use in the hybrid integration, *Advances in Computer Methods for Partial Differential Equations VII*, edited by R. Vichnevetsky, D. Knight and G. Richter, IMACS, pp. 565–571, 1992.

[19] M.-T. Noda and E. Miyahiro: Extension of the hybrid integral by using the hybrid rational function approximation, *J. Jap. Appl. Math.*, **2**, pp. 193–206, 1992 (in Japanese).

[20] M.-T. Noda and T. Sasaki: The interval arithmetic for the ill-conditioned polynomial equation, *RIMS* (Research Institute of Mathematical Sciences, Kyoto University) *Lecture Note*, **673**, pp. 47–61, 1988.

[21] M.-T. Noda and T. Sasaki: Approximate GCD and its application to ill-conditioned algebraic equations, JCAM, **38**, pp. 335–351, 1991.

[22] M. Ochi, M.-T. Noda and T. Sasaki: Approximate GCD of multivariate polynomials and application to ill-conditioned system of algebraic equations, *J. Inf. Proces.*, **14**, pp. 292–300, 1991.

[23] V. Y. Pan: Computation of approximate polynomial GCDs and an extension, *Proc. 9th Annual ACM-SIAM Symp. on Discrete Algorithms*, pp. 68–77, 1998.

[24] J. R. Rice: *The Approximation of Functions* II, Addison-Wesley, pp. 76–122, 1969.

[25] T. J. Rivlin: *An Introduction to Approximation of Functions*, Blaisdell, pp. 120–141, 1969.

[26] T. Sasaki and M.-T. Noda: Approximate square-free decomposition and root-finding of ill-conditioned algebraic equations, *J. Inf. Proces.*, **12**, pp. 159–168, 1989.

[27] T. Sasaki and M. Sasaki: A unified method for multivariate polynomial factorization, *J. Inf. Proces.*, **10**, pp. 21–39, 1993.

[28] A. Schönhage: Quasi-GCD computations, *J. Complexity*, **1**, pp. 118–137, 1985.

[29] T. W. Sederberg and G. Z. Chang: Best linear common divisors for approximate degree reduction, *Computer-Aided Design*, **25**, pp. 163–168, 1993.

[30] K. Shirayanagi and M. Sweedler: A theory of stabilizing algebraic algorithms, Tech. Rep. 95-28, Cornell Univ., pp. 1–92, 1995.

Matu-Tarow Noda
Center for Information Technology
Ehime University
Ehime Campus Information Service
Co. Ltd., Bunkyo-cho 3
Matsuyama, 790-8577, Japan
e-mail: noda@ecis.co.jp