ALGORITHMS FOR PARTIAL FRACTION DECOMPOSITION AND RATIONAL FUNCTION INTEGRATION

BY

Ellis Horowitz

Abstract

Algorithms for symbolic partial fraction decomposition and indefinite integration of rational functions are described. Two types of partial fraction decomposition are investigated, square-free and complete square-free. A method is derived, based on the solution of a linear system, which produces the square-free decomposition of any rational function, say A/B. The computing time is shown to be $O(n^4(\ln nf)^2)$ where deg(A)<deg(B) =n and f is a number which is closely related to the size of the coefficients which occur in A and B. The complete square-free partial fraction decomposition can then be directly obtained and it is shown that the computing time for this process is also bounded by $O(n^4(\ln nf)^2)$.

A thorough analysis is then made of the classical method for rational function integration, due to Hermite. It is shown that the most efficient implementation of this method has a computing time of $O(k^3n^5(\ln nc)^2)$, where c is a number closely related to f and k is the number of square-free factors of B. A new method is then presented which avoids entirely the use of partial fraction decomposition and instead relies on the solution of an easily obtainable linear system. Theoretical analysis shows that the computing time for this method is $O(n^5(\ln nf)^2)$ and extensive testing substantiates its superiority over Hermite's method.

INTRODUCTION

The idea of using computers to do symbolic mathematics has been with us now for almost two decades. Many computer systems have been developed, some of which concentrate on a particular task (e.g. symbolic integration) while others provide a wide variety of operations on large classes of mathematical expressions. A rather complete list of these systems can be found in [9] and [10].

The classical method for symbolic rational function integration is due to the 19th century mathematician Charles Hermite, [5]. Several systems, such as Engelman's MATHLAB, [3], and Moses' SIN, [8], have relied on Hermite's method to perform partial fraction decomposition and rational function integration. Tobey, in [11] undertook a detailed study of possible implementations resulting from Hermite's method, expressed these strategies in algorithmic form and then proved their correctness.

More recent people such as Collins [1] and Knuth [7] have developed analysis techniques which can be applied to algorithms which perform complex, symbolic mathematical operations. Section 2 will lay the theoretical framework for the development of partial fraction decomposition and symbolic integration algorithms. Also, the basic theorems for performing a computing time analysis of these algorithms will be presented. In Section 3, several algorithms for partial fraction decomposition, including a new method, will be derived and their computing times will be analyzed. In Section 4, algorithms which implement Hermite's method for rational function integration will be developed and analyzed. These algorithms will then be compared to a new method. Both a theoretical and an empirical analysis will be done. In Section 5, several extensions of this new method for integration will be discussed.

DEFINITIONS AND THEORY

In order to analyze the efficiency of algorithms it is necessary to develop the computing time for all subparts of the algorithm. Moreover, we require that these computing times shall be independent of the particular computer on which these algorithms may be implemented. Let us assume that integers are represented in radix form with arbitrary base β. Then computing times for the arithmetic operations can be expressed as functions of the number of β-digits of the numbers which occur in the algorithms. Hoever, since the number of β-digits of N is $[\log_\beta N]+1$ and since $\log_\beta N = (\ln N)/(\ln \beta)$ where "ln" is the natural logarithm function and since we will usually ignore constant multipliers, the computing times are given in terms of ln N. Also, these constant multipliers are dependent upon the particular computer of implementation, the data representation and numerous other details. It is, therefore, useful to ignore them uniformly and provide an analysis which is independent of any particular computer. I will now state several theorems which give the computing times for operations on integers and univariate polynomials. For proofs of these theorems, see [1]. These results will be used in analyzing the algorithms which occur in the latter sections.

Definition: $f(x)=O(g(x))$ means that there exists a constant c such that $f(x)\leq c.g(x)$ for all sufficiently large x.

The following theorems present computing times for algorithms which can be found in [7].

Theorem 2.1. Let t(a,b) be the time required to compute a+b (or a-b). Let $T(d) = \max\{t(a,b): |a|, |b| \leq d\}$. Then $T(d) = O(\ln d)$.

Theorem 2.2 Let t(a,b) be the time required to compute a·b. Let $T(d,e)=\max\{t(a,b): |a|\leq d, |b|\leq e\}$. Then $T(d,e)=O((\ln d)(\ln e))$.

Theorem 2.2 refers to a classical multiplication algorithm. Although recently developed algorithms for multiplication of large integers are much faster, see [7], we will assume in this paper that the above computing time applies.

Theorem 2.3 Let t(a,b) be the time required to compute q and r given a and b, such that $a=bq+r, 0\leq|r|<|b|, ar\geq 0, abq\geq 0$. Let $T(d,e)=\max\{t(a,b): |b|\leq d, |q|\leq e\}$. Then $T(d,e)=O((\ln d)(\ln e))$.

Definition: Let $A(x)=\sum_{i=0}^{m} a_i x^i$ be a univariate polynomial with integer coefficients. Then, deg(A) = m and norm $(A)=\sum_{i=0}^{m}|a_i|$.

Norm(A) is a norm for the ring of polynomials over the integers and hence satisfies the following rules: norm(A+B)$\leq$norm(A)+norm(B) norm(A$\cdot$B)$\leq$norm(A)$\cdot$norm(B).

Definition: $U(d,m)=\{A(x)$: norm(A)$\leq d$, deg(A)$\leq m\}$.

Theorem 2.4. The time to compute $A(x)\pm B(x)$ for $A,B\in U(d,m)$ is $O(m(\ln d))$.

Theorem 2.5. The time to compute $A(x)\cdot B(x)$ for $A\in U(d,m)$, $B\in U(e,n)$ is $O(m\,n(\ln d)(\ln e))$.

Theorem 2.6. The time to compute $A(x)/B(x)$ for $B\in U(d,m)$, $A/B\in U(e,n)$ is $O(m\,n(\ln d)(\ln e))$.

The foundation for the analysis of more complex algorithms dealing with polynomial manipulation has now been presented. Before beginning a discussion of the algorithms for partial fraction decomposition and rational function integration, it is necessary to define certain notions.

A rational function R(x) will be regarded as a numerator - denominator pair of polynomials A(x)/B(x) where A(x) and B(x) have integer coefficients, are relatively prime and the leading coefficient of B(x) is positive.

Definition: A rational function $R(x)=A(x)/B(x)$ is called regular if $\deg(A)<\deg(B)$.

We note that every rational function can be uniquely expressed as a polynomial plus a regular rational function. Since symbolic integration of polynomials is a comparatively easy process, we will concern ourselves primarily with regular rational functions.

Let I denote the integral domain of the integers and Q(I) its quotient field, the rational numbers.

Definition: Let B(x) be a polynomial of positive degree. Then B(x) is said to be square-free if it cannot be written in the form $B(x)=C(x)D^2(x)$ where D(x) is a polynomial of positive degree.

It follows that a polynomial which is square-free has only roots of multiplicity one.

Definition: Let $B\in I[x]$ and suppose $B=b\Pi_{i=1}^{k}B_i^i$ where each $B_i\in I[x]$, B_i is primitive and has a positive leading coefficient for $1\leq i\leq k$. Also, $b\in I$, $\deg(B_k)>0$ and the B_i are pairwise relatively prime. Then $b\Pi_{i=1}^{k}B_i^i$ is called the square-free factorization of B.

We now define two forms of partial fraction decomposition which will be investigated in later sections. Both of these decompositions are necessary in connection with the classical method of rational function integration.

Definition: Let $A(x)/B(x)$ be a regular rational function and $B(x)=b\Pi_{i=1}^{k}B_i^i(x)$ the square-free factorization of $B(x)$. Suppose also that there exist polynomials A_i with coefficients in Q(I), $1\leq i\leq k$ such that $A(x)/B(x)=\sum_{i=1}^{k}A_i(x)/B_i^i(x)$, $\deg(A_i)<\deg(B_i^i)$ or if $\deg(B_i)=0$, $A_i=0$. Then, this sum is called a square-free partial fraction decomposition of $A(x)/B(x)$.

Definition: Let $A(x)/B(x)$ be a regular rational function and $B(x)=b\Pi_{i=1}^{k}B_i^i(x)$ the square-free factorization of $B(x)$. Suppose also that there exist polynomials $A_{i,j}(x)$ $(1\leq j\leq i, 1\leq i\leq k)$ with coefficients in Q(I) such that $A(x)/B(x)=\sum_{i=1}^{k}\sum_{j=1}^{i}A_{i,j}(x)/B_i^j(x)$, $\deg(A_{i,j})<\deg(B_i)$ or if $\deg(B_i)=0$, $A_{i,j}=0$ for $1\leq j\leq i$. Then this sum is called a complete, square-free partial fraction decomposition of $A(x)/B(x)$.

For an example of these decompositions see Figure 1.

Let

$A(x)/B(x)=1/(x^2+1)(x-1)^2(x-2)^3(x-3)^3$. The Square-Free Partial Fraction Decompostion of

$$\frac{A(x)}{B(x)}=\frac{-(x+1)}{1000(x^2+1)}+\frac{7x-5}{32(x-1)^2}$$
$$+\frac{871x^5+11944x^4-65567x^3+180458x^2-24979x+139864}{4000(x-2)^3(x-3)^3}$$

The Complete Square-Free Partial Fraction Decomposition of

$$\frac{A(x)}{B(x)}=\frac{-(x+1)}{1000(x^2+1)}+\frac{7}{32(x-1)}+\frac{1}{16(x-1)^2}$$
$$+\frac{-871x+3234}{4000(x-2)(x-3)}+\frac{-1000x+3540}{4000(x-2)^2(x-3)^2}$$
$$+\frac{-700x+220}{4000(x-2)^3(x-3)^3}$$

Figure 1

In [6] the uniqueness of both of these decompositions is established. The method of Hermite depends first upon obtaining these two decompositions, successively. In Section 4 it will be shown how the integral is obtained from the terms of the complete square-free partial fraction decomposition.

Let us now look at some theoretical results about the form of the integral of a rational function. The proofs of these theorems have appeared several times and so I have omitted some of them here for the purposes of brevity.

Theorem 2.7. ([4], pp. 12) Let $R(x)=A(x)/B(x)$ be a regular rational function. Then $\int R(x)dx=S(x)+\sum_{i=1}^{m}d_i\log(x-b_i)$ where S(x) is a regular rational function, b_i are in the comlpex number field, $\mathbb{C}$, and the b_i are the

distinct roots of $B(x), d_i \in \mathbb{C}$ for $1 \le i \le m$.

Proof: We can write $B(x)$ as $B(x)=b_0(x-b_1)^{n_1}\cdots(x-b_m)^{n_m}$ where $b_i \in \mathbb{C}, b_i$ are the distinct roots of $B(x)$ and the n_i are their multiplicities. Then, expanding $R(x)$ into a complete partial fraction decomposition where the denominators are the linear factors of $B(x)$ we get constants $\alpha_{11}, \alpha_{12}, \ldots \in \mathbb{C}$ such that

$$R(x)=\sum_{i=1}^{m}\left\{\alpha_{i,1}/(x-b_i)+\alpha_{i,2}/(x-b_i)^2+\ldots+\alpha_{i,n_i}/(x-b_i)^{n_i}\right\}.$$

It then follows that $\int R(x)dx=$

$$\sum_{i=1}^{m}\left\{\alpha_{i,1}\log(x-b_i)-\alpha_{i,2}/(x-b_i)-\cdots-\alpha_{i,n_i}/(n_i-1)(x-b_i)^{n_i-1}\right\}.$$

Hence, let

$$S(x)=\sum_{i=1}^{m}\left\{-\alpha_{i,2}/(x-b_i)-\cdots-\alpha_{i,n_i}/(n_i-1)(x-b_i)^{n_i-1}\right\}$$

and $d_i=\alpha_{i,1}$ for $1 \le i \le m$.

Theorem 2.8. ([4],pp.14) let $b_1,\ldots,b_m$ be distinct elements of $\mathbb{C}$ and $\alpha_1,\ldots,\alpha_m \in \mathbb{C}$.

If $\sum_{i=1}^{m}\alpha_i\log(x-b_i)$ is a rational function, then $\alpha_i=0$ for $1 \le i \le m$.

Theorem 2.9. Given a regular rational function $A(x)/B(x)$ where $B(x)$ is square-free, then $\int A(x)/B(x)dx=\sum_{i=1}^{m}\alpha_i\log(x-b_i)$ where the b_i are the distinct roots of $B(x), \alpha_i, b_i \in \mathbb{C}$.

Proof Since $B(x)$ is square-free, let $b_1,\ldots,b_m$ be the distinct roots of $B(x)$. We can write $B(x)$ as $B(x)=b_0(x-b_1)\ldots(x-b_m)$. Then there exists constants $\alpha_{i,1}$ for $1 \le i \le m$ such that

$$A(x)/B(x)=\sum_{i=1}^{m}\left\{\alpha_{i,1}/(x-b_i)\right\}.$$

It follows that $\int A(x)/B(x)dx=\sum_{i=1}^{m}\alpha_{i,1}\log(x-b_i)$. By the previous theorem, no part of this sum can be equal to a rational function. We can now make the following definitions.

Definition: Let $R(x)$ be a rational function such that

$\int R(x)dx=S(x)+\sum_{i=1}^{m}\alpha_i\log(x-b_i)$, where b_i are the distinct roots of the demoninator of $R(x)$. Then $S(x)$ is called the rational part and $\sum_{i=1}^{m}\alpha_i\log(x-b_i)$ is called the transcendental part of $\int R(x)dx$.

Theorem 2.10. It $R(x)$is a rational function, then the rational and transcendental parts of $\int R(x)dx$ are unique.

Proof Suppose $\int R(x)dx=S(x)+\sum_{i=1}^{m}\alpha_i\log(x-b_i)$

$=T(x)+\sum_{i=1}^{n}\beta_i\log(x-c_i)$ where $S(x)$, $T(x)$ are rational functions. Then

$$S(x)-T(x)=\sum_{i=1}^{n}\beta_i\log(x-c_i)-\sum_{i=1}^{m}\alpha_i\log(x-b_i).$$

Then by Theorem 2.8, we have the right-hand side of this equation equal to a rational function only if that rational function is equal to zero. Therefore, $S(x)=T(x)$ and $\sum_{i=1}^{n}\beta_i\log(x-c_i)=\sum_{i=1}^{m}\alpha_i\log(x-b_i)$.

We now have defined two types of partial fraction decomposition which will be needed to implement Hermite's method. Also, the rational and transcendental parts of the integral of a rational function have been defined and shown to be unique.

Then, what does Hermite's method really do. It gives us a constructive means of obtaining exactly the rational part of the integral of a rational function. Moreover, this method requires only rational operations and not a priori knowledge of the roots of the denominator. The method has two main phases. The first consists of obtaining the complete square-free partial fraction decomposition of the integrand. The second phase applies a reduction scheme to these partial sums, producing two rational functions. One of these is the rational part of the integral while the integral of the other is the transcendental part of the original integral.

Section 3 will develop and analyze algorithms for finding partial fraction decompositions. Section 4 will combine the partial fraction decomposition algorithms of Section 3 with an algorithm that implements the second phase of Hermite's method. This resulting procedure will be compared to a new method for obtaining the same results as Hermite's method. Both theoretical and empirical computing time analyses of the two methods will be presented.

PARTIAL FRACTION DECOMPOSITION

There are two distinct types of decompositions we wish to obtain, square-free and complete square-free partial fraction decomposition. The latter should be recognized as a refinement of the former. The first algorithm which will be presented is one which computes the square-free factorization of a given polynomial. These

factors will constitute the denominators for both of these partial fraction decompositions. This algorithm will be followed by a theorem which bounds its computing time. Then, to obtain the numerators in the square-free partial fraction decomposition, I will first describe and analyze the approach suggested by Hermite [5] and implemented by several others, e.g.[3]. A new method for square-free partial fraction decomposition will then be derived and shown to be computationally more efficient than the previously used method. An algorithm based on this new method will be formally presented and analyzed.

In deriving a method for producing the complete square-free partial fraction decomposition, two sub-algorithms are needed. The first actually reduces the partial sums of the square-free decomposition while the second controls its application. Several theorems will be presented which establish the existence of a certain type of polynomial with integer coefficients which occurs in the reduction to the complete square-free decomposition. The existence of these polynomials allows for a more efficient algorithm.

Let us now begin with the algorithm which computes the square-free factorization of a primitive polynomial. A univariate polynomial with integer coefficients, A(x) is primitive if the greatest common divisor of its coefficients is one, [7]. Similarly, the content of A(x), abbreviated cont(A), is equal to the greatest common divisor of the coefficients of A. Given an arbitrary polynomial $A(x) \in I[x]$ such that $A \in U(f,n)$, then the time to compute its content and primitive part (abbreviated pp(A)) is $O(n(\ln f)^2)$. For a proof of this result see [1]. Throughout the remainder of this paper, let ldcf(A) stand for the leading coefficient of A(x).

Algorithm PSQFRE(B)

Input $B \in I[x], B \neq 0$, B primitive, deg(B)>0, ldcf(B)>0;

Output A list $L=(B_1,\ldots,B_k)$ where $B_i \in I[x]$ for $1 \leq i \leq k$ and $B=\prod_{i=1}^{k} B_i^i$ is the square-free factorization of B;

1) I←0;Q←0;D←0;P←1;A←B;

2) E←gcd(A,A'); If deg(E)≠0,do(F←A/E;go to (4))

3) F←A;

4) If I=0, go to (7);

5) If deg(D)≠deg(F),do(Adjoin D/F to Q; go to (7));

6) Adjoin P to Q;

7) If deg(E)≠0,do(I←1;A←E;D←F;go to (2));

8) Adjoin A to Q;

9) L←INVERSE(Q);

10) Return;

Note that if k>1, then $D=\prod_{i=1}^{k} B_i$ for j=1,..., k-1 at successive executions of step (7) and $E=\prod_{i=j}^{k} B_i^{i-j+1}$ for j=2,...,k at successive executions of step (2) in the algorithm. If the polynomial B is initially square-free, then the gcd(B,B') in step (2) has degree zero and hence Algorithm PSQFRE(B) performs only one greatest common divisor calculation and then terminates. If $B \in U(f,n)$, then the minimum computing time for PSQFRE(B) is $O(n^3(\ln nf)^2)$. In any case, step (2) is executed k times, where B is the square-free factorization

$$B=\prod_{i=1}^{k} B_i^i .$$

Theorem 3.1. Let $B \in I[x], B \neq 0$, n=deg(B) and $\prod_{i=1}^{k} B_i^i$ and square-free factorization of B. Let $f_i=\text{norm}(B_i)$ for $1 \leq i \leq k$ and $f=\prod_{i=1}^{k} f_i^i$. Then the computing time for PSQFRE(B) is bounded by $O(kn^3(\ln nf)^2)$.

Proof: The times for steps (1),(3),(4),(6), (7),(8), and (9) are bounded by O(k). The successive values of A at step (2) are $B_1B_2^2\ldots B_k^k, B_2B_3^2\ldots B_k^{k-1}, \ldots, B_{k-1}B_k^2, B_k$. Each of these polynomials belongs to U(f,n) and hence their derivatives belong to U(nf,n). The time for one execution of step (2) is $O(n^3(\ln g)^2)$ and hence the total time for step (2) in

$$O(\sum_{i=1}^{k} n^3(\ln nf)^2)=O(kn^3(\ln nf)^2).$$

In step (5), the successive values of D belong to U(f,n) and the values of D/F belong to $U(f,n_i)$, where $n_i=\deg(B_i)$. Hence, the total time for step (5) is

$$O(\sum_{i=1}^{k-1} n_i \cdot n(\ln f)^2)=O(n(\ln f)^2 \sum_{i=1}^{k-1} n_i \leq$$

$$O(n^2(\ln f)^2).$$

Now that we have investigated the determination of the square-free factorization of B, let us examine how we might determine the polynomials A_i such that

$$A/B = \sum_{i=1}^{k} A_i/B_i^i \text{ is the square-free}$$

partial fraction decomposition of the regular rational function A/B. Assume that the B_i, $1 \leq i \leq k$ have already been calculated. Hermite has suggested that we use the following procedure:

1) $F_0 \leftarrow A$; $C_0 \leftarrow B$;

2) For i=2,...,k do

2.1) $C_{i-1} \leftarrow C_{i-2} \;/\; B_{i-1}^{i-1}$;

2.2) Find F_{i-1}, A_{i-1} such that

$$F_{i-1}\, B_{i-1}^{i-1} + A_{i-1}\, C_{i-1} = F_{i-2},$$ where

$$\deg(F_{i-1}) < \deg(C_{i-1}),$$

$$\deg(A_{i-1}) < \deg(B_{i-1}^{i-1});$$

3) $A_k \leftarrow F_{k-1}$;

The difficulty is in step (2.2). We know that in general F_{i-1} and A_{i-1} will have rational number coefficients. One method of solution would be to equate coefficients. If $n = \deg(B_{i-1}^{i-1}) + \deg(C_{i-1})$ this approach would produce n linear equations in as many unknowns. It is known that the time needed to solve such a system exactly is $O(n^5(\ln nf)^2)$, where the elements of the matrix and the right-hand side are integers bounded by the postive integer f. This computing time applies if we use either the exact division method over the integers or Gaussian elimination over the rationals. Also, the numerators and denominators of the elements of the solution vector will be bounded by $(nf)^n$. This follows directly by applying Hadamard's theorem,

$$\det(E) \le \prod_{i=1}^{n} \Big\{\sum_{j=1}^{n} e_{i,j}^2\Big\}^{\frac{1}{2}}$$ to any matrix

$E = (e_{i,j})_{n\chi n}$ where $|e_{i,j}| \le f$.

We can apply these results to solving for F_{i-1} and A_{i-1} in (2.2). However before we do we realize that to obtain all the numerators, the A_i, we must perform step (2.2) for $2 \le i \le k$. It must be noted that one of the outputs of step (2.2) is F_{i-1}. Then, F_{i-1} is used as input to step (2.2) in the next iteration. If f_i = norm (B_i) and $f = \prod_{i=1}^{k} f_i^i$, then f is a bound for the coefficients of B_{i-1}^{i-1} and C_{i-1} for all iterations. However, if the numerators and denominators of the coefficients of F_o are bounded by f, then the coefficients of F_1 (which are elements of the solution vector of a linear system) will be bounded by $(nf)^n$ and the corresponding bound for F_2 will be $(n(nf)^n)^n$. If we continue the analysis we will find that the total computing time for solving step (2.2) for $2 \le i \le k$ using an algorithm for solution of linear systems is an exponential function of n and k.

Suppose instead that we consider an alternative way of implementing step (2.2), namely by using the extended Euclidean algorithm for polynomials, see [7] pp. 377. A new, fast algorithm has been developed which has a computing time of $O(n^3(\ln g)^2)$ where n bounds the degree of the inputs and g bounds their norms. Initially, this would seem to be better than using the linear systems approach. However, if we use the bound for the coefficients of F_{i-1} and A_{i-1} that was obtained in the analysis of the extended Euclidean algorithm, we would again arrive at a total computing time which is an exponential function of n and k.

This growth of the coefficient bound leads us to consider a non-iterative approach for determining the numerators of the square-free partial fraction decomposition.

Given the regular rational function A/B, let $\sum_{i=1}^{k} A_i/B_i^i$ be the square-free partial fraction decomposition of A/B, the coefficients of A_i being rational numbers. Let $n = \deg(B)$, $n_i = \deg(B_i)$ and $E = B/B_i^i$ for $1 \le i \le k$.

Then, if we consider the equation

$$A = \sum_{i=1}^{k} A_i E_i$$

where the coefficients of the A_i are undetermined, we can equate the corresponding coefficients on the two sides of this equation. This will produce an $n\chi n$ linear system of equations whose solution contains the coefficients of the numerators in the square-free decomposition of A/B. Let us examine more closely what this system will look like.

If $A_i(x) = \sum_{j=0}^{in_i-1} a_{i,j}x^j$, $E_i(x) = \sum_{j=0}^{n-in_i} e_{i,j}x^j$,

then $A_i(x)\, E_i(x) = \sum_{j=0}^{n-1} c_{i,j}x^j$ where

$$c_{i,k} = \sum_{j=k-n+in_i}^{in_i-1} a_{i,j}e_{i,k-j}$$ for

$0 \le k \le n-1$, where $j < 0$ or $k-j < 0$ implies $a_{ij}\, e_{i,k-j} = 0$. The coefficient of x^j in $\sum_{i=1}^{k} A_i\, E_i$ is given by $\sum_{i=1}^{k} c_{i,j}$. If we consider the matrix which is formed from these coefficients, we see that we have k distinct groups of columns, the i^{th} group consisting of the coefficients of E_i. Thus we are able to derive the following matrix:

$\left[\begin{array}{cc} A & B \end{array}\right]$ where

$$A = \begin{bmatrix} e_{1,n-n_1} & 0 & \cdots & 0 & \cdots \\ e_{1,n-n_1-1} & e_{1,n-n_1} & & \vdots & \\ \vdots & e_{1,n-n_1-1} & \ddots & \vdots & \\ \vdots & \vdots & \ddots & e_{1,n-n_1} & \\ e_{1,1} & & & e_{1,n-n_1-1} & \\ e_{1,0} & e_{1,1} & & \vdots & \\ 0 & e_{1,0} & \ddots & \vdots & \\ \vdots & 0 & \ddots & \vdots & \\ \vdots & \vdots & \ddots & e_{1,1} & \\ 0 & 0 & \cdots & e_{1,0} & \cdots \end{bmatrix} \quad (3.1)$$

n_1 columns

$$\left[\begin{array}{cccc} e_{k,n-kn_k} & 0 & \cdots & 0 \\ e_{k,n-kn_k-1} & e_{k,n-kn_k} & & \vdots \\ \vdots & e_{k,n-kn_k-1} & \ddots & 0 \\ \vdots & \vdots & \ddots & e_{k,n-kn_k} \\ e_{k,1} & & & e_{k,n-kn_k-1} \\ e_{k,0} & e_{k,1} & & \vdots \\ 0 & e_{k,0} & \ddots & \vdots \\ \vdots & \vdots & \ddots & e_{k,1} \\ 0 & 0 & \cdots & e_{k,0} \end{array}\right] = B \quad (3.1)$$

kn_k columns

If this matrix is called E and if $A(x) = \sum_{j=0}^{n-1} a_j x^j$, we obtain the linear system $Ey = F$ where $y=(a_{1,n_1-1},\ldots,a_{1,0},\ldots,a_{k,kn_k-1},\ldots,a_{k,0})$ and $F=(a_{n-1},a_{n-2},\ldots,a_0)$. (3.2)

Examining E columnwise, we see k distinct groups, the i^{th} group consisting of in_i adjacent columns, for $1 \le i \le k$. The i^{th} group consists of columns which contain the coefficients of E_i with possibly leading and trailing zeros. Hence, we see that to form the matrix E, we need only compute $E_i = B/B_i^i$ for $1 \le i \le k$. Also, if $f_i = \text{norm}(B_i)$ and $f = \Pi_{i=1}^{k} f_i^i$, then f is a bound for the elements of E. Morever, the elements of the solution vector, which are the coefficients of the A_i, have numerators and denominators which are bounded by $(nf)^n$ provided f is also a bound for the coefficients of A. Thus, this provides a good bound for all of the coefficients of the A_i. Furthermore, we now have a new method for obtaining the square-free partial fraction decomposition and the computing time will be the time needed to solve a single linear system, $O(n^5(\ln nf)^2)$.

Suppose instead that we reconsider using the extended Euclidean algorithm with the new bound we have just derived. Substituting $(nf)^n$ for g and applying the algorithm k - 1 times we have that the computing time is

$$O\left(\sum_{i=2}^{k} n^3(n \ln nf)^2\right) = O(kn^5(\ln nf)^2).$$

This is clearly not superior to $O(n^5(\ln nf)^2)$ which we obtain by solving a linear system. In [2], a more efficient algorithm for the exact solution of linear systems has been developed which is based on modular arithmetic. The computing time for this new algorithm is $O(n^4(\ln nf) + n^3(\ln nf)^2)$.

Therefore, I will now present an algorithm for square-free partial fraction decomposition which is based upon solving the matrix E. A subprogram, MUSSLE has been developed which takes as input an n by n non-singular matrix, E, and an n element vector F, both with integer entries. The n+1 vector of integers, $(g_o,g_1,\ldots,g_n)$ is returned such that $g_o = \det(E)$ and $EG=F$ where $G=(g_1/g_o,\ldots,g_n/g_o)$. The integers g_i are bounded by $(nf)^n$ for $o \le i \le n$, if f bounds the elements of E and F. The computing time for MUSSLE is $O(n^4(\ln nf) + n^3(\ln nf)^2)$. For the precise specifications of MUSSLE, see [2].

Algorithm RSQDEC(A/B)

Input: A non-zero, regular rational function A/B, where deg(B) > 0 and ldcf(B) > 0;

Output: A list L = (X,Y,Z) where X= $(A_1,\ldots,A_k)$, Y= $(B_1,\ldots,B_k)$ and Z = $(v_1,\ldots,v_k)$.

$A_i, B_i \in I[x]$, $v_i \in I$ and $A/B = \sum_{i=1}^{k} A_i/v_i B_i^i$ is the square-free partial fraction decomposition of A/B. If $B_i = 1$, then $A_i = 0$ and $v_i = 1$. Otherwise, $\deg(A_i) < \deg(B_i^i)$, $v_i > 0$ and $\gcd(v_i, \text{cont}(A_i)) = 1$;

1) b ← cont(B); $\overline{B}$ ← pp(B); X← Z ← 0;

2) Y ← PSQFRE ($\overline{B}$) obtaining Y = $(B_1,\ldots,B_k)$; k ← LENGTH(Y);

3) If k = 1, do (L = ((A), (B), (1)); return);

4) For i = 1,..., k-1 do (If $\deg(B_i) \neq 0$, go to (5));

4.1) Adjoin A to X, b to Z;

4.2) For i = 1,...,k-1 do (Adjoin 0 to X, 1 to Z);

4.3) L = (X,Y,Z); return;

5) Create the matrix E as defined in (3.1); Create the vector F as defined in (3.2);

6) Use MUSSLE(E,F) to solve for the vector $\overline{G}$: EG = F where E is nxn, $G = (g_1/g_o,\ldots,g_n/g_o)$ and $\overline{G} = (g_o,g_1,\ldots,g_n)$;

7) w ← $b.g_o$;

8) For i = 1,...,k do

8.1) m ← $\deg(B_i)$; If m = 0, do (Adjoin 0 to X, 1 to Z; go to (8.6));

8.2) n ← im; Take the next n elements of G and create polynomial A_i.

8.3) h ← gcd(w,cont(A_i));

8.4) v ← w/h ; A_i ← A_i/h ; If v < 0 do (v ← - v; A_i ← - A_i);

8.5) Adjoin A_i to X, v to Z;

8.6) Continue;

9) X ← INVERSE(X); Z ← INVERSE(Z); L ← (X,Y,Z):

10) Return;

Theorem 3.2 Let A/B be a non-zero, regular rational function, deg(B) > 0 and $b\prod_{i=1}^{k} B_i^i$ the square-free factorization of B. Let f_i = norm (B_i), $n_i = \deg(B_i)$ for $1 \leq i \leq k$, and let $f = \max\{\text{norm}(A), b\prod_{i=1}^{k} f_i^i\}$. Then, the computing time for RSQDEC(A/B) is $O(kn^3(\ln nf)^2 + n^4(\ln nf))$.

Proof: Step (1) takes $O(n(\ln nf)^2)$. By Theorem 3.1 step (2) takes $O(kn^3(\ln nf)^2)$. The time for steps (3),(4),(9), and (10) is bounded by O(n). The time to form the matrix and right-hand side in step (5) is $O(n^2(\ln nf)^2)$ and the elements of E, F are bounded by f. Hence, the time for step (6) is $O(n^4(\ln nf) + n^3(\ln nf)^2)$ and the elements of G are bounded by $(nf)^n$. Step(7) takes $O(n(\ln nf)^2)$. Steps (8.1), (8,2), and (8.5) are bounded by O(n) and the time for steps (8.3) and (8.4) is

$$O(\sum_{i=1}^{k} in_i n^2(\ln nf)^2) = O(n^3(\ln nf)^2).$$

Hence, the total time for RSQDEC(A/B) is $O(kn^3(\ln nf)^2 + n^4(\ln nf))$.

I have now presented an algorithm for obtaining the square-free partial fraction decomposition of a regular rational function, $A/B = \sum_{i=1}^{k} A_i/v_i B_i^i$, where the computing time is bounded by $O(n^4(\ln nf)^2)$. This algorithm is faster by at least one order of magnitude then the algorithms which resulted from iteratively applying either the extended Euclidean algorithm or the solution of a linear system to the determination of the A_i.

Before we can apply Hermite's reduction process, the partial fractions A_i/B_i^i for $i \geq 2$ must be decomposed further into a sum of partial fractions $A_i/B_i^i = \sum_{j=1}^{i} A_{i,j}/B_i^j$. In this case, either $A_{i,j} = 0$ or $\deg(A_{i,j}) < \deg(B_i)$ for $1 \leq j \leq i$.

We know that in general, the coefficients of $A_{i,j}$ are rational numbers. The following theorems will show that in fact a decomposition over I exists in the form $A_i/B_i^i =$

$$(1/b^{m-n+1}) \sum_{j=1}^{i} A_{i,j}/B_i^j \text{ where } b = \text{ldcf}(B_i)$$

$m = \deg(A_i)$, $n = \deg(B_i)$ and $A_{i,j} \in I[x]$ such that either $A_{i,j} = 0$ or $\deg(A_{i,j}) < \deg(B_i)$ for $1 \leq j \leq i$.

Theorem 3.3. Let A and B be non-zero polynomials over an integral domain I, with $m = \deg(A)$, $n = \deg(B)$, $m \geq n \geq 0$. Let Q and R be the unique polynomials over I such that $b_n^{m-n+1} A = BQ + R$, with $b_n = \text{ldcf}(B)$ and either $R = 0$ or $\deg(R) < n$. Then there exist $q_o, \ldots, q_{m-n} \in I$ such that $Q(x) = \sum_{i=0}^{m-n} q_i (b_n x)^i$.

Proof; Let $B(x) = \sum_{i=0}^{n} b_i x^i$, $Q(x) = \sum_{i=0}^{m-n} \bar{q}_i x^i$ and $A_i(x) = \sum_{j=0}^{m-i} a_j^{(i)} x^j$, for $0 \leq i \leq m-n+1$ and let $A_0, A_1, \ldots, A_{m-n-i+1}$ be defined by $A_0 = b_n^{m-n+1} A$, $A_i(x) = A_{i-1}(x) - (a_{m-i+1}^{(i-1)}/b_n) \cdot B(x) \cdot x^{m-n-i+1}$, for $1 \leq i \leq m - n+1$. Then $\bar{q}_{m-n-i} = a_{m-i}^{(i)}/b_n$ for $0 \leq i \leq m - n$. If we can show that A_i is of the form $A_i(x) = \sum_{j=0}^{m-i} b_n^{m-n-i+1} q_j^{(i)} x^j$, $q_j^{(i)} \in I$ $0 \leq i \leq m-n$, it follows that $\bar{q}_{m-n-i} = b^{m-n-i} \cdot q^{(i)}$ for $0 \leq i \leq m-n$ and hence $Q(x)$ is of the desired form. We proceed by induction.

Case (1): $A_0(x) = b_n^{m-n+1} A(x) = \sum_{i=0}^{m} b^{m-n+1} a_i x^i$, so $A_0(x)$ is in the desired form.

Case (i): Assume A_{i-1} has the form $A_{i-1}(x) = \sum_{j=0}^{m-i+1} b_n^{m-n+2-i} q_j^{(i-1)} x^j$.

Then $A_i(x) = A_{i-1}(x) - (b_n^{m-n-i+2} q_{m-i+1}^{(i-1)}/b_n) \sum_{j=0}^{n} b_j x^{j+m-n-i+1} =$

$$\sum_{j=0}^{m-i+1} b_n^{m-n+2-i} q_j^{(i-1)} x^j - \sum_{j=m-n-i+1}^{m-i+1} b_n^{m-n-i+1} q_{m-i+1}^{(i-1)} b_{j-m+n+i-1} x^j$$

$$= \sum_{j=0}^{m-1} b_n^{m-n-i+1} q_j^{(i)} x^j.$$

Theorem 3.4 Let A and B be non-zero polynomials over an integral domain I, with $m = \deg(A)$, $n = \deg(B)$ and $m \geq n > 0$. Let $B(x) = \sum_{i=0}^{n} b_i x^i$ and $A(x) = \sum_{i=0}^{m} a_i (b_n x)^i$ where $a_i, b_i \in I$. Then, there exist Q and R over I such that $A = BQ + R$, either $R = 0$ or $\deg(R) < n$, and Q is of the form $Q(x) = \sum_{i=0}^{m-n} q_i (b_n x)^i$ where $q_i \in I$.

Proof: Suppose $A_i(x) = \sum_{j=0}^{m-i} a_j^{(i)} x^j$ for $0 \leq i \leq m-n+1$, and let $A_0, A_1, \ldots, A_{m+n-1}$ be defined by $A_o = A$, $A_i(x) = A_{i-1}(x) - (a_{m-i+1}^{(i-1)}/b_n) \cdot B(x) \cdot x^{m-n-i+1}$ for $1 \leq i \leq m-n+1$. Then we define $Q(x) = \sum_{i=0}^{m-n} \bar{q}_i x^i$ where $\bar{q}_{m-n-i} = a_{m-i}^{(i)}/b_n$ for $0 \leq i \leq m-n$. If we can show that A_i is of the form $A_i(x) = \sum_{j=0}^{m-i} q_j^{(i)} (b_n x)^j$, $q_j^{(i)} \in I$, for $o \leq i \leq m-n$. Since $m - i \geq m - (m-n) - 1 = n - 1 \geq 0$, it follows that Q is over I and has the form $Q(x) = \sum_{i=0}^{m-n} q_i^* b_n^{i+n-1} x^i = \sum_{i=0}^{m-n} q_i (b_n x)^i$. We proceed by induction.

Case (1): $A_0(x) = A(x) = \sum_{i=0}^{m} a_i (b_n x)^i$ by hypothesis.

Case (i): Assume that A_{i-1} is of the form $A_{i-1}(x) = \sum_{j=0}^{m-i+1} q_j^{(i-1)} (b_n x)^j$.

Then $A_i(x) = A_{i-1}(x) - (a_{m-i+1}^{(i-1)}/b_n) \sum_{j=0}^{n} b_j x^{j+m-n-i+1} =$

$$A_{i-1}(x) - \sum_{j=m-n-i+1}^{m-i+1} q_{m-i+1}^{(i-1)} b_n^{m-i} b_{j-m+n+i-1} x^j =$$

$$\sum_{j=0}^{m-j} q_j^{(i)} (b_n x)^j.$$

Theorem 3.5. Let A and B be non-zero polynomials over an integral domain I, with $m = \deg(A)$, $n = \deg(B)$ and $m \geq n > 0$. Let $b = \text{ldcf}(B)$ and $k = [m/n] + 1$. Then there exist polynomials $A_1, \ldots, A_k$ over I such that $A/B^k = (1/b^{m-n+1}) \sum_{i=1}^{k} A_i/B^i$ and where either $A_i = 0$ or $\deg(A_i) < n$ for $1 \leq i \leq k$.

Proof: By Theorem 3.3. there exist polynomials Q_1, A_k over I such that $b_n^{m-n+1} A = BQ_1 + A_k$ and either $A_k = 0$ or $\deg(A_k) < n$. Also, Q_1 is of the form $Q_1(x) = \sum_{i=0}^{m-n} q_i(bx)^i$. Now assume that $Q_1,\ldots,Q_i$ and $A_k, A_{k-1},\ldots,A_{k-i+1}$ have been defined, that $\deg(Q_i) = m - in$ and that Q_i has the form $Q_i(x) = \sum_{j=0}^{m-in} q_j^{(i)}(bx)^j$. If $i + 2 \leq k$, then $i + 1 \geq [m/n]$. $m \geq (i+1)n$ and $m-in \geq n$. We then define Q_{i+1}, A_{k-i}, using Theorem 3.4 by $Q_i = BQ_{i+1} + A_{k-i}$ where either $A_{k-i}=0$ or $\deg(A_{k-i}) < n$. If $i+1 = k$, then $i = [m/n]$, $m < (i+1)n$ and $m-n < n$. We then define $A_{k-i} = A_1 = Q_i = Q_{k-1}$. Then $b^{m-n+1} A = BQ_1 + A_k$ and $Q_i = BQ_{i+1} + A_{k-i}$ for $1 \leq i \leq k-2$. It follows that $A/B^k = (1/b^{m-n+1}) \sum_{i=1}^{k}(A_i/B^i)$ and either $A_i = 0$ or $\deg(A_i) < n$.

Theorems 3.3, 3.4, and 3.5 establish the fact that we can decompose A/B^k in the form $A/B^k = (1/b^{m-n+1}) \sum_{j=1}^{k}(\bar{A}_j/B^j)$ where $k = [m/n] + 1$ and either $\bar{A}_j = 0$ or $\deg(\bar{A}_j) < \deg(B)$ for $1 \leq j \leq k$. In general, one needs a decomposition of the form $A/B^i = (1/b^{m-n+1}) \sum_{j=1}^{k}(\bar{A}_j/B^{j+i-k})$ so we can take $A_j = \bar{A}_{j+k-i}$ for $i - k + 1 \leq j \leq i$ and $A_j = 0$ for $1 \leq j \leq i - k$ and obtain the required decomposition.

We will apply this procedure to the partial fractions in the square-free partial fraction decomposition $A/B = \sum_{i=1}^{k} (A_k/v_i B_i^i)$. We will then obtain the complete square-free partial fraction decomposition in the form $A/B = \sum_{i=1}^{k}(1/w_i) \sum_{j=1}^{i}(A_{ij}/B_i^j)$, where either $A_{i,j}= 0$ or $\deg(A_{i,j}) < \deg(B_i)$ for $1 \leq j \leq i$, $1 \leq i \leq k$.

An algorithm is now given which performs this decomposition.

Algorithm PCDEC(A,B)

Input: $A,B \in I[x]$, $A,B \neq 0$, $m = \deg(A)$, $n = \deg(B)$, $m \geq n > 0$;

Output: The list (X,b) where $X = (A_k,\ldots,A_1)$ and b is an integer such that $k = [m/n] + 1$ and $A/B^k = (1/b)\cdot \sum_{j=1}^{k} A_j/B^j$. Also, $b = \mathrm{ldcf}(B)^{m-n+1}$ and either $A_j = 0$ or $\deg(A_j) < \deg(B)$ for $1 \leq j \leq k$;

1) $m \leftarrow \deg(A)$; $n \leftarrow \deg(B)$; $b \leftarrow \mathrm{ldcf}(B)^{m-n+1}$; $X \leftarrow 0$;

2) $Q \leftarrow b \cdot A$;

3) Obtain $\bar{Q}, \bar{A} \in I[x]$: $Q = B\bar{Q} + \bar{A}$ and either $\bar{A} = 0$ or $\deg(\bar{A}) < n$;

4) Adjoin $\bar{A}$ to X;

5) If $\deg(\bar{Q}) \leq n$, do ($Q \leftarrow \bar{Q}$; go to (3));

6) Adjoin $\bar{Q}$ to X; $X \leftarrow$ INVERSE(X);

7) Return;

Theorem 3.6. Let $A,B \in I[x]$, $m = \deg(A)$, $d = \mathrm{norm}(A)$, $n = \deg(B)$, $e = \mathrm{norm}(B)$, $m \geq n > 0$ and $i = [m/n] + 1$. Then, the computing time for PCDEC(A,B) is $O(i^2n^2(\ln e)\{\ln d + i^2n(\ln e)\})$.

Proof: The time for step (1) is $O(m^2(\ln e)^2)$. The time for step (2) is $o(m^2(\ln d)(\ln e))$ and norm $(Q) \leq de^{m-n+1}$. Steps (3) - (5) are executed i-1 times. At the j^{th} execution, $Q \in U(d(1 + e)^{j(m-n+1)}, m - (j-1)n)$. The time to obtain Q,A is $o(n(m-n+1)(\ln e)(\ln d + (m-n+1)(\ln e)))$, see [2]. Letting $m - (j-1)n$ replace m and $d(1+e)^{j(m-n+1)}$ replace d we have that the total time for step (3) is $O(\sum_{j=1}^{i-1} n(m-jn+1)(\ln e)\{\ln d(1+e)^{j(m-n+1)} + (m-jn+1)(\ln e)\})$. Now, $m = O(in)$ and $m-jn+1 \leq j(m-n+1)$, so the time for step (3) is $O(i\, n^2(\ln e) \sum_{j=1}^{i-1}\{\ln d + j(m-n+1)(\ln e)\})$ $= O(i\, n^2(\ln e)\{i(\ln d) + i^3 n\,(\ln e)\})$. Steps (4), (5) and (6) are bounded by $O(i)$.

If we view PCDEC as a subpart of the complete algorithm for partial fraction decomposition then empirical tests have shown that the time spent on PCDEC is a small fraction of the time for the entire algorithm. We can combine RSQDEC and PCDEC to produce an algorithm which obtains the complete, square-free partial fraction decomposition in the form A/B =

$\sum_{i=1}^{k} (1/w_i) \sum_{j=s_i}^{i} A_{i,j}/B_i^j$, where

if $B_i = 1$, $s_i = i$, $A_{i,i} = 0$ and $w_i = 1$. Otherwise $1 \le s_i \le i$, $w_i > 0$ and either $A_{i,j}=0$ or $\deg(A_{i,j}) < \deg(B_i)$ for $s_i < j < i$. The two main steps would be: 1) applying RSQDEC(A/B) to produce $A/B = \sum_{i=1}^{k} A_i/v_i B_i^i$ with a computing time of $O(n^4(\ln nf)^2)$; 2) applying PCDEC iteratively to A_i/B_i^i thus giving a computing time of

$$O\Big(\sum_{i=2}^{k} i^2 n_i^2 (\ln f)\{n(\ln nf) + i^2 n_i(\ln f)\}\Big) \le O(n(\ln nf)^2 \sum_{i=2}^{k} (i^2 n_i^2 + i^3 n_i^2))$$

$$= O(n(\ln nf)^2(n^2 + kn^2)) = O(kn^3(\ln nf)^2).$$

Therefore, the time for square-free decomposition bounds the computing time for complete square-free decomposition. For a precise statement of this final algorithm see [2].

Before leaving this section it is necessary to derive good bounds for the sizes of the coefficients in the complete square-free partial fraction decomposition. This is so because these coefficients are manipulated by Hermite to obtain exactly the rational part of the integral. Therefore let $A/B = \sum_{i=1}^{k} \sum_{j=1}^{i} A_{i,j}/B_i^j$ where $A_{i,j} \in Q[x]$, $A_{i,j} = 0$ or $\deg(A_{i,j}) < \deg(B_i)$ for $1 \le j \le i$ be the complete square-free partial fraction decomposition over the rationals of A/B. We wish to have bounds for the numerators and denominators of the coefficients of the $A_{i,j}$. For a given i, considers the rational function $(A_{i,i} + A_{i,i-1}B_i + \ldots + A_{i,1}B_i^{i-1})/B_i^i$. If $n_i = \deg(B_i)$, $f_i = \text{norm}(B_i)$ and if we consider the coefficients of $A_{i,j}$ as undetermined, we then have in_i indeterminates. Hence, if we consider

$$A/B = \left(\sum_{i=1}^{k} \left\{\sum_{j=1}^{i} A_{i,j}B_i^{i-j}\right\} \cdot B/B_i^i\right)/B$$

we can construct a linear system of order $n = \deg(B)$ by equating the corresponding coefficients in the numerator. If $f = \max\{\text{norm}(A), \prod_{i=1}^{k} f_i^i\}$, then the elements of this linear system will be bounded by f. The elements of the solution vector are simply the coefficients of the $A_{i,j}$ and so the numerators and denominators of these coefficients are bounded by $(nf)^n$.

In order to empirically compare the efficiency of these algorithms several classes of rational functions were used as test data. Consider the following class of rational functions: polynomials $B_i(x)$ with random integer coefficients $b_{i,j}$ were generated such that $\deg(B_i) = 1$ and $|b_{i,j}| \le 2^9$ for $j = 1,2$, $1 \le i \le n$; then $B_n(x) = \prod_{i=1}^{n} B_i^i(x)$ is formed. $A_n(x)$ is a random polynomial such that $\deg(A_n) = \deg(B_n) - 1$ and the coefficients of $A_n(x)$, $a_{n,j}$, satisfy $|a_{n,j}| \le 2^9$; then the rational functions $R_n(x) = A_n(x)/B_n(x)$ are formed for $n = 1,2,\ldots$

In Figure 2 below the rational functions $R_n(x)$ were used as input to two algorithms RDEC and QRDEC. RDEC obtains the complete square-free partial fraction decomposition of A/B in the form $\sum_{i=1}^{k} (1/w_i) \sum_{j=1}^{i} A_{i,j}/B_i^j$ and it uses the algorithms RSQDEC and PCDEC. The second algorithm, QRDEC obtains a decomposition in the form $A/B = \sum_{i=1}^{k} \sum_{j=1}^{i} \bar{A}_{i,j}/B_i^j$, where the coefficients of $\bar{A}_{i,j}$ are rational numbers. The method QRDEC uses is the one suggested by Hermite with all computations over the rationals.

Func.	RDEC	QRDEC
R_1	.009	.406
R_2	.218	.361
R_3	1.193	3.488
R_4	3.761	18.782
R_5	17.929	100.406

FIGURE 2. (seconds)

4. Rational Function Integration

Given the complete square-free partial fraction decompositon of $A/B = \sum_{i=1}^{k} (1/w_i) \sum_{j=1}^{k} A_{ij}/B_i^j$, we are now ready to apply Hermite's reduction process to the partial fractions $\sum_{j=1}^{i} A_{i,j}/B_i^j$ for

$2 \leq i \leq k$. Since B_i is square-free, we know that $\deg(\gcd(B_i, B_i')) = 0$ and hence that there exist polynomials C_i, $D_i \in Q[x]$ such that $C_iB_i + D_iB_i' = A_{i,i}$. For proofs of these assertions see [7], pp. 377. If $\deg(B_i) = 0$, then $C_i = 0$, $B_i = A_{i,i}/B_i'$. Otherwise, $\deg(C_i) < \deg(B_i')$ and $\deg(D_i) < \deg(B_i')$. C_i and D_i may be obtained by the extended Euclidean algorithm for polynomials. Then by substitution

$$\int \{A_{i,i}/B_i^i\}\,dx = \int \{C_i/B_i^{i-1}\}\,dx + \int \{D_iB_i'/B_i^i\}\,dx.$$

Using integration by parts on the second term of the right-hand side of this formula we get that

$$\int \{A_{i,i}/B_i^i\}dx = \int \{C_i/B_i^{i-1}\}dx - D_i/(i-1)B_i^{i-1} + \int \{D_i'/(i-1)B_i^{i-1}\}dx = -D_i/(i-1)B_i^{i-1} + \int \{(D_i' + (i-1)C_i)/(i-1)B_i^{i-1}\}dx.$$

Hence we see that by using this procedure we have reduced by one the power of B_i which is within the integral sign. If we continue in this manner we would next find polynomials C_{i-1}, D_{i-1} such that $C_{i-1}B_i + D_{i-1}B_i' = (D_i' + (i-1)C_i + (i-1)A_{i,i-1}) = A_{i,i-1}^*$. We would then substitute for $A_{i,i-1}^*$ in $\int \{A_{i,i-1}^*/B_i^{i-1}\}dx$ and perform integration by parts again. Eventually we arrive at

$$\int \{\sum_{j=1}^{i} A_{i,j}/B_i^j\}dx = -\sum_{j=2}^{i} D_j/(i-1)\ldots(j-1)B_i^{j-1} + \int \{T_i/B_i\}dx$$

where T_i is a polynomial either zero or $\deg(T_i) < \deg(B_i)$. By Theorem 2.9 we know that $\int T_i/B_i$ is stricly transcendental and thus we have exactly found the rational part of $\int \{\sum_{j=1}^{i} A_{i,j}/B_i^j\}\,dx$. If we apply this process for $2 \leq i \leq k$, then we will obtain two rational functions S,T such that $\int \{A/B\}dx = S(x) + \int T(x)$ and $S(x)$ is the rational part of this integral. Note that S,T are of the form $S = U/B_2B_3^2\ldots B_k^{k-1}$ and $T = V/B_1B_2\ldots B_k$.

I will now present two algorithms which represent an implementation of Hermite's method. The first algorithm applies the reduction process just described to a sum of rational functions of the form $\sum_{j=1}^{i} A_{i,j}/B_i^j$. The second algorithm takes as input a regular rational function and determines its rational part exactly. It does this by first performing partial fraction decomposition and then applying the previous algorithm iteratively. The computing time analysis for this last algorithm and hence for this implementation of Hermite's method is presented. Afterwards, a new method for determining the rational part of the integral of a rational function will be derived and analyzed.

Algorithm HERM(A,B,i)

Input: A non-null list of polynomials $\bar{A} = (A_1,\ldots,A_k)$, $B \in I[x]$ and an integer $i \geq 2$ such that $1 \leq k \leq i$, $\deg(B) > 0$, B is square-free and either $A_j = 0$ or $\deg(A_j) < \deg(B)$ for $1 \leq j \leq k$;

Output: (R,T) where R,T are regular rational functions such that $\int \{\sum_{j=1}^{k} A_j/B^{i+1-j}\}dx = R + \int T\,dx$ and R is the rational part of the integral;

1) $R \leftarrow 0$; $h \leftarrow 1$; $S \leftarrow A_1$;
2) $Z \leftarrow$ PEGCD(B,B') obtaining $\bar{C},\bar{D} \in I[x]$, $\bar{r} \in I$ such that $\bar{C}B + \bar{D}B' = \bar{r}$. (Polynomial Extended Euclidean Algorithm);
3) For $j = i,\ldots,2$ do
   3.1) $Z \leftarrow$ EGCD(S,B,B',$\bar{C}$,$\bar{D}$,r) obtaining $C,D \in I[x]$, $r \in I$ such that $CB + DB' = rS$;
   3.2) $Z \leftarrow$ INTPTS(C,D,r,h,j) obtaining $G,H \in I[x]$, $g,h \in I$ such that

$\int \{S/h\ B^j\}dx = G/g\ B^{j-1} + \int \{H/h\ B^{j-1}\}\ dx$;

3.3) $R \leftarrow R + G/gB^{j-1}$;

3.4) If $\overline{A}$ is empty or $A_i = 0$ do ($S \leftarrow H$; go to (3.6);)

3.5) $S \leftarrow H + h \cdot A_i$;

3.6) If $\overline{A}$ is empty and $S = 0$, go to (4);

4) $T \leftarrow S/h \cdot B$;

5) Return;

The exact specifications of algorithms PEGCD, EGCD and INTPTS are not given here because of their length. The operations they perform can be easily understood from the algorithm description of HERM. A computing time analysis of HERM produced the result that if $A_i \in U((nc)^n, n_i)$ $B \in U(c,n_i)$ the computing time for HERM($(A_1,\ldots,A_k)$,B,i) is $O(in^3(\ln nc)^2 + n_i^3 n^2(\ln nc)^2)$. This result will be used in the next algorithm, RHERM, which represents an implementation of Hermite's method.

Algorithm RHERM(A/B)

Input: A non-zero regular rational function A/B, with deg(B) > 0, ldcf(B) > 0 and gcd(A,B) = 1;

Output: $(R,\overline{S})$ where R is a regular rational function, $\overline{S}$ is a list of regular rational functions, $\overline{S} = (S_1,\ldots,S_k)$ such that

$$\int \{A/B\}dx = R + \int \{\sum_{i=1}^{k} S_i\}dx,$$

and R is the rational part of $\int \{A/B\}\ dx$;

1) $R \leftarrow \overline{S} \leftarrow 0$;

2) $Z \leftarrow$ RDEC(A/B) obtaining Z = (L1, L2, L3), L1 = $((A_{1,1}), (A_{2,2}, A_{2,m_2}),\ldots, (A_{k,k},\ldots,A_{k,m_k}))$, L2 = $(B_1,\ldots,B_k)$ L3 = $(w_1,\ldots,w_k)$ such that $A/B = \sum_{i=1}^{k} (1/w_i) \sum_{j=m_i}^{k} A_{i,j}/B_i^j$;

3) If k = 1, do ($\overline{S} \leftarrow A/B$; return); Otherwise, adjoin $A_{1,1}/w_1B_1$ to $\overline{S}$;

4) For i = 2,...,k do

4.1) If $B_i = 1$, do ($Q_i \leftarrow 0$; go to (4.4));

4.2) $Z \leftarrow$ HERM(A_i,B_i,i) where $A_i = (A_{i,i},\ldots,A_{i,m_i})$ obtaining P_i,Q_i such that

$$\int \{\sum_{j=m_i}^{i} A_{i,j}/B_i^j\}dx = P_i + Q_i dx;$$

4.3) $P_i \leftarrow P_i/w_i$; $Q_i \leftarrow Q_i/w_i$;

4.4) Adjoin Q_i to $\overline{S}$;

5) For i = 2,...,k do ($R \leftarrow R + P_i$);

6) $\overline{S} \leftarrow$ INVERSE($\overline{S}$); Return;

Theorem 4.1. Let A/B be a non-zero regular function and $\sum_{i=1}^{k} (1/w_i) \sum_{j=1}^{i} A_{i,j}/B_i^j$ the complete square-free partial fraction decomposition of A/B. Let n = deg(B), f_i = norm(B_i), $f = \Pi_{i=1}^{k} f_i^{\,i}$ and D/E the rational part of $\int \{A/B\}dx$. Let c = max{norm(A), norm(D),f}. Then, the computing time for RHERM(A/B) is $O(k^3n^5(\ln nc)^2)$.

Proof: Step (1) takes O(1) and by Theorems 3.2 and 3.6, step (2) takes $O(n^4(\ln nc)^2)$. Also, $A_{i,j} \in U((nc)^n,n_i)$, $|w_i| \leq (nc)^n$. Step (3) takes $O(n^2(\ln nc)^2)$. Steps (4.1), (4.4) are bounded by O(k). Step (4.2) takes $O(\sum_{i=2}^{k} \{i\ n^3(\ln nc)^2 + n_i^3n^2(\ln nc)^2\}) = O(n^5(\ln nc)^2)$. Also, at the i^{th} iteration, the numerators and denominators of $P_i,Q_i \in U((nc)^{(i+1)n+2},in_i)$. Therefore, step (4.3) takes $O(\sum_{i=2}^{k} i\ n_i\ i\ n^2(\ln nc)^2) = O(k\ n^3(\ln nc)^2)$. The time for step (5) is bounded by the time to reduce the rational function R to lowest terms which is

$$O(\sum_{i=2}^{k} (i\ n_i\ i\ n^2(\ln nc)^2) = O(n^2(\ln nc)^2 \sum_{i=2}^{k} i^5n_i^3) = O(k^3n^5(\ln nc)^2).$$

Hermite's method has been intensively studied, algorithms specified and computing time analyses done. In searching for better bounds for the coefficients of the outputs of this method an entirely new method was discovered. Moreover, the algorithm which is derived from this method will be at least one order of magnitude faster than this efficient implementation of Hermite's method. This method is now presented.

Let A/B be a non-zero regular rational function, $\deg(B) > 0$ and let $A/B = \sum_{i=1}^{k} \sum_{j=1}^{i} A_{i,j}/B_i^j$ constitute the complete square-free partial fraction decomposition of A/B over the rationals. Then, by Hermite's method we are able to obtain polynomials C,D such that

$$\int \{A/B\}dx = C/(B_2B_3^2\cdots B_k^{k-1}) + \int \{D/(B_1\cdots B_k)\}dx, \qquad (4.1)$$

where $C/B_2\cdots B_k^{k-1}$ is the rational part of the integral. Suppose C and D are undetermined and note that $B_2\cdots B_k^{k-1}$ and $B_1\cdots B_k$ are easily calculated from B, see Section 3. Then, differentiating both sides of (4.1) we get

$$\frac{A}{B} = \frac{C'(B_2\cdots B_k^{k-1}) - C(B_2\cdots B_k^{k-1})'}{(B_2\cdots B_k^{k-1})^2} + \frac{D}{B_1\cdots B_k}$$

$$= \{C'(B_1\cdots B_k)(B_2\cdots B_k^{k-1}) - C(B_1\cdots B_k)(B_2\cdots B_k^{k-1})' + D(B_2\cdots B_k^{k-1})^2\} / (B_1\cdots B_k)(B_2\cdots B_k^{k-1})^2 \qquad (4.2)$$

We now prove a theorem about the form of $(B_2B_3^2\cdots B_k^{k-1})'$.

Theorem 4.2 Let $B = \prod_{i=1}^{k} B_i^i$, $k \geq 2$. Then

$$B' = (\prod_{i=2}^{k} B_i^{i-1}) (\sum_{i=1}^{k} i\, B_1\cdots B_{i-1}B_i' B_{i+1}\cdots B_k).$$

Basis: $k = 2$ and $B = B_1B_2^2$.

Then $B' = B_1'B_2^2 + 2B_1B_2B_2' = B_2(B_1'B_2 + 2B_1B_2')$.

Induction: Let $k > 2$, $B = B_1B_2^2\cdots B_k^k$ and suppose that $(B_1B_2^2\cdots B_{k-1}^{k-1})' = (B_2B_3^2\cdots B_{k-1}^{k-2})$ $(\sum_{i=1}^{k} i\, B_1\cdots B_{i-1}B_i'B_{i+1}\cdots B_{k-1})$.

Then $B' = (B_1B_2^2\cdots B_{k-1}^{k-1})'B_k^k + (B_1B_2^2\cdots B_{k-1}^{k-1})kB_k^{k-1}B_k'$

$= (B_2B_3^2\cdots B_{k-1}^{k-2}B_k^{k-1})$ $(\sum_{i=1}^{k-1} i\, B_1\cdots B_{i-1}B_i'B_{i+1}\cdots B_{k-1}B_k)$ $+ (B_2B_3^2\cdots B_{k-1}^{k-2}B_k^{k-1})$ $(k\, B_1B_2\cdots B_{k-1}B_k')$

$= (B_2B_3^2\cdots B_k^{k-1})$ $(\sum_{i=1}^{k} i\, B_1\cdots B_{i-1}B_i'B_{i+1}\cdots B_k)$.

It follows immediately that, for $k \geq 2$, $(B_2B_3^2\cdots B_k^{k-1})' = (B_3B_4^2\cdots B_k^{k-2})$ $(\sum_{i=2}^{k}(i-1)B_2\cdots B_{i-1}B_i'B_{i+1}\cdots B_k)$. Then, applying Theorem 4.2. to equation (4.2) we get $A/B = \{C'(B_1\cdots B_k)(B_2\cdots B_k^{k-1}) - CB_1(B_2B_3^2\cdots B_k^{k-1})(\sum_{i=2}^{k}(i-1)B_2\cdots B_{i-1}B_i'B_{i+1}\cdots B_k) + D(B_2\cdots B_k^{k-1})^2\}/(B_1\cdots B_k)(B_2\cdots B_k^{k-1})^2$

$$= \{C'(B_1\cdots B_k) - C(\sum_{i=2}^{k}(i-1)B_1\cdots B_{i-1}B_i'B_{i+1}\cdots B_k)/(B_1\cdots B_k)(B_2\cdots B_k^{k-1}) + D(B_2\cdots B_k^{k-1})\} \qquad (4.3)$$

But $B = (B_1\cdots B_k)(B_2\cdots B_k^{k-1})$ and so we can equate coefficients of like powers in the numerators of (4.3). This will produce an $n \times n$ linear system where $n = \deg(B)$ and so we can solve for C and D. In Section 3 it was stated that the computing time for solving such a system is $O(n^4(\ln nf) + n^3(\ln nf)^2)$ where n is the order of the system and f is a bound for the elements of the matrix and the right-hand side. Therefore, this algorithm should be at least one order of magnitude faster than Hermite's method. Let us now take a closer look at the linear system which is formed under this new procedure.

Given the non-zero regular rational function A/B, let $n = \deg(B)$ and $B = \prod_{i=1}^{k} B_i^i$ the square-free factorization of B. Let $V(x) = \sum_{i=o}^{m} v_i x^i = \prod_{i=2}^{k} B_i^{i-1}$, $U(x) = \sum_{i=0}^{n-m} u_i x^i = \prod_{i=1}^{k} B_i$, and $W(x) = \sum_{i=0}^{n-m-1} w_i x^i = - \sum_{i=2}^{k} \{(i-1)B_1 \ldots B_{i-1} B_i' B_{i+1} \ldots B_k\}$. We will determine the polynomials C and D by equating coefficients in the expression

$$A = C'U + CW + DV.$$

Since $\int \{A/B\}dx = C/V + \int \{D/U\}dx$, C/V and D/U are regular rational functions and we can write $C(x) = \sum_{i=0}^{m-1} c_i x^i$, $D(x) = \sum_{i=0}^{n-m-1} d_i x^i$, and $C'(x) = \sum_{i=0}^{m-2} (i+1) c_{i+1} x^i$.

Then $C'U = \sum_{i=0}^{n-2} e_i x^i$ and

$$e_i = \sum_{j=0}^{m-2} (j+1)\, c_{j+1} u_{i-j},$$

$$CW = \sum_{i=0}^{n-2} f_i x^i \text{ and}$$

$$f_i = \sum_{j=0}^{m-1} c_j w_{i-j}, \text{ and}$$

$$DV = \sum_{i=0}^{n-1} g_i x^i \text{ and}$$

$$g_i = \sum_{j=0}^{m} d_{i-j} v_j.$$

Therefore, if $A(x) = \sum_{i=0}^{n-1} a_i x^i$, then

$$a_i = \sum_{j=0}^{m} \{(j+1) c_{j+1} u_{i-j} + c_j w_{i-j} + d_{i-j} v_j\}.$$

In particular, we have

$$a_{n-1} = d_{n-m-1} v_m,$$

$$a_{n-2} = c_{m-1} w_{n-m-1} + (m-1) c_{m-1} u_{n-m} + d_{n-m-1} v_{m-1} + d_{n-m-2} v_m,$$

$$a_{n-3} = (m-1) c_{m-1} u_{n-m-1} + (m-2) c_{m-2} u_{n-m} + c_{m-1} w_{n-m-2} + c_{m-2} w_{n-m-1} + d_{n-m-3} v_m + d_{n-m-2} v_{m-1} + d_{n-m-1} v_{m-2},$$

$$\ldots\ldots\ldots\ldots\ldots\ldots$$

$$a_2 = 3c_3 u_0 + 2c_2 u_1 + c_1 u_0 + c_2 w_0 + c_1 w_1 + c_0 w_2 + d_2 v_0 + d_1 v_1 + d_0 v_2,$$

$$a_1 = 2c_2 u_0 + c_1 u_1 + c_1 w_0 + c_0 w_i + d_1 v_0 + d_0 v_1,$$

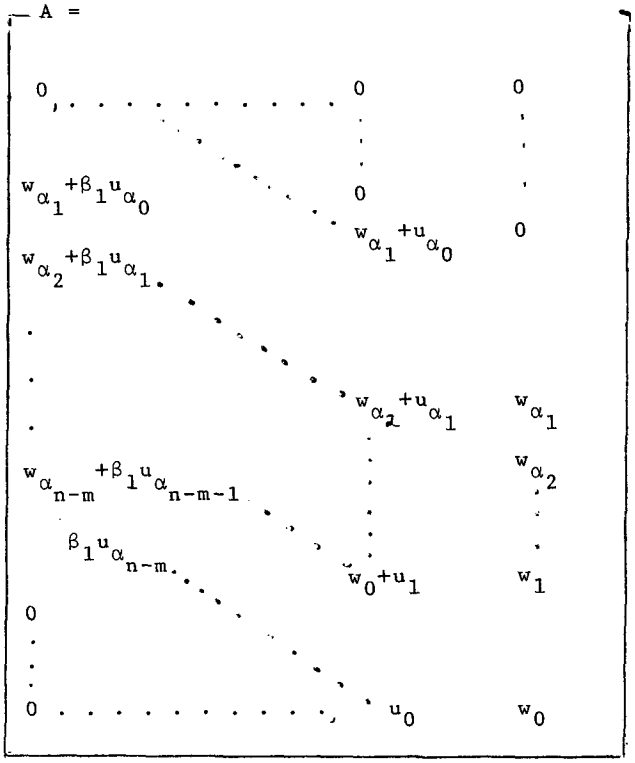
$$a_0 = c_1 u_0 + c_0 w_0 + d_0 v_0.$$

Hence if $X = (c_{m-1}, \ldots, c_0, d_{n-m-1}, \ldots, d_0)$ and $F = (a_{n-1}, \ldots a_0)$, then X is the unique vector satisfying $EX = F$, where E is the following matrix: let $\alpha_i = n-m-i$ and $\beta_i = m-i$, then

$$E = \begin{bmatrix} A & B \end{bmatrix}$$

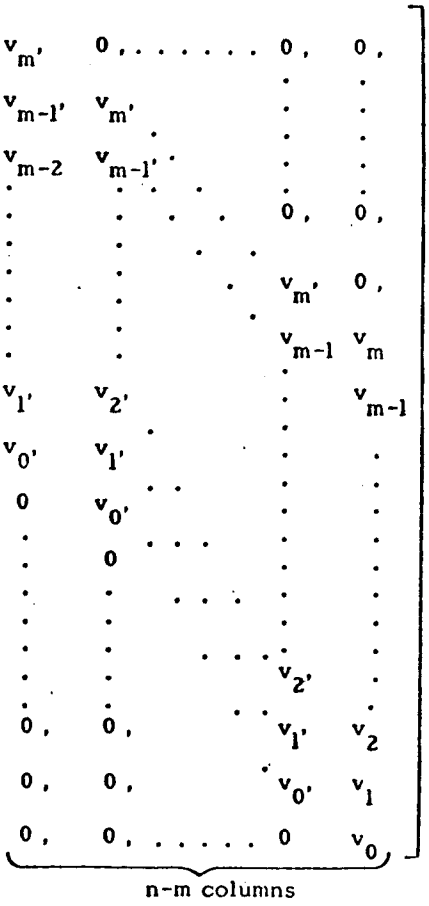
and

$A =$



m columns

B =



Algorithm RINTG(A/B)

Input: A non-zero regular rational function A/B, deg(B) > 0 and gcd(A,B) = 1;

Output: (R,S) where R,S are either zero or regular rational functions, R is the rational part of $\int\{A/B\}dx$ and $\int\{A/B\}dx = R + \int_S dx$;

1) b ←cont(B); $\overline{B}$← pp(B); U← 1;

2) Z← PSQFRE(B); returning Z = $(B_1,\ldots,B_k)$ where $B = \Pi_{i=1}^{k} B_i^i$ is the square-free factorization of B;

3) If k=1, do (R←0; S←A/B; return);

4) For i = 1,...,k do (U←U•B_i); V←$\overline{B}$/U; i←deg(V)-1; j←deg(U)-1;

5) Form the matrix E previously defined;
Form the vector F previously defined;

6) G←MUSSLE(E,F) obtaining $G = (g_o,g_1,\ldots,g_n)$ where n = deg(B), g_o = det(E) and $E\overline{G} = F$ where $\overline{G} = (g_1/g_o,\ldots,g_n/g_o)$

7) The first i elements of $\overline{G}$ are the coefficients of the numerator of R

8) The next j elements of $\overline{G}$ are the coefficients of the numerator of S

9) R←R/b•V; S←S/b•U;

10) Return;

Theorem 4.3 Let A/B be a non-zero regular rational function, n = deg(B) > 0, $B = \Pi_{i=1}^{k} B_i^i$ the square-free factorization of B. Let f_i = norm(B_i) and f = max {norm(A), $b\Pi_{i=1}^{k} f_i^i$}. Then, the computing time for RINTG(A/B) is $O(n^5(\ln nf)^2)$.

Proof: Step (1) takes $O(n(\ln f)^2)$. By Theorem 3 step (2) takes $O(kn^3(\ln nf)^2)$. Steps (3),(4) and (5) are all bounded by $O(n^2(\ln f)^2)$. The time for MUSSLE is $O(n^4(\ln nf) + n^3(\ln nf)^2)$. Now the numerators of R and S are polynomials in $U((nf)^n,n)$. Therefore, in step (9) when we reduce the rational functions to lowest terms, the computing time is bounded by the gcd operation. This takes $O(n^3(n(\ln nf))^2)$ = $O(n^5(\ln nf)^2)$.

The theoretical computing time for Hermite's method was shown to be $O(k^3n^5(\ln nc)^2)$. Remember, $n = \sum_{i=1}^{k} in_i$ where n_i = deg(B_i) and so in general all we can say about k is that is is bounded above by n. For the rational functions used in the table below, $k \simeq n^{\frac{1}{2}}$. In any case, the theoretical computing time for the new method, $O(n^5(\ln nf)^2)$, is clearly superior.

The empirical studies which were done agree with these theoretical bounds. Below, I present the results from one set of rational functions which were input to both algorithms. R_i is a rational function of the form $R_i = A_i/B_{i,1}B_{i,2}^2 \ldots B_{i,i}^i$ where $A_i,B_{i,j} \in I[x]$ and they have random coefficients in the range

$[-2^9, 2^9]$. Also, $\deg(B_{i,j}) = 1$ for $1 \le j \le i$ and $\deg(A_i) = \deg(B_{i,1} \cdot B_{i,2}^2 \cdots B_{i,i}^i) - 1$.

Comparsion of Computing Times for the New Method vs. Hermite's Method (RHERM) in Seconds

Function	RINTG	RHERM
R_1	.001	.017
R_2	.034	.542
R_3	.886	3.432
R_4	4.412	22.246
R_5	82.684	271.523

5. Extensions

There are three questions which arise in connection with the previous work. Can the new partial fraction decomposition and integration methods be extended to multivariate rational functions? What results can be derived for iterated integrals? How can the transcendental part be exactly obtained? In this section I will mention some results and outline some approaches which apply to the solution of these questions.

In a natural way we many extend the notions of square-free factorization to polynomials in n variables. Both of these definitions will then make reference to a specific variable. The notion of the rational part of the integral of a multivariable rational function can also be defined. Then, the final methods of Section 3 and 4 will apply to n variable rational functions for all $n \ge 1$. The only difference is that instead of matrices with integer entries, the entries will be polynomials in n-1 variables. These polynomials are easily obtainable when the machine representation for polynomials is in recursive canonical form (i.e. given $A(x_1,\ldots,x_n)$ the coefficients of x_i are polynomials in $x_1,\ldots,x_{i-1}$ for $2 \le i \le n$). A capability for solving exactly a linear system with multivariable polynomial entires is needed. Algorithms for this operation are currently being developed.

Suppose we wish to find the m^{th} iterated integral $\underbrace{\int\cdots\int}_{m} R(x)\underbrace{dx\ldots dx}_{m}$ for some univariate rational function $R(x)$. We can apply our method iteratively e.g.

$$\int R(x)dx = S_1(x) + \int T_1(x)dx$$

$$\int\int R(x)dx\,dx = S_2(x) + \int T_2(x)dx + \iint T_1(x)dx.$$

Finally we have that

$$\int\cdots\int R(x)dx\ldots dx = S_m(x) + \int T_m(x)dx + \ldots + \int\cdots\int T_1(x)dx\ldots dx.$$

The computing time for determining $S_m, T_m, \ldots, T_1$ would be

$$O\Big(\sum_{i=1}^{m} n^5(\ln nf)^2\Big) = O(mn^5(\ln nf)^2),$$

if $R(x) = A(x)/B(x)$, $n = \deg(B)$ and f bounds all of the coefficients in the numerators and denominators of $S_m, T_m, \ldots, T_1$. We can go further if we recognize that the denominators of the T_i are $\prod_{j=1}^{k} B_j$ for $1 \le i \le m \le k$ where $B(x)$ has the square-free factorization $\prod_{i=1}^{k} B_i^i$. In fact, we can generalize the method of Section 4 to produce a single linear system whose solution vector contains the coefficients of $S_m, T_m, \ldots, T_1$. The consequences of this result are twofold. First, a single bound for these coefficients in terms of the coefficients of $R(x)$ is produced. Secondly, the total computing time bound for determining $S_m, T_m, \ldots, T_1$ is reduced to $O(n^5(\ln nf)^2)$.

Let us return to the one dimensional case of $\int R(x)dx = S(x) + \int T(x)dx$ where $S(x)$ is the rational part of the integral. There are two approaches which can be followed to obtain a more precise answer than $\int T(x)dx$, one numeric and one symbolic. $T(x)$ has the form $T(x) = U(x)/B_1(x)\ldots B_k(x)$, where $B_i(x)$ is square-free for $1 \le i \le k$. If we are given a range of integration, $[a,b]$, we can use a numerical integration technique and apply it to $\int_a^b T(x)dx$. Care must be taken if the poles of $T(x)$ lie within $[a,b]$. In any case this method will work quite well to provide us with a numerical result for the transcendental part of the integral.

If we try to continue our symbolic approach, we must do the following: 1) factor the denominator, 2) perform a partial fraction decomposition and 3) check if the numerators are constant multiples of the derivative of their denominators. If (3) is satisfied for all partial fractions, then we are done. If not, we continue these three steps, factoring first into irreducibles over $Q(I)$ and then factoring over successively larger algebraic extensions fields of $Q(I)$. Tobey, in [11] has given a more thorough treatment of these problems. At this time, algorithms for performing these operations are extremely time consuming and suffer from exponential growth.

1) Collins, G. E., "Computing Time Analyses for Some Arithmetic and Algebraic Algorithms," Proceedings of the 1968 Summer Institute on Symbolic Mathematical Computation, I.B.M. Corp., June, 1969.

2) Collins, G. E. and Horowitz, E., "The SAC-1 Partial Fraction Decomposition and Rational Function Integration System," Computer Sciences Department, University of Wisconsin, Technical Report No. 80, February, 1970.

3) Engelman, C., "MATHLAB: A Program For On-line Assistance in Symbolic Computations," Proceedings 1965 F.J.C.C., Spartan Books, Washington, D.C.

4) Hardy, G. H. The Integration of Functions of a Single Variable, second ed., Cambridge University Press, Cambridge, England 1916.

5) Hermite, Charles, Oeuvres de Charles Hermite, edited by Emil Picard, Vol. III, Paris, Gauthier-Villars, Imprimeur-Libraire, 1912.

6) Horowitz, Ellis. Algorithms for Symbolic Integration of Rational Functions, Ph.D. Dissertation, University of Wisconsin, Madison, Wisconsin, Nov. 1969.

7) Knuth, D. E., The Art of Computer Programming, Vol. II: Semi-Numerical Algorithms, Chapter IV, Arithmetics, Addison-Wesley, 1969.

8) Moses, Joel. Symbolic Integration. Ph.D. Dissertation, Massachusetts Institute of Technology, Cambridge, Mass., Sept. 1967.

9) Sammet, J. E. An annotated descriptor-based bibliography on the use of computers for doing non-numerical mathematics. Computing Reviews 7(1966), B1-B31.

10) Sammet, J. E. Modification No. 1 to an annotated descriptor-based bibliography on the use of computers for doing non-numerical mathematics. ACM SIGSAM Bulletin, No. 5(Dec. 1966) Appendix, pp. 1-19.

11) Tobey, R. G. Algorithms for Anti-Differentiation of Rational Functions, Ph.D. thesis, Harvard University, 1967.