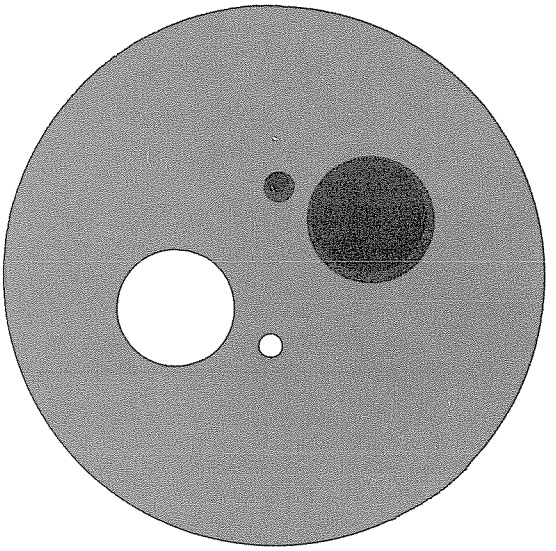
COMPUTER SCIENCES
DEPARTMENT

University of Wisconsin-
Madison



THE SAC-1 PARTIAL FRACTION DECOMPOSITION AND RATIONAL FUNCTION INTEGRATION SYSTEM*

by

G. E. Collins and E. Horowitz

Technical Report #80

February 1970

Computer Sciences Department
The University of Wisconsin
1210 West Dayton Street
Madison, Wisconsin 53706

*This research was supported in part by the National Science Foundation under Grant Number GJ-239. This report is also being distributed as the University of Wisconsin Computing Center Technical Report #12.

TABLE OF CONTENTS

1. INTRODUCTION

The SAC-1 Partial Fraction Decomposition and Rational Function Integration System is the sixth subsystem of the SAC-1 System for Symbolic and Algebraic Calculation. The five previously completed subsystems are the List Processing System, [4], the Polynomial System, [5], the Rational Function System, [6], and the Modular Arithmetic System [7].

All SAC-1 subsystems are programmed in strict accordance with A.S.A. specifications, [1]. In this subsystem, the subprograms are designed to perform the operations of partial fraction decomposition and integration on univariate rational functions with infinite precision integer coefficients. These operations will be precisely defined in Section 2.

In using the previous subsystems three common blocks were required, of the following forms: COMMON/TR1/AVAIL, STAK, RECORD(72), COMMON/TR2/SYMLST, COMMON/TR3/BETA. For the use of this subsystem all previous subsystems are necessary and a fourth common block is required: COMMON/TR4/PRIME, PEXP. PRIME = $(p_r, \ldots, p_r)$ is the location of a non-null list of distinct single-precision odd primes and PEXP = $\lfloor \log_2 (\min \{p_1, \ldots, p_r\}) \rfloor$. That is, PEXP is a Fortran integer equal to the exponent of the largest power of 2 less than the smallest prime p_i, $1 \leq i \leq r$. The list PRIME can be formed by using the subprogram GENPR described in the modular arithmetic system [7]. PEXP

may be calculated by using subprogram ELPOF2 described in Section 3.4 of this manual.

Section 2 provides an introduction to the theory of partial fraction decomposition and rational function integration. In Section 3 the input and output specifications are given for all programs in this system. All these programs are written in A.S.A. Fortran and no new machine language primitives have been introduced. The listings of these programs can be found in Section 6 of this manual.

In Section 4 an algorithm description is given for each subprogram. This is done to add greater clarity to the subprogram specifications in Section 3 and to aid in the detailed understanding of the programs in the appendix. Following each algorithm is a theoretical maximum computing time for each subprogram, expressed in O-notation as a function of parameters describing the inputs.

Both the partial fraction decomposition and rational function integration algorithms require the solution of certain types of linear systems. At present, both of these capabilities are restricted to univariate rational functions. Extensions to multivariate rational functions depend upon the development of programs for solving linear systems with multivariate polynomial entries. It is expected that this extension will be made by late 1970. In any case, the subprograms in Section 3.1 can solve a linear system with integer entries. The algorithms used are based on more general algorithms developed in [2]; they were programmed by Michael T. McClellan.

2. THEORETICAL BACKGROUND

In SAC-1 a rational function A/B is considered to be a numerator-denominator pair of polynomials over the integers where $\mathrm{ldcf}(B) > 0$ and $\gcd(A, B) = 1$. Suppose we let I designate the integral domain of the integers and $\mathbb{C}$ the field of complex numbers.

Definition: Let $R(x) = A(x)/B(x)$ be a rational function, $A(x), B(x) \in I[x]$. Then $R(x)$ is called regular if $\deg(A) < \deg(B)$.

We note that every rational function may be uniquely represented as the sum of a polynomial with rational coefficients (its polynomial part) and a regular rational function.

Theorem 2.1. Let $R(x)$ be a regular rational function. Then

$$\int R(x)dx = S(x) + \sum_{i=1}^{k} c_i \log(x - b_i)$$

where $S(x)$ is a regular rational function over I, b_i are the distinct roots of the denominator of $R(x)$, $b_i, c_i \in \mathbb{C}$ for $1 \le i \le k$.

A proof of this theorem can be found in [8], Section 2.3. Its major consequence is that the rational functions are not closed under the operation of integration. However, as developed in [8], one can show that no sum of the form $\sum_{i=1}^{k} c_i \log (x - b_i)$ can be non-trivially equal to a rational function. Therefore, we can say that the integral of a rational function is equal to the sum of a rational function (its rational part) and the integral of a rational function whose rational part is zero (its transcendental part).

The rational and transcendental part of the integral of a rational function can be shown to be unique.

In 1872, Hermite developed a method whereby, given a rational function $R(x)$, one could find rational functions, $S(x)$, and $T(x)$, such that $\int R(x)dx = S(x) + \int T(x)dx$ and $S(x)$ is the rational part of $\int R(x)dx$. This method has been studied in detail in [8] and a new, more efficient method for calculating $S(x)$, $T(x)$ has been determined. The subprograms in Section 3.3 are based on this new method.

The operation of partial fraction decomposition is usually closely connected with rational function integration. Though the method in Section 3.3 for finding the rational part entirely avoids the use of partial fraction decomposition, nevertheless this operation considered by itself is extremely useful. Hence, this capability has been included in this subsystem. We now define the type of partial fraction decomposition to which we refer.

Definition: Let B be a polynomial of positive degree over I, $\mathrm{ldcf}(B) > 0$. Let $b = \mathrm{cont}(B)$ and $B = b \prod_{i=1}^{k} B_i^{\,i}$ where $\deg(B_k) > 0$, $\mathrm{ldcf}(B_i) > 0$, and the B_i are primitive and pairwise relatively prime for $1 \le i \le k$. Then $b \prod_{i=1}^{k} B_i^{\,i}$ is called the square-free factorization of B.

We note that B_i contains the roots of B of multiplicity i. The B_i are said to be square-free (i.e. containing only roots of multiplicity one), but they are not necessarily irreducible over I.

Definition: Given a regular rational function A/B, suppose $A/B = \sum_{i=1}^{k} A_i/v_i\, B_i^i$ where $A_i, B_i \in I[x]$, $v_i \in I$, $b\prod_{i=1}^{k} B_i^i$ is the square-free factorization of B and either $B_i = 1$, $A_i = 0$ and $v_i = 1$ or $\deg(A_i) < \deg(B_i^i)$ and $\gcd(v_i, \text{cont}(A_i)) = 1$ for $1 \le i \le k$. Then this sum is called the square-free partial fraction decomposition of A/B over I.

When $\deg(B_i) > 0$, the corresponding numerator A_i, satisfies $\deg(A_i) < \deg(B_i^i)$, and we can decompose these partial fractions A_i/B_i^i even further to obtain the following decomposition.

Definition: Let A/B be a regular rational function and suppose $A/B = \sum_{i=1}^{k} (1/w_i) \sum_{j=1}^{i} A_{i,j}/B_i^j$ where $A_{i,j}\, B_i \in I[x]$, $w_i \in I$ and $b\prod_{i=1}^{k} B_i^i$ is the square-free factorization of B. Also, if $B_i = 1$, then each $A_{i,j} = 0$, and $w_i = 1$, while otherwise $\deg(A_{i,j}) < \deg(B_i)$ for $1 \le j \le i$, $1 \le i \le k$. Then, this sum is called the complete square-free partial fraction decomposition of A/B over I.

The subprograms in Section 3.2 provide for obtaining both the square-free and the complete square-free partial fraction decomposition of a regular rational function.

3. PROGRAM DESCRIPTIONS

3.1. Solution of Linear Systems

(1) Subroutine ADJCOL(V, M)

M is a non-null list, $(M_1, \ldots, M_n)$, of length $n > 0$ in which each M_i is a list and such that the reference count of every final segment of M (including M itself) is one. V is a list satisfying the same conditions and having the same length. Then M is modified by prefixing V_i to M_i for $1 \le i \le n$ and V is altered to become the null list.

(2) Subroutine CUSSLE(p, L)

p is an odd prime represented as a Fortran integer and L is a second order list $(L_1, \ldots, L_m)$ representing an $m \times (m+1)$ matrix M over GF(p) by rows. L_i is a list of $m + 1$ atoms, each atom an element of GF(p). M is of the form (A, -b) corresponding to the linear system of equations $Ax = b \pmod p$. The matrix L is erased. If A is singular, then the new value of L is the Fortran integer -1 . If A is non-singular, then L is a list of $m + 1$ atoms, $(v_1, \ldots, v_m, v_0)$, each atom an element of GF(p). Then $v_0 = \det(A)$ and if $x_i = v_i/v_0 \pmod p$ for $1 \le i \le m$ and $x = (x_1, \ldots, x_m)$, then x is the unique solution of the linear system $Ax = b \pmod p$.

(3) Z = MUSSLE(A, B)

A is a third order list, $(A_1, \ldots, A_m)$ representing an $m \times m$ matrix over the integers by rows. $A_i = (A_{i,1}, \ldots, A_{i,m})$, where $A_{i,j}$ is an L-integer for $i \leq i,j \leq m$. B is an $m \times 1$ vector represented as a list $(B_1, \ldots, B_m)$ of L-integers. Then, if the list of primes is exhausted, Z = -2. Otherwise, if A is singular, Z = -1. If A is non-singular, then Z is the list of $m + 1$ L-integers, $(Z_0, Z_1, \ldots, Z_m)$, where $Z_0 = \det(A)$ and $X = (Z_1/Z_0, \ldots, Z_m/Z_0)$ is the unique solution vector of the linear system $AX = B$.

(4) Z = PVECT(A, K, N)

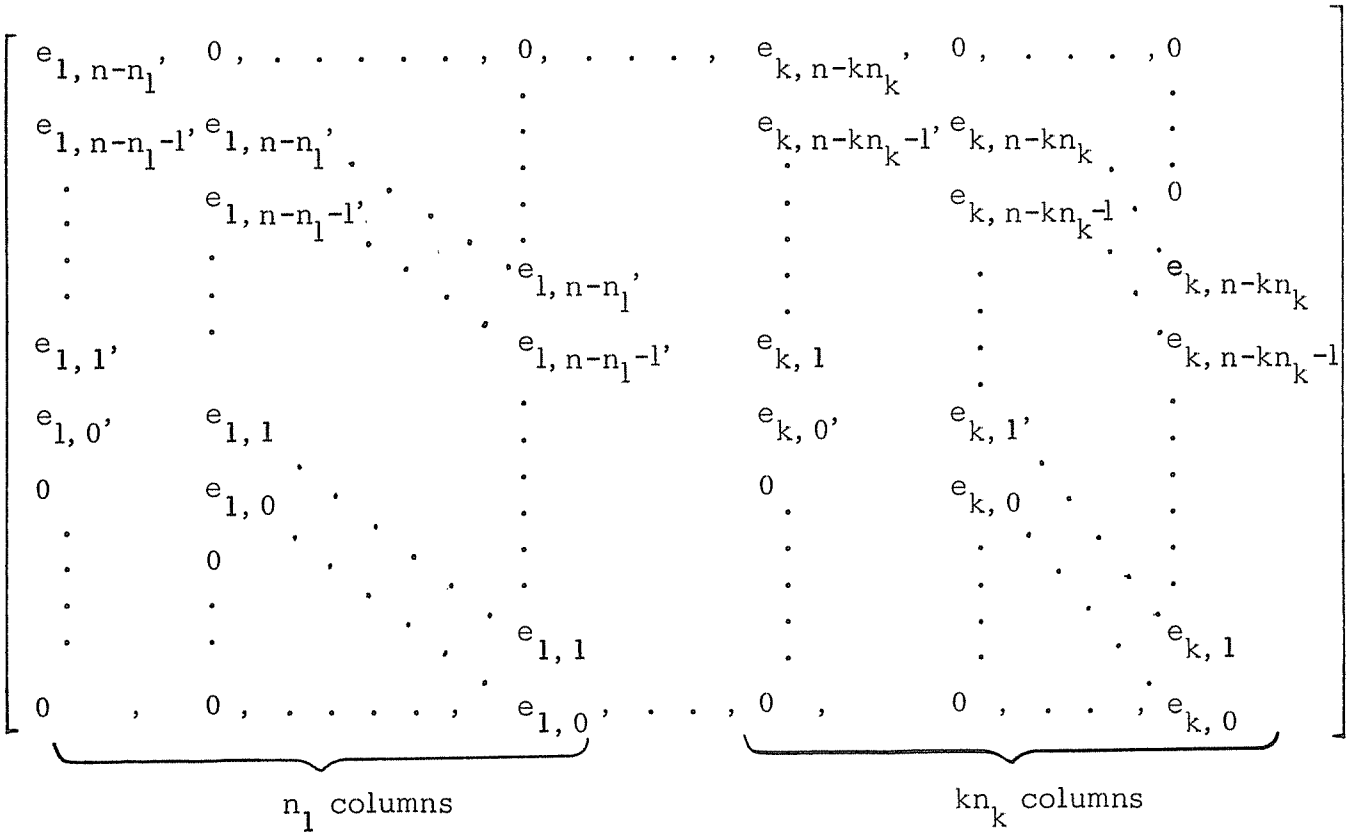
A is a univariate polynomial over the integers, K and N are Fortran integers such that $K \geq 0$ and $\deg(A) + K < N$. Then, if $A(x) \cdot x^k = \Sigma_{i=0}^{n-1} a_i x^i$, Z is the list $(a_{n-1}, \ldots, a_1, a_0)$. If A = 0, then Z is the list of lenth N equal to ((), . . . , ()).

3.2. Partial Fraction Decomposition

(1) Z = MATSFD(B, F)

B is a non-zero univariate polynomial over the integers such that ldcf(B) > 0, cont(B) = 1 and $n = \deg(B) > 0$. F is the list $(B_1, \ldots, B_k)$ such that $\Pi_{i=1}^{k} B_i^i$ is the square-free factorization of B. Then Z is a third order list, $(Z_1, \ldots, Z_n)$, which is the row-wise representation of the

following $n \times n$ matrix M. Let $n_i = \deg(B_i)$ and $E_i(x) = B(x)/B_i^i(x) = \Sigma_{j=0}^{n-in_i} e_{i,j} x^j$; then $M =$



(2) Z = PCDEC(A, B)

A and B are univariate polynomials over the integers, where $n = \deg(A)$, $n = \deg(B)$ and $m \geq n > 0$. Let $k = [m/n] + 1$, $S = \text{ldcf}(B)^{m-n+1}$ and $A/B^k = (1/S) \Sigma_{j=1}^{k} A_j/B^j$, where $\deg(A_j) < \deg(B)$ for $1 \leq j \leq k$. Then $Z = (R, S)$, where $R = (A_k, \ldots, A_1)$.

(3) Z = PSQFRE(A)

A is a non-zero, primitive, univariate polynomial over the integers, deg (A) > 0 and ldcf (A) > 0 . Then Z is the list $(A_1, \ldots, A_k)$ where $A = \Pi_{i=1}^{k} A_i^i$ is the square-free factorization of A , ldcf $(A_i) > 0$, A_i is primitive for $1 \leq i \leq k$, and deg $(A_k) > 0$.

(4) Z = RDEC(R)

R is a non-zero, univariate, regular rational function, A/B, where deg (B) > 0, ldcf (B) > 0 and gcd (A, B) = 1. Then Z is the list (L_1, L_2, L_3), where $L_1 = ((A_{1,1}), (A_{2,2}, A_{2,s_2}), \ldots, (A_{k,k}, \ldots, A_{k,s_k}))$, $L_2 = (B_1, \ldots, B_k)$, and $L_3 = (w_1, \ldots, w_k)$. $A_{i,j}$ and B_i are polynomials over the integers, w_i are L-integers, and $A/B = \Sigma_{i=1}^{k} (1/w_i) \Sigma_{j=s_i}^{i} A_{i,j}/B_i^i$ is a complete square-free partial fraction decomposition of A/B. If $B_i = 1$, then $s_i = i$, $A_{i,i} = 0$ and $w_i = 1$. Otherwise, deg $(B_i) > 0$, $1 \leq s_i \leq i$, $w_i > 0$, deg $(A_{i,j}) <$ deg (B_i) for $s_i \leq j \leq i$ and $1 \leq i \leq k$, and $A_{i,s_i} \neq 0$.

(5) Z = RSQDEC(R)

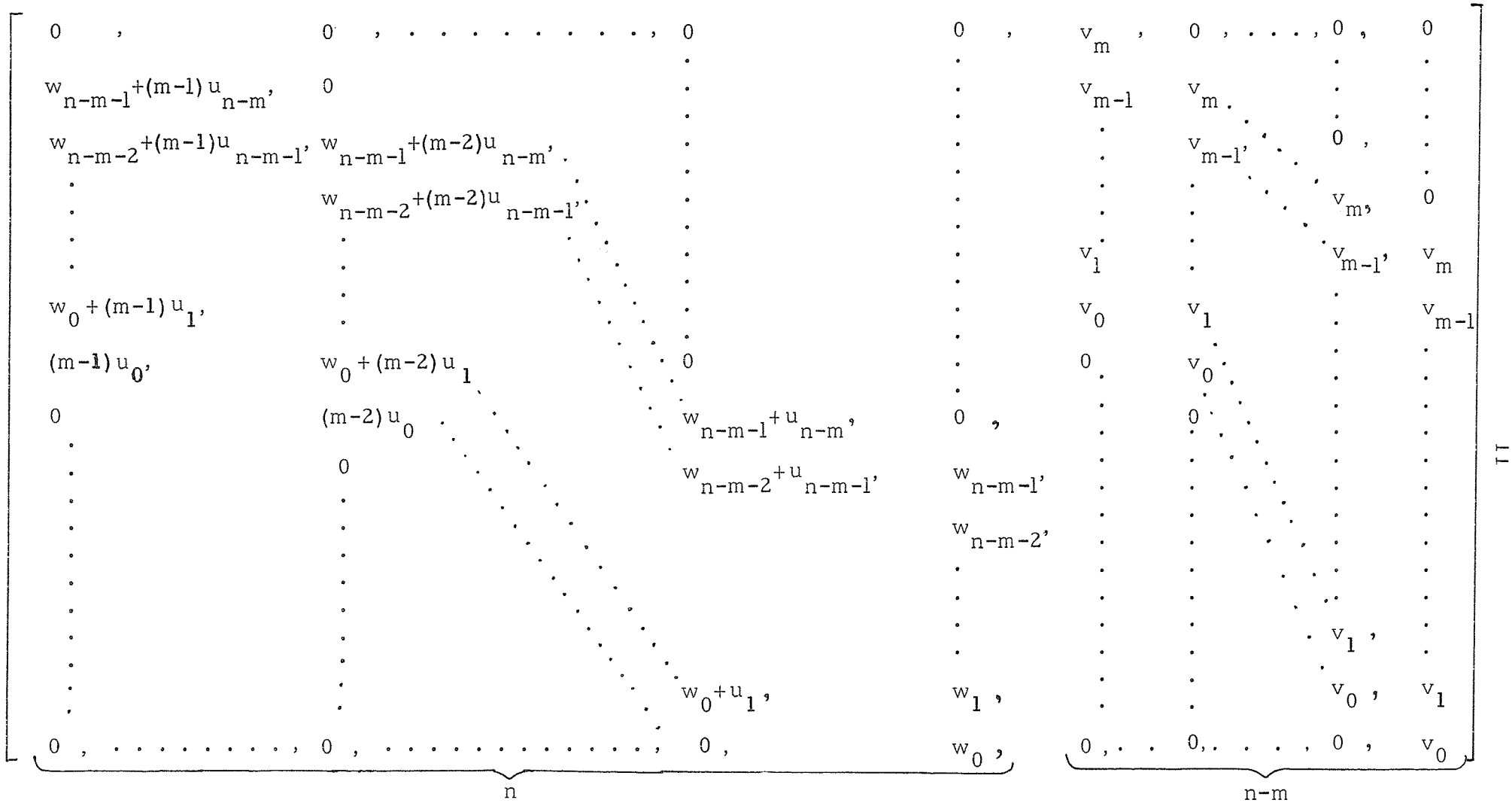
R is a non-zero, univariate regular rational function, A/B, where deg (B) > 0 , ldcf (B) > 0, and gcd (A, B) = 1. Then Z is the list (L_1, L_2, L_3) where $L_1 = (A_1, \ldots, A_k)$, $L_2 = (B_1, \ldots, B_k)$ and $L_3 = (w_1, \ldots, w_k)$. A_i and B_i are polynomials over the integers and w_i are L-integers, $1 \leq i \leq k$. $A/B = \Sigma_{i=1}^{k} A_i/w_i B_i^i$ is the square-free partial

fraction decomposition of A/B. If $B_i = 1$, then $A_i = 0$ and $w_i = 1$. Otherwise, deg $(A_i) <$ deg (B_i^i), $w_i > 0$, and gcd $(w_i,$ cont $(A_i)) = 1$.

3.3. Rational Function Integration

(1) Z = MATX(F, U, V)

F = $(B_1, \ldots, B_k)$, U and V are non-zero polynomials with integer coefficients such that $U = \Pi_{i=1}^{k} B_i$, $k \geq 2$, $V = \Pi_{i=2}^{k} B_i^{i-1}$, and $\Pi_{i=1}^{k} B_i^i$ is the square-free factorization of some positive, primitive polynomial B of positive degree n . Then Z is a third order list $(Z_1, \ldots, Z_n)$ which is the row-wise representation of the following $n \times n$ matrix M . Let $V(x) = \Sigma_{i=0}^{m} v_i x^i$, $U(x) = \Sigma_{i=0}^{n-m} u_i x^i$ and $W(x) = \Sigma_{i=0}^{n-m-1} w_i x^i = - \Sigma_{i=2}^{k} \{(i-1)B_1 \cdots B_{i-1} B_i' B_{i+1} \cdots B_k\}$. Then M =


n
n-m

(2) Z = PINTG(R)

R is a univariate rational function A/B in SAC-1 canonical form with deg (B) = 0. Then Z is the rational function $Z = \int R$.

(3) Z = RINTG(R)

R is a non-zero, univariate, regular rational function, A/B, in SAC-1 canonical form and such that deg (B) > 0. Z = (S, T), where S, T are regular rational functions, $\int R = S + \int T$, and S is the rational part of $\int R$.

(4) Z = RINTGS(R)

R is a univariate rational function in SAC-1 canonical form. Z = (S, T) where S and T are the rational functions such that $\int R = S + \int T$ and S is the rational part of $\int R$. If R = 0, then Z = ((), ()) .

3.4. Miscellaneous

(1) SUBROUTINE ELPOF2(X, Y, Z)

The only input parameter is X, a non-zero L-integer. The outputs Y and Z are the Fortran integers $Y = \lfloor \log_2 |X| \rfloor$ and $Z = \lceil \log_2 |X| \rceil$.

(2) Z = LCM(L)

$L = (a_1, \ldots, a_n)$ is a non-null list of L-integers, at least one of which is non-zero. Z is the positive L-integer which is the least common multiple of $\{a_1, \ldots, a_n\}$.

(3) Z = PQREM(A, B)

A and B are non-zero univariate polynomials with integer coefficients, deg (A) $\geq$ deg (B) $\geq$ 0 such that there exist polynomials Q and R with integer coefficients for which A = BQ + R and either R = 0 or deg (R) $<$ deg (B). Then Z = (Q, R) .

4. ALGORITHM DESCRIPTIONS

Algorithm ADJCOL(V, M)

Input: M is a non-null list $(M_1, \ldots, M_n)$ of length $n > 0$ in which each M_i is a list and such that the reference count of every final segment of M (including M itself) is one. V is a list satisfying the same conditions and having the same length.

Output: M is modified by prefexing V_i to M_i for $1 \leq i \leq n$ and V is altered to become the null list.

(1) $T \leftarrow M$.

(2) DECAP(VI, V); MI $\leftarrow$ FIRST(T); MI $\leftarrow$ PFL(VI, MI).

(3) ALTER(MI, T); T $\leftarrow$ TAIL(T); if $T \neq 0$, go to (2).

(4) Return.

Computing Time: O(n), where n = length (M).

Algorithm CUSSLE(p, L)

Input: p, an odd prime, and L, a second order list $(L_1, \ldots, L_m)$, representing an $m \times (m+1)$ matrix M over GF(p) by rows. M is of the form (A, -b) corresponding to the linear system $Ax = b \pmod p$.

Output: If A is singular, L = -1. Otherwise, L = $(v_1, \ldots, v_m, v_0)$, $v_i \in GF(p)$ such that $v_0 = \det(A)$ and if $x_i = v_i/v_0$, $1 \le i \le m$, then $x = (x_1, \ldots, x_m)$ is the unique solution of $Ax = b \pmod p$.

(1) $A \leftarrow L$; $B \leftarrow 0$; $g \leftarrow 1$; $i \leftarrow 0$.

(2) [Search A for a pivot row.]

(2.1) $S \leftarrow A$; $A \leftarrow 0$; $j \leftarrow 0$.

(2.2) If $S = 0$, go to (2.5).

(2.3) Remove first row, T, from S and remove first element, d, from T; if $d \neq 0$, go to (3).

(2.4) $j \leftarrow j + 1$; $A \leftarrow PFL(T, A)$; go to (2.2).

(2.5) If $A = 0$, go to (10); otherwise, go to (11).

(3) $B \leftarrow PFL(T, B)$; if $j > 0$, $i \leftarrow i + 1$.

(4) $e \leftarrow -d^{-1} \pmod p$; $g \leftarrow g \cdot d \pmod p$.

(5) Multiply each element of T by e.

(6) If $S = 0$, ($A \leftarrow INV(A)$; go to (2)).

(7) Remove next row Q from S, remove the first element d from Q; $A \leftarrow PFL(Q, A)$.

(8) If $d \neq 0$ (add $d \cdot$ (row T) to row Q).

(9) Go to (6).

(10) [Triangularization is done; solve for the v_i.]

(10.1) $g \leftarrow g \cdot (-1)^i \pmod p$.

(10.2) $V \leftarrow$ PFA(g, 0).

(10.3) DECAP(C, B); $V \leftarrow$ PFA(CVPROD(P, C, V),V); ERLA(C).

(10.4) If $B \neq 0$, go to (10.3); otherwise, go to (12).

(11) $V \leftarrow -1$.

(12) $L \leftarrow V$.

(13) Return.

Computing Time: $O(m^3)$.

Algorithm ELPOF2(X, Y, Z)

Input: X , an L-integer.

Output: Y and Z , FORTRAN integers such that $Y = \lfloor \log_2 |X| \rfloor$ and $Z = \lceil \log_2 |X| \rceil$.

(1) $V \leftarrow X$; $Y \leftarrow Z \leftarrow 0$; if $V = 0$, return.

(2) $Q \leftarrow$ IABSL(V); if PONE(Q) = 1, (ERLA(Q); return).

(3) TWO $\leftarrow$ PFA(2, 0); $RP \leftarrow 0$.

(4) $Z \leftarrow$ IQRS(Q, TWO); ERLA(Q); DECAP(Q, Z); DECAP(R, Z).

(5) If $R = 0$, go to (7).

(6) $RP \leftarrow RP + 1$; ERLA(R).

(7) If $Q = 0$, go to (9).

(8) $Y \leftarrow Y + 1$; go to (4).

(9) If RP $\neq$ 0, Z $\leftarrow$ 1.

(10) Z $\leftarrow$ Y + Z; ERLA(TWO).

(11) Return.

Computing Time: $O((\ln X)^2)$.

Algorithm B = LCM(L)

Input: L = $(a_1, \ldots, a_n)$, a non-null list of L-integers, at least one of which is non-zero.

Output: B, the positive L-integer which is the least common multiple of $\{a_1, \ldots, a_n\}$.

(1) ADV(A, L); if A = 0, go to (1).

(2) B $\leftarrow$ BORROW (A).

(3) If L = 0, go to (6); ADV(A, L); if A = 0, go to (3).

(4) L1 $\leftarrow$ IGCD(A, B); L2 $\leftarrow$ IQ(B, L1); ERLA(B); ERLA(L1).

(5) B $\leftarrow$ IPROD(A, L2), ERLA(L2); go to (3).

(6) If ISIGN(B) < 0, B $\leftarrow$ INEG(B).

(7) Return.

Computing Time: $O(n^2(\ln d)^2)$, where $d = \max\{|a_1|, \ldots, |a_n|\}$.

Algorithm M = MATSFD(B, F)

Input: $B \in I[x]$, $B \neq 0$, ldcf(B) > 0, cont(B) = 1, deg(B) > 0. F is the list $(B_1, \ldots, B_k)$ such that $\Pi_{i=1}^{k} B_i^i$ is the square-free factorization of B .

Output: M, the third order list which is the row-wise representation of the matrix defined in Section 3.2.

(1) N ← deg(B); k ← length(F); T ← CINV(F); M ← PVECT(0, 0, N); I ← k.

(2) DECAP(BI, T); If deg(BI) = 0, go to (4); CI ← BI^{I}; kI ← deg(CI); FI ← B/CI; J ← 0.

(3) V ← PVECT(FI, J, N); ADJCOL(V, M); J ← J + 1; If J < kI, go to (3).

(4) I ← I - 1; If I ≠ 0, go to (2).

(5) Return.

Computing Time: $O(n^2(\ln f)^2)$, where n = deg(B), f_i = norm(B_i) and $f = \Pi_{i=1}^{k} f_i^i$.

Algorithm MATX(F, U, V)

Input: $F = (B_1, \ldots, B_k)$ where $k \geq 2$, $U, V \in I[x]$, $U = \prod_{i=1}^{k} B_i$, $V = \prod_{i=2}^{k} B_i^{i-1}$ and $\prod_{i=1}^{k} B_i^{i}$ is the square-free factorization of some positive, primitive polynomial B of positive degree n.

Output: A third order list $M = (M_1, \ldots, M_n)$ which represents row-wise the matrix described in Section 3.3.

(1) $n \leftarrow \deg(U) + \deg(V)$; $M \leftarrow$ PVECT(0, 0, n).

(2) $F1 \leftarrow F$; $i \leftarrow 1$; $W \leftarrow 0$; ADV(BI, F1).

(3) If $F1 \neq 0$, (ADV(BI, F1); $X \leftarrow U/BI$; $W \leftarrow W + i \cdot BI' \cdot X$; $i \leftarrow i + 1$; go to (3)).

(4) $W \leftarrow -W$; $j \leftarrow 0$; $m \leftarrow \deg(V)$.

(5) $R \leftarrow$ PVECT(V, j, n); ADJCOL(R, M); $j \leftarrow j + 1$; if $j < n - m$, go to (5).

(6) $R \leftarrow$ PVECT(W, 0, n); ADJCOL(R, M); $j \leftarrow 0$; $W(x) \leftarrow x \cdot W(x)$.

(7) If $j < m - 1$, ($X \leftarrow W + (j + 1) \cdot U$; $R \leftarrow$ PVECT(X, j, n); ADJCOL(T, M); $j \leftarrow j + 1$; go to (7)).

(8) Return.

Computing Time: $O(n^2(\ln nf)(\ln f))$, where $n = \deg(B)$, $f_i = \text{norm}(B_i)$ and $f = \prod_{i=1}^{k} f_i^{i}$.

Algorithm Z = MUSSLE(A, B)

Input: A is a third order list $(A_1, \ldots, A_m)$ representing an $m \times m$ matrix over the integers by rows. $A_i = (A_{i,1}, \ldots, A_{i,m})$, where A_{ij} are L-integers for $1 \leq j \leq m$. B is an $m \times 1$ vector $B = (B_1, \ldots, B_m)$, a list of L-integers.

Output: If the list of primes is exhausted, Z = -2. Otherwise, if A is singular, Z = -1. If A is non-singular, $Z = (Z_0, \ldots, Z_m)$, a list of L-integers such that $Z_0 =$ det (A) and if $x_i = Z_i/Z_0$ for $1 \leq i \leq m$, where $X = (x_1, \ldots, x_m)$, then X is the unique solution of AX = B.

(1) Q ← PFA(1, 0); n ← length (A) + 1; C ← PVECT(0, 0, n); F ← 2; Z ← PRIME; L ← 0; h ← 0; T ← A; U ← B; i ← 1.

(2) [Compute an upper bound, k, on the number of primes needed.]

(2.1) ADV(E, U); E ← IABSL(E); ADV(S, T).

(2.2) ADV(D, S); D ← IABSL(D); if D > E, E ← D.

(2.3) If S ≠ 0, go to (2.2).

(2.4) D ← F · E; F ← i · D; i ← i + 1; If U ≠ 0, go to (2.1).

(2.5) If F = 0, go to (4.11).

(2.6) ELPOF2(F, S, T); k ← [T/PEXP] + 1.

(3) [Get next prime p and reduce the system (A, -B) modulo p.]

(3.1) ADV(p, Z); $M \leftarrow 0$; $i \leftarrow 0$; $U \leftarrow B$; $T \leftarrow A$.

(3.2) $F \leftarrow 0$; ADV(S, T).

(3.3) ADV(E, S); $F \leftarrow$ PFA(CMOD(P, E), F); if $S \neq 0$, go to (3.3).

(3.4) ADV(E, U); $F \leftarrow$ PFA(CDIF(p, 0, CMOD(p, E)), F).

(3.5) $M \leftarrow$ PFL(INV(F), M); if $U \neq 0$, go to (3.2).

(3.6) $M \leftarrow$ INV(M); go to (4).

(4) [Solve $Ax = B \pmod p$ and modify the tentative solution C.]

(4.1) CUSSLE(p, M).

(4.2) If $M = -1$, go to (4.8).

(4.3) $L \leftarrow L + 1$; $S \leftarrow C$.

(4.4) $T \leftarrow$ FIRST(S); DECAP(U, M); ALTER(CGARN(Q, T, P, U), S).

(4.5) $S \leftarrow$ TAIL(S); if $S \neq 0$, go to (4.4).

(4.6) $S \leftarrow$ PFA(p, 0); $T \leftarrow Q \cdot S$; ERLA(Q); ERLA(S); $Q \leftarrow T$; if $L = k$, go to (5).

(4.7) Go to (4.9).

(4.8) $h \leftarrow h + 1$; if $h = k$, go to (4.11).

(4.9) If $Z \neq 0$, go to (3.1)

(4.10) $C \leftarrow -2$; go to (6).

(4.11) $C \leftarrow -1$; go to (6).

(5) $S \leftarrow$ INV(C); DECAP(D, S); $C \leftarrow$ PFL(D, INV(S)).

(6) $Z \leftarrow C$.

(7) Return.

Computing Time: $O(m^3(\ln mf)^2 + m^4(\ln mf))$, where m is the order of A and f bounds all the elements of A and B.

Algorithm L = PCDEC(A, B)

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

Output: Let $k = [m/n] + 1$, $C = \mathrm{ldcf}(B)^{m-n+1}$ and $A/B^k = (1/C)\ \Sigma_{j=1}^{k} A_j/B^j$, where $\deg(A_j) < \deg(B)$ for $1 \leq j \leq k$. Then $L = (X, C)$, where $X = (A_k, \ldots, A_1)$.

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

(2) $Q \leftarrow C \cdot A$;

(3) $Z \leftarrow$ PQREM(Q, B), obtaining $A = (\overline{Q}, \overline{A})$ such that $Q = B\overline{Q} + \overline{A}$ and $\deg(\overline{A}) < n$.

(4) $X \leftarrow$ PFL($\overline{A}$, X).

(5) If $\deg(\overline{Q}) \geq n$, ($Q \leftarrow \overline{Q}$; go to (3)).

(6) $X \leftarrow$ INV(PFL($\overline{Q}$, X)); $L \leftarrow$ PFL(X, PFL(C, 0)).

(7) Return.

Computing Time: $O(k^2 n^2(\ln e)\{\ln d + k^2 n(\ln e)\})$, where $m = \deg(A)$, $n = \deg(B)$, $k = [m/n] + 1$, $d = \mathrm{norm}(A)$, and $e = \mathrm{norm}(B)$.

Algorithm S = PINTG(R)

Input: A univariate rational function R = A/B such that either R = 0 or deg(B) = 0.

Output: The rational function $S = \int R$.

(1) If R = 0, (S ← 0; return).

(2) ADV(A, R); ADV(B, R); $\overline{A}$ ← A; ADV(k, $\overline{A}$); T ← 0.

(3) P ← PFL(BORROW(k), 0).

(4) ADV(k, $\overline{A}$); ADV(k, $\overline{A}$); T ← PFL(PFA(k+1, 0), T); If $\overline{A} \neq 0$, go to (4).

(5) Z ← LCM(T); T ← INV(T); $\overline{A}$ ← TAIL(A).

(6) ADV(k, $\overline{A}$); DECAP(L, T); M ← IQ(Z, L).

(7) N ← IPROD(M, k); DECAP(I, L); P ← PFA(I, PFL(N, P)).

(8) ADV(k, $\overline{A}$); If $\overline{A} \neq 0$, go to (6).

(9) P ← INV(P); Q ← PSPROD(B, Z, 0).

(10) I1 ← PCONT(P); I2 ← PCONT(Q); I3 ← IGCD(I1, I2).

(11) If PONE(I3) ≠ 1, (P ← PSQ(P, I3); Q ← PSQ(Q, I3)).

(12) S ← PFL(P, PFL(Q, 0)).

(13) Return.

Computing Time: $O(n^3(\ln nd)^2)$ where n = deg(A) and d = max {norm(A), norm(B)}.

Algorithm L = PQREM(A, B)

Input: $A, B \in I[x]$, $A \neq 0$, $B \neq 0$, $\deg(A) \geq \deg(B) \geq 0$ such that there exist $Q, R \in I[x]$ for which $A = BQ + R$ and either $R = 0$ or $\deg(R) < \deg(B)$.

Output: $L = (Q, R)$.

(1) $R \leftarrow$ BORROW(A); $m \leftarrow \deg(R)$; $n \leftarrow \deg(B)$.

(2) If $n = 0$, ($L \leftarrow$ FIRST(TAIL(B)), $L \leftarrow$ PFL(PSQ(R, L), PFL(0, 0)); return).

(3) $LB \leftarrow$ PLDCF(B); $MB \leftarrow$ PRED(B).

(4) $Q \leftarrow$ PFL(BORROW(FIRST(R)), 0); $d \leftarrow m - n$.

(5) For $i = 1, \ldots, m - n + 1$ do:

(5.1) $s \leftarrow$ ldcf(R)/LB; Q - PFA(d, PFL(s, Q)).

(5.2) $R \leftarrow$ PRED(R) - $s \cdot MB \cdot x^d$.

(5.3) If $R = 0$, go to (6).

(5.4) $d \leftarrow \deg(R) - n$.

(5.5) If $d < 0$, go to (6).

(6) $L \leftarrow$ PFL(INV(Q), PFL(R, 0)).

(7) Return.

Computing Time: $O(n(m - n + 1)(\ln e)\{\ln d + (m - n + 1)(\ln e)\})$, where $m = \deg(A)$, $n = \deg(B)$, $d = \text{norm}(A)$ and $e = \text{norm}(B)$.

Algorithm L = PSQFRE(A)

Input: $A \in I[x]$, $A \neq 0$, A primitive, $\deg(A) > 0$, $\mathrm{ldcf}(A) > 0$.

Output: $L = (A_1, \ldots, A_k)$, where $A_i \in I[x]$ for $1 \leq i \leq k$ and $A = \prod_{i=1}^{k} A_i^i$ is the square-free factorization of A.

(1) $I \leftarrow 0$; $Q \leftarrow D \leftarrow 0$; $P \leftarrow 1$; $B \leftarrow A$.

(2) $E \leftarrow \gcd(B, B')$; if $\deg(E) \neq 0$, ($F \leftarrow B/E$; go to (4)).

(3) $F \leftarrow B$.

(4) If $I = 0$, go to (7).

(5) If $\deg(D) \neq \deg(F)$, ($Q \leftarrow \mathrm{PFL}(D/F, Q)$; go to (7)).

(6) $Q \leftarrow \mathrm{PFL}(\mathrm{BORROW}(P), Q)$.

(7) If $\deg(E) \neq 0$, ($I \leftarrow 1$; $B \leftarrow E$; $D \leftarrow F$; go to (2)).

(8) $L \leftarrow \mathrm{INV}(\mathrm{PFL}(B, Q))$.

(9) Return.

Computing Time: $O(kn^3(\ln nf)^2)$, where $n = \deg(B)$, k is the highest multiplicity of any factor of B, $f_i = \mathrm{norm}(A_i)$, and $f = \prod_{i=1}^{k} f_i^i$.

Algorithm V = PVECT(A, k, n)

Input: $A \in I[x]$, k and n are Fortran integers, $k \geq 0$ and $\deg(A) + k < n$.

Output: If $A(x) \cdot x^k = \Sigma_{i=0}^{n-1} a_i x^i$ then $V = (a_{n-1}, \ldots, a_1, a_0)$. If $A = 0$, then V is the list of length n equal to $((), \ldots, ())$.

(1) $B \leftarrow A$; If $B \neq 0$, $B \leftarrow$ TAIL(B); $V \leftarrow 0$; $I \leftarrow n - 1$.

(2) $AI \leftarrow 0$; If $B = 0$, go to (3); $E \leftarrow$ FIRST(TAIL(B)) + k; if $E \neq I$, go to (3); ADV(AI, B); $B \leftarrow$ TAIL(B).

(3) $V \leftarrow$ PFL(BORROW(AI), V); $I \leftarrow I - 1$; if $I \geq 0$, go to (2).

(4) $V \leftarrow$ INV(V).

(5) Return.

Computing Time: O(n).

Algorithm L = RDEC(A/B)

Input: A non-zero regular rational function A/B, $\deg(B) > 0$, $\mathrm{ldcf}(B) > 0$, and $\gcd(A, B) = 1$.

Output: (L_1, L_2, L_3), where $L_1 = ((A_{1,1}), (A_{2,2}, A_{2,s_2}), \ldots, (A_{k,k}, \ldots, A_{k,s_k}))$, $L_2 = (B_1, \ldots, B_k)$, and $L_3 = (w_1, \ldots, w_k)$. $A_{i,j}$, $B_i \in I[x]$, $w_i \in I$ and $A/B = \sum_{i=1}^{k} (1/w_i) \sum_{j=s_i}^{i} A_{i,j}/B_i^j$ is a complete square-free partial fraction decomposition of A/B. If $B_i = 1$, then $s_i = i$, $A_{i,i} = 0$ and $w_i = 1$. Otherwise, $1 \le s_i \le i$, $w_i > 0$, and $\deg(A_{i,j}) < \deg(B_i)$ for $s_i \le j \le i$.

(1) $L1 \leftarrow L3 \leftarrow 0$.

(2) $Z \leftarrow$ RSQDEC(A/B), obtaining $M1 = (A_1, \ldots, A_k)$, $M2 = (B_1, \ldots, B_k)$, and $M3 = (v_1, \ldots, v_k)$ such that $A/B = \sum_{i=1}^{k} A_i/v_i B_i^i$; $L2 \leftarrow M2$.

(3) DECAP(AI, M1); ADV(BI, M2); DECAP(v, M3).

(4) If $AI = 0$ or $\deg(AI) < \deg(BI)$, ($L1 \leftarrow$ PFL(PFL(AI, 0), L1); $L3 \leftarrow$ PFL(v, L3); go to (7)).

(5) $Z \leftarrow$ PCDEC(AI, BI), obtaining $C = (A_{i,i}, \ldots, A_{i,s_i})$ and $d = \mathrm{ldcf}(BI)^{\deg(AI) - \deg(BI) + 1}$.

(6) $w \leftarrow d \cdot v$; $L1 \leftarrow$ PFL(C, L1); $L3 \leftarrow$ PFL(w, L3).

(7) If $M1 \neq 0$, go to (3).

(8) $L \leftarrow PFL(INV(L1), PFL(L2, PFL(INV(L3), 0)))$.

(9) Return.

Computing Time: $O(n^4(\ln nf)^2)$, where $n = \deg(B)$, $f_i = \text{norm}(B_i)$, $b = \text{cont}(B)$, and $f = \max\{\text{norm}(A), b\prod_{i=1}^{k} f_i^i\}$.

Algorithm L = RINTG(A/B)

Input: A non-zero regular rational function A/B, $\deg(B) > 0$, $\text{ldcf}(B) > 0$ and $\gcd(A, B) = 1$.

Output: L = (R, S) where R, S are regular rational functions, R is the rational part of $\int (A/B)$, and $\int (A/B) = R + \int S$.

(1) $b \leftarrow \text{cont}(B)$; $\overline{B} \leftarrow \text{pp}(B)$; $\overline{R} \leftarrow \overline{S} \leftarrow 0$; $U \leftarrow 1$.

(2) $Z \leftarrow PSQFRE(\overline{B})$; $k \leftarrow LENGTH(Z)$; $Z1 \leftarrow Z$.

(3) If $k = 1$, ($R \leftarrow 0$; $S \leftarrow A/B$; return).

(4) ADV(BI, Z1); $U \leftarrow U \cdot BI$; if $Z1 \neq 0$, go to (4).

(5) $V \leftarrow \overline{B}/U$; $i \leftarrow \deg(V) - 1$; $k \leftarrow \deg(U) - 1$.

(6) $E \leftarrow MATX(Z, U, V)$; $F \leftarrow PVECT(A, 0, \deg(\overline{B}))$.

(7) $X \leftarrow MUSSLE(E, F)$; DECAP(d, X); $w \leftarrow b \cdot d$.

(8) DECAP(P, X); if $P \neq 0$, $\overline{R} \leftarrow$ PFA(i, PFL(P, $\overline{R}$)); $i \leftarrow i - 1$.

(9) If $i \geq 0$, go to (8); if $\overline{R} \leftarrow$ PFL(PVBL($\overline{B}$), INV($\overline{R}$)).

(10) DECAP(P, X); if $P \neq 0$, $\overline{S} \leftarrow$ PFA(k, PFL(P, $\overline{S}$)); $k \leftarrow k - 1$.

(11) If $k \geq 0$, go to (10); If $\overline{S} \neq 0$, $\overline{S} \leftarrow$ PFL(PVBL($\overline{B}$), INV($\overline{S}$)).

(12) $R \leftarrow$ RPOLY2 ($\overline{R}$, w · V); $S \leftarrow$ RPOLY2($\overline{S}$, w · U); $L \leftarrow$ PFL(R, PFL(S, 0)).

(13) Return.

Computing Time: $O(n^5(\ln nf)^2)$, where $n = \deg(B)$, $b \prod_{i=1}^{k} B_i^i$ in the square-free factorization of B, $f_i = \text{norm}(B_i)$ and $f = \max\{\text{norm}(A), b \prod_{i=1}^{k} f_i^i\}$.

Algorithm L = RINTGS(R)

Input: A univariate rational function $R = A/B$, $\text{ldcf}(B) > 0$, and $\gcd(A, B) = 1$.

Output: If $R = 0$, $L = ((), ())$. Otherwise, $L = (S, T)$ where S and T are rational functions such that $\int R = S + \int T$ and S is the rational part of $\int R$.

(1) $L \leftarrow$ PFL(0, PFL(0,0)); If $R = 0$, return; ERASE(L); $\overline{R} \leftarrow R$.

(2) ADV(A, $\overline{R}$); ADV(B, $\overline{R}$); $m \leftarrow \deg(A)$; $n \leftarrow \deg(B)$.

(3) If $n = 0$, ($L \leftarrow$ PFL(PINTG(R), PFL(0, 0)); return).

(4) If $m < n$, ($L \leftarrow$ RINTG(R); return).

(5) $C \leftarrow \text{ldcf}(B)^{m-n+1}$; $E \leftarrow c \cdot A$; $F \leftarrow c \cdot B$.

(6) $Z \leftarrow$ PQREM(E, B) obtaining $Z = (X, Y)$ such that $E = BX + Y$.

(7) $P \leftarrow$ PFL(X, PFL(PFL(PVBL(B), PFL(C, PFA(0, 0))), 0)); $Q \leftarrow$ PFL(Y, PFL(F, 0)).

(8) $G \leftarrow$ PINTG(P); $T \leftarrow$ RINTG(Q).

(9) DECAP(H, T); $S \leftarrow$ RSUM(G, H).

(10) $L \leftarrow$ PFL(S, T).

(11) Return.

Computing Time: Let $m = \deg(A)$, $n = \deg(B)$, $B = b \prod_{i=1}^{k} B_i^i$ be the square-free factorization of B, $f_i = \text{norm}(B_i)$ and $f = \max\{\text{norm}(A), b \prod_{i=1}^{k} f_i^i\}$. Then, if $n = 0$, the computing time is $O(m^3(\ln mf)^2)$. If $m < n$, the computing time is $O(n^5(\ln nf)^2)$ and if $m \geq n$, the computing time is $O(m^3 n^2 (m-n+2)^2 (\ln mf)^2))$.

Algorithm RSQDEC(A/B)

Input: A non-zero, regular rational function A/B, where $\deg(B) > 0$, $\text{ldcf}(B) > 0$ and $\gcd(A, B) = 1$.

Output: $L = (X, Y, Z)$ where $X = (A_1, \ldots, A_k)$, $Y = (B_1, \ldots, B_k)$ $Z = (v_1, \ldots, v_k)$, A_i, $B_i \in I[x]$, $v_i \in I$, and $A/B = \Sigma_{i=1}^{k} A_i/v_i B_i^i$ 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) $b \leftarrow \text{cont}(B)$; $\overline{B} \leftarrow \text{pp}(B)$; $X \leftarrow Z \leftarrow 0$.

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

(3) If $k = 1$, ($L \leftarrow \text{PFL}(\text{PFL}(\text{BORROW}(A), 0), \text{PFL}(Y, \text{PFL}(\text{PFL}(b, 0), 0)))$; return).

(4) For $i = 1, \ldots, k-1$ do (ADV(BI, Y); if $\deg(BI) = 0$, go to (5)); $X \leftarrow \text{PFL}(\text{BORROW}(A), 0)$; $Z \leftarrow \text{PFL}(b, 0)$; for $i = 1, \ldots, k-1$ do ($X \leftarrow \text{PFL}(0, X)$; $Z \leftarrow \text{PFL}(\text{PFL}(1, 0), Z)$)); $L \leftarrow \text{PFL}(X, \text{PFL}(Y, \text{PFL}(Z, 0)))$.

(5) $E \leftarrow \text{MATSFD}(\overline{B}, Y)$; $F \leftarrow \text{PVECT}(A, 0, \text{PDEG}(\overline{B}))$.

(6) $G \leftarrow \text{MUSSLE}(E, F)$, obtaining $G = (g_0, g_1, \ldots, g_n)$.

(7) $\text{DECAP}(g_0, G)$; $w \leftarrow b \cdot g_0$.

(8) For $i = 1, \ldots, k$ do:

(8.1) ADV(BI, Y); $m \leftarrow \deg(BI)$; if $m = 0$, ($X \leftarrow \text{PFL}(0, X)$; $Z \leftarrow \text{PFL}(\text{PFA}(1, 0), Z)$; go to (8)).

(8.2) $n \leftarrow i \cdot m$; $AI \leftarrow 0$; for $j = 1, \ldots, n$ do (DECAP(C, G); if $C \neq 0$, $AI \leftarrow \text{PFA}(n-j, \text{PFL}(C, AI))$); $AI \leftarrow \text{PFL}(\text{PVBL}(A), \text{INV}(AI))$.

(8.3) $k \leftarrow \gcd(w, \text{cont}(AI))$.

(8.4) $v \leftarrow w/k$; $AI \leftarrow AI/k$; If $v < 0$, ($v \leftarrow -v$; $AI \leftarrow -AI$).

(8.5) $X \leftarrow \text{PFL}(AI, X)$; $Z \leftarrow \text{PFL}(v, Z)$.

(9) $X \leftarrow \text{INV}(X)$; $Z \leftarrow \text{INV}(Z)$; $L \leftarrow \text{PFL}(X, \text{PFL}(Y, \text{PFL}(Z, 0)))$.

(10) Return.

Computing Time: $O(kn^3(\ln nf)^2 + n^4(\ln nf))$, where $n = \deg(B)$, $b\prod_{i=1}^{k} B_i^i$ is the square-free factorization of B, $f_i = \text{norm}(B_i)$, and $f = \max\{\text{norm}(A), b\prod_{i=1}^{k} f_i^i\}$.

5. FORTRAN SUBPROGRAMS

```
      SUBROUTINE ADJCOL(V,M)
      INTEGER V,T,VI,FIRST,PFL,TAIL
      T=M
    2 CALL DECAP(VI,V)
      MI=FIRST(T)
      MI=PFL(VI,MI)
      CALL ALTER(MI,T)
      T=TAIL(I)
      IF (T.NE.0) GO TO 2
      RETURN
      END



      SUBROUTINE CUSSLE(P,L)
      INTEGER A,B,C,D,E,S,I,J,L,P,Q,R,S,T,U,V
      INTEGER CPROD,CRECIP,CSUM,CVPROD,FIRST,INV,PFA,PFL,TAIL
      A=L
      B=0
      G=1
      I=0
2     S=A
      A=0
      J=0
21    IF (S.EQ.0) GO TO 22
      CALL DECAP(T,S)
      CALL DECAP(U,T)
      IF (U.NE.0) GO TO 3
      J=J+1
      A=PFL(T,A)
      GO TO 21
22    IF (A.EQ.0) GO TO 4
      GO TO 5
3     B=PFL(T,B)
      IF (J.GT.0) I=I+1
      E=P-CRECIP(P,D)
      G=CPROD(P,G,D)
      U=T
31    CALL ALTER(CPROD(P,FIRST(U),E),U)
      U=TAIL(U)
      IF (U.NE.0) GO TO 31
32    IF (S.NE.0) GO TO 33
      A=INV(A)
      GO TO 2
33    CALL DECAP(Q,S)
      CALL DECAP(U,U)
      A=PFL(Q,A)
```

```
      IF (U.EQ.0) GO TO 32
      U=T
      R=Q
34    IF (U.EQ.0) GO TO 32
      CALL ALTER(CSUM(P,FIRST(R),CPROD(P,U,FIRST(U))),P)
      U=TAIL(U)
      R=TAIL(R)
      GO TO 34
4     IF ((PROD(2,I,1).NE.0) G=P-G
      V=PFA(G,0)
41    CALL DECAP(C,B)
      V=PFA(CVPROD(P,C,V),V)
      CALL ERLA(C)
      IF (B.NE.0) GO TO 41
      GO TO 6
5     CALL ERASE(A)
      CALL ERASE(B)
      V=-1
6     L=V
      RETURN
      END



      SUBROUTINE ELPOF2(X,EFLR,ECEIL)
      INTEGER ECEIL,EFLR,Q,R,RP,TWO,V,X,Z
      INTEGER IABSL,IQRS,PFA
      V=X
      EFLR=0
      ECEIL=0
      IF (V .EQ. 0) RETURN
      Q=IABSL(V)
      TWO=PFA(2,0)
      RP=0
1     Z=IQRS(Q,TWO)
      CALL ERLA(Q)
      CALL DECAP(Q,Z)
      CALL DECAP(R,Z)
      IF (R .EQ. 0) GO TO 2
      RP=RP+1
      CALL ERLA(R)
2     IF (Q .EQ. 0) GO TO 3
      EFLR=EFLR+1
      GO TO 1
3     IF (RP .GT. 1) ECEIL=1
      ECEIL=EFLR+ECEIL
      CALL ERLA(TWO)
      RETURN
      END
```

```
      INTEGER FUNCTION LCM(M)
      INTEGER A,L,B,BORROW
      L=M
    1 CALL ADV(A,L)
      IF (A.EQ.0) GO TO 1
      B=BORROW(A)
    3 IF (L.EQ.0) GO TO 6
      CALL ADV(A,L)
      IF (A.EQ.0) GO TO 3
      L1=IGCD(A,B)
      L2=IQ(B,L1)
      CALL ERLA(B)
      CALL ERLA(L1)
      B=IPROD(A,L2)
      CALL ERLA(L2)
      GO TO 3
    6 IF (ISIGNL(B).GE.0) GO TO 7
      L1=INEG(B)
      CALL ERLA(B)
      B=L1
    7 LCM=B
      RETURN
      END
```

```
      INTEGER FUNCTION MATSFD(B,F)
      INTEGER B,F,T,CI,FI,V,BI
      INTEGER PDEG,CINV,PVECT,PPOWER,PQ
      N=PDEG(B)
      K=LENGTH(F)
      T=CINV(F)
      M=PVECT(0,0,N)
      I=K
    2 CALL DECAP(BI,T)
      IF (PDEG(BI).EQ.0) GO TO 4
      CI=PPOWER(BI,I)
      KI=PDEG(CI)
      FI=PQ(B,CI)
      CALL PERASE(CI)
      J=0
    3 V=PVECT(FI,J,N)
      CALL ADJCOL(V,M)
      J=J+1
      IF (J.LT.KI) GO TO 3
      CALL PERASE(FI)
    4 I=I-1
      CALL PERASE(BI)
      IF (I.NE.0) GO TO 2
      MATSFD=M
      RETURN
      END
```

```
      INTEGER FUNCTION MAIX(F,U,V)
      INTEGER F,U,V,F1,W,BI,X,XI,BIP,XIB,WI,R
      INTEGER PDEG,PVECT,PFA,PQ,PDERIV,FIRST,PSUM,PNEG,PSPROD,PPROD
      LM=PDEG(V)
      N=PDEG(U)+LM
      M=PVECT(0,0,N)
      F1=F
      I=PFA(1,0)
      W=0
      CALL ADV(BI,F1)
    3 IF (F1.EQ.0) GO TO 4
      CALL ADV(BI,F1)
      X=PQ(U,BI)
      XI=PSPROD(X,I,0)
      CALL PERASE(X)
      BIP=PDERIV(BI,FIRST(BI))
      XIB=PPROD(BIP,XI)
      CALL PERASE(BIP)
      CALL PERASE(XI)
      WI=PSUM(W,XIB)
      CALL PERASE(XIB)
      CALL PERASE(W)
      W=WI
      CALL ALTER(FIRST(I)+1,I)
      GO TO 3
    4 CALL DECAP(X,I)
      WI=PNEG(W)
      CALL PERASE(W)
      J=0
    5 R=PVECT(V,J,N)
      CALL ADJCOL(R,M)
      J=J+1
      IF (J.LT.N-LM) GO TO 5
      R=PVECT(WI,0,N)
      CALL ADJCOL(R,M)
      J=0
      I=PFA(1,0)
      W=PSPROD(WI,I,1)
      CALL DECAP(X,I)
      CALL PERASE(WI)
      JJ=PFA(J+1,0)
    7 IF (J.GE.LM-1) GO TO 8
      IU=PSPROD(U,JJ,0)
      X=PSUM(IU,W)
      CALL PERASE(IU)
      R=PVECT(X,J,N)
      CALL PERASE(X)
      CALL ADJCOL(R,M)
      J=J+1
      CALL ALTER(J+1,JJ)
      GO TO 7
    8 CALL DECAP(I,JJ)
```

```
      CALL ERASE(W)
      MATX=M
      RETURN
      END



      INTEGER FUNCTION MUSSLE(A,B)
      COMMON /TR4/PRIME,PEXP
      INTEGER PRIME,PEXP
      INTEGER A,B,C,D,E,F,H,I,K,L,M,R,P,U,S,T,U,Z
      INTEGER CGARN,CMOD,FIRST,IABSL,ICMP,INV,IPROD,LENGTH
      INTEGER PFA,PFL,TAIL
      Q=PFA(1,0)
      N=LENGTH(A)+1
      C=0
      DO 1 I=1,N
1     C=PFL(0,C)
      F=PFA(2,0)
      Z=PRIME
      L=0
      U=B
      T=A
      I=PFA(1,0)
2     CALL ADV(E,U)
      E=IABSL(E)
      CALL ADV(S,T)
21    CALL ADV(D,S)
      D=IABSL(D)
      IF (ICOMP(D,E).EQ.1) GO TO 22
      CALL ERLA(D)
      GO TO 23
22    CALL ERLA(E)
      E=D
23    IF (S.NE.0) GO TO 21
      D=IPROD(F,E)
      CALL ERLA(F)
      CALL ERLA(E)
      F=IPROD(I,D)
      CALL ERLA(D)
      CALL ALTER(FIRST(I)+1,I)
      IF (U.NE.0) GO TO 2
      CALL ERLA(I)
      IF (F.EQ.0) GO TO 43
      CALL ELPOF2(F,S,T)
      K=T/PEXP+1
      CALL ERLA(F)
3     CALL ADV(P,Z)
      M=0
      U=B
      T=A
31    F=0
```

```
      CALL ADV(S,T)
32    CALL ADV(E,S)
      F=PFA(CMOD(P,F),F)
      IF (S.NE.0) GO TO 32
      CALL ADV(E,U)
      F=PFA(CDIF(P,U,CMOD(P,E)),F)
      M=PFL(INV(F),M)
      IF (U.NE.0) GO TO 31
      M=INV(M)
      CALL CUSSLE(P,M)
      IF (M.EQ.-1) GO TO 41
      L=L+1
      S=C
4     T=FIRST(S)
      CALL DECAP(U,M)
      CALL ALTER(CGARN(Q,T,P,U),S)
      CALL ERLA(T)
      S=TAIL(S)
      IF (S.NE.0) GO TO 4
      S=PFA(P,0)
      T=IPROD(Q,S)
      CALL ERLA(Q)
      CALL ERLA(S)
      Q=T
      IF (L.EQ.K) GO TO 5
      GO TO 42
41    H=H+1
      IF (H.EQ.K) GO TO 43
42    IF (Z.NE.0) GO TO 3
      CALL ERASE(C)
      C=-2
      GO TO 6
43    CALL ERASE(C)
      C=-1
      GO TO 6
5     S=INV(C)
      CALL DECAP(D,S)
      C=PFL(D,INV(S))
6     MUSSLE=C
      CALL ERLA(Q)
      RETURN
      END



      INTEGER FUNCTION PCDEC(A,B)
      INTEGER A,B,C,Q,Z,QB,AB,X
      INTEGER PDEG,PPOWER,FIRST,TAIL,PSPROD,PGREA,PFL
      X=0
      M=PDEG(A)
      N=PDEG(B)
      C=PPOWER(FIRST(TAIL(B)),M-N+1)
```

```
      Q=PSPROD(A,C,0)
    3 Z=PQREM(Q,B)
      CALL DECAP(QB,Z)
      CALL DECAP(AB,Z)
      X=PFL(AB,X)
      CALL PERASE(Q)
      IF (PDEG(QB).LT.N) GO TO 6
      Q=QB
      GO TO 3
    6 X=PFL(QB,X)
      PCDEC=PFL(INV(X),PFL(C,0))
      RETURN
      END



      INTEGER FUNCTION PINTO(RI)
      INTEGER R,A,B,AB,I,P,Z,Z1,Q,Q1,P1,RI
      INTEGER FIRST,PFL,BORROW,PFA,TAIL
      INTEGER PCONT,PONE,PSQ,PSPROD
      R=RI
      PINTO=0
      IF (R.EQ.0) RETURN
      CALL ADV(A,R)
      CALL ADV(B,R)
      AB=A
      CALL ADV(R,AB)
      T=0
      P=PFL(BORROW(R),0)
    4 CALL ADV(R,AB)
      CALL ADV(R,AB)
      T=PFL(PFA(R+1,0),T)
      IF (AB.NE.0) GO TO 4
      Z=LCM(T)
      T=INV(T)
      AB=TAIL(A)
    6 CALL ADV(R,AB)
      CALL DECAP(L,T)
      M=IQ(Z,L)
      N=IPROD(M,R)
      CALL ERLA(M)
      I=FIRST(L)
      CALL ERLA(L)
      P=PFA(I,PFL(N,P))
      CALL ADV(R,AB)
      IF (AB.NE.0) GO TO 6
      P=INV(P)
      Q=PSPROD(B,Z,0)
      CALL ERLA(Z)
      I1=PCONT(P)
      I2=PCONT(Q)
      I3=IGCD(I1,I2)
```

```
      CALL ERLA(I1)
      CALL ERLA(I2)
      IF (PONE(I3).EQ.1) GO TO 12
      P1=PSQ(P,I3)
      CALL PERASE(P)
      P=P1
      Q1=PSQ(Q,I3)
      CALL PERASE(Q)
      Q=Q1
   12 CALL ERLA(I3)
      PINTG=PFL(P,PFL(Q,0))
      RETURN
      END



      INTEGER FUNCTION PQREM(A,B)
      INTEGER A,B,R,BB,M,N,D,D1,RL,RP,S,A2
      INTEGER BORROW,PSQ,PDIF,PSPROD,PFA,PQ
      INTEGER PDEG,FIRST,TAIL,PFL,PLDCF,PRED
      R=BORROW(A)
      BB=B
      M=PDEG(R)
      N=PDEG(BB)
      IF (N.NE.0) GO TO 3
      L=FIRST(TAIL(BB))
      PQREM=PFL(PSQ(R,L),PFL(0,0))
      CALL PERASE(R)
      RETURN
    3 LB=PLDCF(BB)
      MB=PRED(BB)
      Q=PFL(BORROW(FIRST(R)),0)
      D=M-N
      D1=D+1
      DO 5 I=1,D1
      RL=PLDCF(R)
      RP=PRED(R)
      CALL PERASE(R)
      S=PQ(RL,LB)
      Q=PFA(D,PFL(S,Q))
      A2=PSPROD(MB,S,D)
      CALL PERASE(RL)
      R=PDIF(RP,A2)
      CALL PERASE(RP)
      CALL PERASE(A2)
      IF (R.EQ.0) GO TO 6
      D=PDEG(R)-N
      IF (D.LT.0) GO TO 6
    5 CONTINUE
    6 CALL PERASE(LB)
      CALL PERASE(MB)
      PQREM=PFL(INV(Q),PFL(R,0))
```

```
      RETURN
      END



      INTEGER FUNCTION PSQFRE(A)
      INTEGER Q,D,P,A,B,BP,E,F
      INTEGER PPOWER,BORROW,FIRST,PDERIV,PGCD1,PDEG,PQ,PFL
    1 I=C
      Q=0
      D=0
      P=PPOWER(A,0)
      B=BORROW(A)
    2 BP=PDERIV(B,FIRST(B))
      E=PGCD(B,BP)
      CALL PERASE(PP)
      IF (PDEG(E).EQ.0) GO TO 21
      F=PQ(B,E)
      GO TO 4
   21 F=BORROW(B)
    4 IF (I.EQ.0) GO TO 7
      IF (PDEG(D).EQ.PDEG(F)) GO TO 23
      Q=PFL(PQ(D,F),Q)
      GO TO 7
   23 Q=PFL(BORROW(P),Q)
    7 IF (PDEG(E).EQ.0) GO TO 8
      I=1
      CALL PERASE(D)
      CALL PERASE(B)
      B=E
      D=F
      GO TO 2
    8 PSQFRE=INV(PFL(P,Q))
      CALL PERASE(D)
      CALL PERASE(E)
      CALL PERASE(F)
      CALL PERASE(P)
      RETURN
      END



      INTEGER FUNCTION PVECI(A,K,N)
      INTEGER A,B,AI,E,V
      INTEGER TAIL,FIRST,PFL,BORROW
      B=A
      IF (B.NE.0) B=TAIL(B)
      V=0
      I=N-1
    2 AI=0
      IF (B.EQ.0) GO TO 3
      E=FIRST(TAIL(B))+K
```

```
      IF (E.NE.I) GO TO 3
      CALL ADV(AI,B)
      B=TAIL(B)
    3 V=PFL(BORROW(AI),V)
      I=I-1
      IF (I.GE.0) GO TO 2
      PVECT=INV(V)
      RETURN
      END



      INTEGER FUNCTION RDEC(R)
      INTEGER X,R,Z,AI,BI,VI,C,D,W
      INTEGER RSQDEC,PFL,PDEG,PCDEC
      X=R
      L1=0
      L3=0
      Z=RSQDEC(X)
      CALL DECAP(M1,Z)
      CALL DECAP(M2,Z)
      CALL DECAP(M3,Z)
      L2=M2
    3 CALL DECAP(AI,M1)
      CALL ADV(BI,M2)
      CALL DECAP(VI,M3)
      IF (AI.NE.0) GO TO 41
    4 L1=PFL(PFL(AI,0),L1)
      L3=PFL(VI,L3)
      GO TO 7
   41 IF (PDEG(AI).LT.PDEG(BI)) GO TO 4
      Z=PCDEC(AI,BI)
      CALL ERASE(AI)
      CALL DECAP(C,Z)
      CALL DECAP(D,Z)
      W=IPROD(VI,D)
      CALL ERLA(VI)
      CALL ERLA(D)
      L1=PFL(C,L1)
      L3=PFL(W,L3)
    7 IF (M1.NE.0) GO TO 3
      RDEC=PFL(INV(L1),PFL(L2,PFL(INV(L3),0)))
      RETURN
      END



      INTEGER FUNCTION RINTO(X)
      INTEGER Z1,BI,V,E,F,C,P,X,S,A,B,BB,R,RB,SB,U,Z
      INTEGER PFL,PPROD,PSPROD,RPOLY2,PR,PDEG,PVECT,TAIL,BORROW,PCPP
      INTEGER PPOWER,PSQRE,PFA,PVBL
      IY=X
```

```
    S=BORROW(IY)
    CALL ADV(A,IY)
    CALL ADV(B,IY)
    IB=PCPP(B)
    CALL DECAP(LB,IB)
    CALL DECAP(BB,IB)
    R=0
    RB=0
    SB=0
    U=PPOWER(A,0)
    Z=PSQFRE(BB)
    Z1=Z
    IF (TAIL(Z).NE.0) GO TO 41
  3 RINTG=PFL(R,PFL(S,0))
    CALL ERLA(LB)
    CALL PERASE(PB)
    CALL PERASE(U)
    CALL ERASE(Z)
    RETURN
 41 CALL RERASE(S)
  4 CALL ADV(BI,Z1)
    IU=PPROD(U,BI)
    CALL PERASE(U)
    U=IU
    IF (Z1.NE.0) GO TO 4
    V=PQ(BB,U)
    I=PDEG(V)-1
    K=PDEG(U)-1
    E=MATX(Z,U,V)
    F=PVECT(A,0,PDEG(BB))
    IX=MUSSLE(E,F)
    CALL ERASE(E)
    CALL ERASE(F)
    CALL DECAP(LD,IX)
    W=IPROD(LD,LP)
    CALL ERLA(LD)
  8 CALL DECAP(P,IX)
    IF (P.NE.0) RB=PFA(I,PFL(P,RB))
    I=I-1
    IF (I.GE.0) GO TO 8
    IF (RB.NE.0) RB=PFL(PVEL(BB),INV(RB))
 10 CALL DECAP(P,IX)
    IF (P.NE.0) SB=PFA(K,PFL(P,SB))
    K=K-1
    IF (K.GE.0) GO TO 10
    IF (SB.NE.0) SB=PFL(PVEL(BB),INV(SB))
    IV=PSPROD(V,W,0)
    CALL PERASE(V)
    IU=PSPROD(U,W,0)
    CALL ERLA(W)
    R=RPOLY2(RB,IV)
    CALL PERASE(RB)
```

```
      CALL PERASE(IV)
      S=RPOLY2(SB,IU)
      CALL PERASE(SB)
      CALL PERASE(IU)
      GO TO 3
      END



      INTEGER FUNCTION RINTGS(R)
      INTEGER R,RB,A,B,C,E,F,Z,X,Y,P,Q,G,T,H,S
      INTEGER PDEG,PFL,PINTG,RINTG,PPOWER,FIRST,TAIL,PSPROD
      INTEGER PQREM,PVBL,PFA,RSUM
      RINTGS=PFL(0,PFL(0,0))
      IF (R.EQ.0) RETURN
      CALL ERASE(RINTGS)
      RB=R
      CALL ADV(A,R+)
      CALL ADV(B,RB)
      M=PDEG(A)
      N=PDEG(B)
      IF (N.NE.0) GO TO 4
      RINTGS=PFL(PINTG(R),PFL(0,0))
      RETURN
    4 IF (M.GE.N) GO TO 5
      RINTGS=RINTG(R)
      RETURN
    5 C=PPOWER(FIRST(TAIL(B)),M-N+1)
      E=PSPROD(A,C,0)
      F=PSPROD(B,C,0)
      Z=PQREV(E,B)
      CALL DECAP(X,Z)
      CALL DECAP(Y,Z)
      CALL PERASE(F)
      P=PFL(X,PFL(PFL(PVBL(B),PFL(C,PFA(0,0))),C))
      Q=PFL(Y,PFL(F,0))
      G=PINTG(P)
      CALL RERASE(P)
      T=RINTG(Q)
      CALL RERASE(Q)
      CALL DECAP(H,T)
      S=RSUM(G,H)
      CALL RERASE(G)
      CALL RERASE(H)
      RINTGS=PFL(S,T)
      RETURN
      END
```

```
      INTEGER FUNCTION RSQDEC(R)
      INTEGER A,AI,B,BB,BI,B1,CF,E,F,D,VI,X,X,Z,Z1,R
      INTEGER BORROW,PCONT,PCPP,PDEG,PFA,PFL,PSQ,PSQFRE,PVBL,PVECT,PNEG
      X=R
      L1=0
      L3=0
      CALL ADV(A,X)
      CALL ADV(B,X)
      LB=PCPP(B)
      CALL DECAP(B1,LB)
      CALL DECAP(BB,LB)
      Z=PSQFRE(BB)
      K=LENGTH(Z)
      IF (K.NE.1) GO TO 1
      L1=PFL(BORROW(A),0)
      L3=PFL(B1,0)
    4 RSQDEC=PFL(L1,PFL(Z,PFL(L3,0)))
      CALL ERASE(BB)
      RETURN
    1 KM1=K-1
      Z1=Z
      DO 2 I=1,KM1
      CALL ADV(BI,Z1)
      IF (PDEG(BI).NE.0) GO TO 10
    2 CONTINUE
      L1=PFL(BORROW(A),L1)
      L3=PFL(B1,L3)
      DO 3 I=1,KM1
      L1=PFL(0,L1)
    3 L3=PFL(PFA(1,0),L3)
      GO TO 4
   10 E=MATSFD(BB,Z)
      F=PVECT(A,0,PDEG(BB))
      X=MUSSLE(E,F)
      CALL ERASE(E)
      CALL ERASE(F)
      IF (X.EQ.-1.OR.X.EQ.-2) GO TO 7
      CALL DECAP(D,X)
      W=IPROD(D,BI)
      CALL ERLA(B1)
      CALL ERLA(D)
      Z1=Z
      DO 5 I=1,K
      CALL ADV(BI,Z1)
      NI=PDEG(BI)
      IF (NI.NE.0) GO TO 6
   12 L1=PFL(0,L1)
      L3=PFL(PFA(1,0),L3)
      GO TO 5
    6 INI=I*NI
      AI=0
      DO 8 J=1,INI
```

```
      CALL DECAP(CF,X)
      IF (CF.EQ.0) GO TO 8
      AI=PFA(1NI-J,PFL(CF,AI))
    8 CONTINUE
      IF (AI.EQ.0) GO TO 12
      AI=PFL(PVBL(A),INV(AI))
      LA=PCONT(AI)
      KK=IGCD(LA,W)
      CALL ERLA(LA)
      VI=IQ(W,KK)
      IAI=PSQ(AI,KK)
      CALL PERASE(AI)
      AI=IAI
      CALL ERLA(KK)
      IF (ISIGNL(VI).NE.-1) GO TO 11
      IVI=INEG(VI)
      CALL ERLA(VI)
      VI=IVI
      IVI=PNEG(AI)
      CALL PERASE(AI)
      AI=IVI
   11 L1=PFL(AI,L1)
      L3=PFL(VI,L3)
    5 CONTINUE
      L1=INV(L1)
      L3=INV(L3)
      CALL ERLA(W)
      GO TO 4
    7 PRINT 9,X
    9 FORMAT(3H X=,I3)
      STOP
      END
```

6. REFERENCES

1. American Standards Association. A Programming Language for Information Processing on Automatic Data Processing Systems. C.A.C.M., Volume 7, No. 10 (October, 1964), pp. 591-625.

2. Collins, G. E. Algebraic Algorithms, Prentice-Hall (to be published in 1971).

3. Collins, G. E. The SAC-1 List Processing System. University of Wisconsin Computing Center Report, September, 1967.

4. Collins, G. E. and Pinkert, James R. The Revised SAC-1 Integer Arithmetic System. University of Wisconsin Computing Center, Technical Reference No. 9, November, 1968.

5. Collins, G. E. The SAC-1 Polynomial System, University of Wisconsin Computing Center, Technical Reference No. 2, January, 1968.

6. Collins, G. E. The SAC-1 Rational Function System. University of Wisconsin Computing Center, Technical Reference No. 8, July, 1968.

7. Collins, G. E., et. al. The SAC-1 Modular Arithmetic System. University of Wisconsin Computing Center, Technical Reference No. 10, June, 1969.

8. Horowitz, E. Algorithms for Symbolic Integration of Rational Functions, Ph.D. Thesis, University of Wisconsin, November, 1969.