IMPLEMENTATION IN ALTRAN FOR
RATIONAL FUNCTION INTEGRATION AND
POLYNOMIAL FACTORIZATION
IMPLEMENTATION IN ALTRAN FOR
RATIONAL FUNCTION INTEGRATION AND
POLYNOMIAL FACTORIZATION

by

MAHMOUD MAGDY MAKHLOUF, B.Sc.

A Project
Submitted to the School of Graduate Studies
in Partial Fulfilment of the Requirements
for the Degree
Master of Science

McMaster University
April 1975
MASTER OF SCIENCE (1975)  
(Computation)  
McMASTER UNIVERSITY  
Hamilton, Ontario  
Canada  

TITLE:  
Implementation in Altran for Rational Function Integration and Polynomial Factorization  

AUTHOR:  
Mahmoud Magdy Makhlouf  
B.Sc. Electrical Engineering  
Alexandria University  
Alexandria, Egypt  

SUPERVISOR:  
Dr. R.A. Rink  

NUMBER OF PAGES: vii, 185
ABSTRACT

This project is a study involving the application of the ALTRAN system to rational function integration. A discussion and the implementation of two methods are given, one by Hermite [HER 12] and a second by Horowitz [HOR 70]. Included is a brief discussion of the integration of the transcendental part over the rational field using polynomial factorization over the integers. Furthermore, an extension for multivariate rational function integration and multivariate polynomial factorization is included.
ACKNOWLEDGEMENTS

I wish to express my gratitude to Professor R.A. Rink for his assistance, comments and suggestions during the course of this work and his aid in the preparation of this manuscript.

An informal acknowledgement is due to P.S. Wang and L.P. Rothschild. A considerable part of this project is based on their work.

Also I would like to thank Mrs. Jean Salamy for her patient and careful typing of this project.

My thanks go to the friends I have made who made my stay at McMaster enjoyable.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER 1:</th>
<th>INTRODUCTION</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Brief History of Symbolic Integration by Computer</td>
<td>2</td>
</tr>
<tr>
<td>1.3</td>
<td>Purpose of this Project</td>
<td>4</td>
</tr>
<tr>
<td>1.4</td>
<td>Outline of Further Chapters</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHAPTER 2:</th>
<th>INTEGRATION OF RATIONAL FUNCTIONS</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Introduction to ALTRAN</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>Definition and Theorems</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>Hermite's Method for Rational Function Integration</td>
<td>17</td>
</tr>
<tr>
<td>2.4</td>
<td>Horowitz's Method for Rational Function Integration</td>
<td>32</td>
</tr>
<tr>
<td>2.5</td>
<td>Discussion on the Methods and Empirical Results</td>
<td>41</td>
</tr>
<tr>
<td>2.6</td>
<td>Extension of Rational Function Integration to Multivariate Rational Functions</td>
<td>42</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CHAPTER 3:</th>
<th>POLYNOMIAL FACTORIZATION OVER INTEGERS</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Introduction to Factoring Problem</td>
<td>49</td>
</tr>
<tr>
<td>3.2</td>
<td>Factoring Polynomials over Finite Fields</td>
<td>51</td>
</tr>
<tr>
<td>3.3</td>
<td>Zassenhaus Algorithm using Hensel's Lemma</td>
<td>54</td>
</tr>
<tr>
<td>3.4</td>
<td>Factoring a Univariate Polynomial over the Integer</td>
<td>56</td>
</tr>
<tr>
<td>3.5</td>
<td>Multivariate Polynomial Factorization over the Integers</td>
<td>58</td>
</tr>
<tr>
<td>3.6</td>
<td>Implementation of Wang's Algorithm for Factoring Multivariate Polynomials</td>
<td>65</td>
</tr>
</tbody>
</table>
CHAPTER 4: INTEGRATION OF TRANSCENDENTAL PART 91

4.1 Introduction to the Basic Problem of Integrating the Transcendental Part 91
4.2 Algebraic Extension Field $K$ of $F$ 93
4.3 Nature of the Problem Solved in Altran 95
4.4 Implementation 97
4.5 Conclusions 104

REFERENCES 105

APPENDIX A: A Listing of the Program HERM 108
APPENDIX B: A Listing of the Program RINTGS 128
APPENDIX C: A Listing of the Program MVFOIT 141
APPENDIX D: A Listing of the Program INTRPT 179
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Hermite algorithm</td>
<td>18</td>
</tr>
<tr>
<td>2.2</td>
<td>Coefficient matrix for Hermite algorithm</td>
<td>22</td>
</tr>
<tr>
<td>2.3</td>
<td>Coefficient matrix for Horowitz algorithm</td>
<td>36</td>
</tr>
<tr>
<td>2.1</td>
<td>Horowitz algorithm</td>
<td>38</td>
</tr>
<tr>
<td>2.5</td>
<td>Example on Univariate rational function integration</td>
<td>45</td>
</tr>
<tr>
<td>2.6</td>
<td>Example on Multivariate rational function integration using Horowitz algorithm</td>
<td>46</td>
</tr>
<tr>
<td>3.1</td>
<td>Multivariate polynomial factorization algorithm</td>
<td>48</td>
</tr>
<tr>
<td>3.2</td>
<td>Examples on Multivariate polynomial factorization</td>
<td>80</td>
</tr>
<tr>
<td>4.1</td>
<td>Example on transcendental function Integration</td>
<td>89</td>
</tr>
<tr>
<td></td>
<td></td>
<td>99</td>
</tr>
<tr>
<td></td>
<td></td>
<td>103</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION TO SYMBOLIC ALGEBRAIC MANIPULATION

1.1 Introduction

While much has been accomplished in the way of solving mathematical problems using numerical techniques on digital computers, many of these techniques fail to give exact solutions in terms of closed forms. To obtain solutions in terms of closed forms, analytical techniques must be employed which not only are well structured, but are carefully defined to enable one to perform operations on mathematical expressions without concern to their numeric value.

The application of analytical techniques on a digital computer is called formal symbolic computation and can include symbolic integration, symbolic differentiation, solutions of simultaneous equations, power series manipulation, polynomial factorization as well as substitution and simplification of expressions.

Before the last decade using a digital computer to perform formal symbolic manipulation was a tedious task due to the slow speed of the machines, their small storage capacity and the demand of having to program in machine language. One of the earliest examples was a program for performing symbolic differentiation written by Nolan [NOL 53]
using a Whirlwind machine. In the later part of the last
decade systems such as Alpak [BRØ 63], Formac [TOB 67,a],
SACL [CØL 71], MATHLAB [ENG 65] and REDUCE [HEA 67] became
available for performing formal symbolic computation, while
some of these earlier systems were designed to manipulate
polynomials in several variables. The ALTRAN [BRØ 73],
MATHLAB and REDUCE2 [HEA 70] systems provide the user with
the capability of manipulating rational functions in several
variables. These later programming systems have offered a
powerful set of logic, passing and testing functions. Many
of these systems include the capabilities to perform pattern
matching, symbolic to numeric conversion as well as
constructing recursive procedures.

This project is primarily concerned with the formal
symbolic integration of rational functions in several
variables, including symbolic factorization of multivariate
polynomials. Programs written in ALTRAN system to perform
these exercises will be demonstrated.

1.2 Brief History of Symbolic Integration by Computer

The first investigation into symbolic integration by
a digital computer came from the area of Artificial
Intelligence in the work of Slagle's SAINT [SLA 61]. In
SAINT a pattern matching routine is applied to determine
the proper transformation needed to obtain results from
tabulated formulas. Three years after Slagle's SAINT,
Manove using the MATHLAB system [MAN 68] developed a rational function integration program. Manove's implementation relies upon the method of Hermite [HER 12]; a method that has attracted considerable interest during the last decade. Unfortunately, Manove's program has difficulty when factoring the denominator of rational functions. A third system for performing formal symbolic integration using a digital computer was developed by Moses [MOS 67]. Called SIN, Moses was able to develop a more superior and faster algorithm than SAINT using a more sophisticated pattern recognition program for finding the optimal method to perform integration. Much of the pattern recognition program depends upon decision procedures such that of the method chosen and applied to the integrand, the exact results will easily be obtained.

The integration of rational functions in SIN makes used of the method of Hermite.

Tobey in his Ph.D. thesis [TÖB 67, b] concentrated on the formal symbolic integration of rational functions. He has given a complete discussion and analysis of the problem including an algorithm for performing the integration using Hermite's method. Included in his discussion is an analysis on performing efficiently the greatest common divisor calculation using the Euclidean algorithm, as well as partial fraction decomposition. Algorithms for perform-
ing these functions are also discussed.

In the beginning of this decade Horowitz [HØR 70] using the SAC1 system performed a complete analysis on rational function integration by applying modular arithmetic to Hermite's method. In addition, Horowitz developed a new and more efficient method for finding the rational part of the integral of a rational function. This method involves the solution of a system of linear equations which are easier to obtain over that of partial fraction decomposition. Horowitz left the transcendental part unfactorized.

Tobey [TØB 67,b] discussed a numerical technique for obtaining the transcendental portion of a rational integral. His method involved approximating the roots of the denominator of the transcendental part numerically while continuing to use a symbolic approach. Tobey also discussed the need for faster polynomial factorization algorithms.

Since Tobey's thesis, Musser [MUS 71] and Wang [WAN 73] have developed more efficient polynomial factorization algorithms using modular arithmetic. Much of what these people have accomplished has been implemented in this project to factorize multivariate polynomials of the transcendental part of a rational integral.

1.3 Purpose of this Project

The purpose of this project is to implement both
the Hermite and Horowitz methods for rational function integration in the ALTRAN system. While the ALTRAN system is a rational function system, our interest here is to extend the capability of ALTRAN to perform the integration of rational functions. In performing this exercise several algorithms have been implemented in ALTRAN to perform polynomial square free factorization, complete partial fraction decomposition and the solution of linear simultaneous equations. In addition, an extension of Horowitz's algorithm to perform the integration of multivariate rational functions is discussed and implemented using ALTRAN.

In continuing the study for integrating the transcendental part, the polynomial factorization algorithm of Wang has been implemented using the modular arithmetic capability of ALTRAN.

The project is concluded by using an algorithm to integrate the transcendental part employing factorization of the denominator, partial fraction decomposition, while using a simple pattern matching program. However the integration of the transcendental part is not complete in some cases, since it requires computation over irrational and complex fields which are at present beyond the capabilities of the ALTRAN system.
1.4 Outline of Further Chapters

In Chapter 2 we will briefly discuss the ALTRAN system, listing some of its capabilities, specifically those used in implementing some of the algorithms discussed in later chapters. Included in Chapter 2 is a discussion of Hermite's and Horowitz's method as well as a description of their implementation in ALTRAN.

In Chapter 3 a discussion of Wang's algorithm for multivariate polynomial factorization is described including its implementation in the ALTRAN system.

In the last chapter a discussion of the integration of the transcendental part along with a description of its implementation in ALTRAN is given. Program listings and results have been included in the appendix.
CHAPTER 2
INTEGRATION OF RATIONAL FUNCTIONS

In this chapter we will discuss ALTRAN and its application for symbolically computing the integrals of rational functions.

2.1 Introduction to ALTRAN

ALTRAN, short for algebraic translator, is both a language and a system for performing formal algebraic computations on algebraic data. Basically it is capable of performing rational operations on rational expressions in one or more variables with integer coefficients.

The ALTRAN system is composed of a translator, interpreter and run time library and has been written almost entirely in FORTRAN IV. Considerable effort was made to achieve a portable system without sacrificing efficiency. To avoid machine limitations, both macros and primitive subroutines are used. Macros permit extensions of the implementation language while primitives allow for the efficient coding of critical operations.

As a programming language ALTRAN supports the elementary arithmetic operations (+, -, *, /, **) while more complicated operations such as symbolic differentiation and greatest common divisor are provided through procedure
calls to library routines.

Syntax and semantics of ALTRAN have been based on that of FORTRAN and PL/I, but with the extensions of new data types. Data types in ALTRAN include LABEL, LOGICAL, INTEGER, RATIONAL, REAL and ALGEBRAIC. ALGEBRAIC is an attribute for declaring rational functions. These last four attributes can also be associated with precision attribute SHORT or LONG, a storage class attribute AUTOMATIC or STATIC and a scope attribute INTERNAL or EXTERNAL. Default attributes are SHORT, AUTOMATIC and INTERNAL. A parenthesized list associated with the ALGEBRAIC attribute is called a layout and serves to declare the maximum exponent associated with the determinates (independent variables of rational functions). For example, LONG ALGEBRAIC (x:20,y:30) A,B declares A and B to be internal automatic ALGEBRAIC's with long integer coefficients. The maximum exponent for x and y are 20 and 30 respectively.

Arrays for all data types can be declared using the array attribute. For example, the declarations

RATIONAL ARRAY(5,6) A
ALGEBRAIC (x:20,y:30) ARRAY (2,3) B

declares A to be a 5*6 array of rational numbers and B to be a 2*3 array of ALGEBRAIC in the indeterminates X,Y.

There are four classes of operators in ALTRAN, these include arithmetic, relational, logical and special. Special
operators include dollar "$", used for multiple assignments, colon ":" used in the layouts, equal "=" for assignment, and comma "," for representing lists.

Expressions in ALTRAN are written by combining constants, variable, array elements, function calls and algebraic references with the arithmetic operators. An algebraic reference, while similar to a function call, denotes a value obtained by substitution rather than by execution of a function. For example, if A is ALGEBRAIC in the variable X and Y, then the expression

\[ A(5**3,T) \]

would result in the simultaneous substitution of \(5^3\) and T for X and Y throughout the expression of A.

ALTRAN also supports assignment statements which are similar in appearance to those of FORTRAN and PL/I. In addition, there are a modest number of control statements which include Do group, labels and jumps, if groups, etc. Input and output are handled by the functions READ and WRITE. Input is in a free-format while output is in a standard format that is input compatible.

An ALTRAN program consists of a collection of one or more procedures each beginning with a procedure declaration and ending with an END statement. A procedure may be a subroutine or a function depending on whether or not it returns a value using the RETURN statement. Only the first procedure, PROCEDURE MAIN has no RETURN statements.
The ALTRAN system also has a variety of library procedures for numerical and symbolic manipulation. These include procedures for numerical analysis, testing and conversion of numerical values, algebraic analysis, algebraic computation, modular reduction, array operations and matrix computation, truncated power series computation and input-output. A more extensive discussion, including examples can be found in the ALTRAN user's manual [BRØ 73].

2.2 Definitions and Theorems

The purpose of this section is to introduce some of the basic definitions and theorems needed in the analysis of Hermite's and Horowitz's algorithms. Since more formal proof to each of the theorems can be found in the literature, only a brief discussion is given for each proof.

2.2D1 A rational function $R(x)$ is defined as a numerator - denominator pair of polynomials $A(x)/B(x)$, where $A(x)$ and $B(x)$ have integer coefficients, are relatively prime and where the leading coefficients of $B(x)$ is positive.

2.2D2 A rational function $R(x) = A(x)/B(x)$ is called regular if the degree of the numerator $A(x)$ is less than the degree of the denominator $B(x)$.

2.2D3 A polynomial $B(x)$ of positive degree over an integer domain $I$ is said to be irreducible over $I$ if it can-
not be expressed as the product of two polynomials of positive degree over I.

2.2T1 If B(x) is a polynomial of positive degree over field F and if "a" is its leading coefficients, then there exist distinct, monic, irreducible polynomials, B_1(x), B_2(x), ..., B_k(x) over F such that

\[ B(x) = a * B_1(x)^{n_1} * B_2(x)^{n_2} * ... * B_k(x)^{n_k} \]

where \( n_i \) are positive integers, \( i = 1, 2, ..., k \), the degree of \( (B_i) > 0 \) and where the degree(B) = \( \sum_{i=1}^{k} (n_i \cdot \text{degree} (B_i)) \),

this factorization being unique except for order [HOR 70]. The proof to this theorem can be given by proving the theorem of uniqueness of prime factorization in principal ideal rings [VAN 53].

2.2D4 A polynomial B(x) of positive degree 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. Thus a polynomial which is square free has only roots of multiplicity 1.

2.2D5 Suppose B(x) = a * B_1(x)^{l_1} * B_2(x)^{l_2} * ... * B_k(x)^{l_k} where a \( \in I \), B_i is primitive and has a positive leading coefficient for \( 1 \leq i \leq k \). In addition a \( \in I \) and deg \( (B_i(x)) > 0 \) and all B_i's are pairwise relatively
prime. Then a $\prod_{i=1}^{k} B_i(x)$ is called the square free factorization of $B(x)$.

2.2T2 If $B_1(x)$ and $B_2(x)$ are two relatively prime polynomials over a field $F$, $m = \deg(B_1)$, $n = \deg(B_2)$, $m,n>0$ and if $A(x)$ is an arbitrary polynomial of degree less than $m+n$, then there exists an identity

$$A(x) = C(x)*B_1(x) + D(x)*B_2(x),$$

where $\deg(C(x))<n$, $\deg(D(x))<m$, $C(x),D(x) \in \mathbb{I}[x]$.

[Hor 70]

Proof follows that of [Wan 53, pp. 88].

By hypothesis, the greatest common divisor of $B_1(x), B_2(x)$ is equal 1. Then the following identity holds:

$$R(x)*B_1(x) + S(x)*B_2(x) = 1$$

Multiplying both sides by $A(x)$ gives

$$A(x) = (R(x)*A(x))*B_1(x) + (S(x)*A(x))*B_2(x) \quad (2.1)$$

To reduce the degree of $(R(x)*A(x))$ to a value less than $n$ we divide this polynomial by $B_2(x)$:

$$R(x)*A(x) = G(x)*B_2(x) + C(x) \quad (2.2)$$

where $\deg(C(x))<n$.

Substituting this into equation (2.1) gives:
2.2T3 Let $A(x)/B(x)$ be a regular rational function, whose denominator $B(x)$ can be resolved into powers of prime polynomials $B_1(x)^{n_1}, B_2(x)^{n_2}, \ldots, B_k(x)^{n_k}$, i.e., $B(x) = \prod_{i=1}^{k} B_i(x)^{n_i}$.

This rational function can then be represented as a sum of partial fractions whose denominators are powers of prime polynomials into which the denominator $B(x)$ resolves. This summation called the partial fraction decomposition of a rational function is given by

$$A(x)/B(x) = \sum_{i=1}^{k} A_i(x)/B_i(x)^{n_i},$$

where

$$\text{Deg } A_i(x) < \text{Deg } B_i(x)^{n_i} \text{ or } A_i(x) = 0 \text{ if } \text{Deg } B_i(x) = 0$$

[HOR 70]

For the proof let $k = 2$, such that $B(x) = B_1(x)^{n_1} \cdot B_2(x)^{n_2}$.
Using 2.2T2 we can write
\[ A(x) = C(x)B^1_1(x) + D(x)B^1_2(x) \]
Dividing both sides by \( B(x) \) we obtain two partial fraction terms
\[ \frac{A(x)}{B(x)} = \frac{D(x)}{B^1_1(x)} + \frac{C(x)}{B^1_2(x)} \]
where
\[ \text{Deg } D(x) < \text{Deg } B^1_1(x), \text{ Deg } C(x) < \text{Deg } B^1_2(x) \]

By induction we can prove the theorem for \( K > 2 \)

2.2T4 The partial fraction decomposition of a rational function is unique. [HOR 70]

2.2T5 Given a regular rational function \( A(x)/B(x) \) whose denominator has the factorization
\[ B(x) = b \prod_{i=1}^{k} B_i(x), \text{ where the } B_i(x) \text{ are pairwise relatively prime polynomials, there exist polynomials } A_{i,j}(x) \text{ for } 1 \leq j \leq n_i, 1 \leq i \leq k, \text{ such that the rational function } A(x)/B(x) \text{ can be represented as} \]
\[ \frac{A(x)}{B(x)} = \sum_{i=1}^{k} \sum_{j=1}^{n_i} \frac{A_{i,j}(x)}{B_i(x)^j}, \text{ where} \]
\[ \text{Deg } A_{i,j}(x) < \text{Deg } B_i(x) \quad [\text{HOR 70}] \]

This summation is referred to as the complete partial fraction decomposition. From 2.2T3, we can write rational function as:
\[
A(x)/B(x) = \sum_{i=1}^{k} \frac{A_i(x)}{B_i(x)}^{n_i} \tag{2.3}
\]

Using the remainder theorem we write

\[A_i(x) = S_i(x) B_i(x)^{n_i-1} + r_i(x)\]

\[r_i(x) = S_2(x) B_i(x)^{n_i-2} + r_2(x)\]

\[
\vdots
\]

\[r_{n_i-1}(x) = S_{n_i}(x)\]

Thus \[A_i(x) = S_i(x)B_i(x)^{n_i-1} + S_2(x)B_i(x)^{n_i-2} + \ldots + S_{n_i}(x)\]

Dividing both sides by \(B_i(x)^{n_i}\)

\[
\frac{A_i(x)}{B_i(x)^{n_i}} = \frac{S_i(x)}{B_i(x)} + \frac{S_2(x)}{B_i(x)^2} + \ldots + \frac{S_{n_i}(x)}{B_i(x)^{n_i}}
\]

and setting

\[A_{i,j}(x) = S_j \quad j = 1, \ldots, n_i\]

\[
\frac{A_i(x)}{B_i(x)^{n_i}} = \sum_{j=1}^{n_i} A_{i,j}(x)/B_i(x)^j
\]

Substituting this last summation into equation \(2.3\) for \(i = 2, 3, \ldots, k\), we obtain

\[
A(x)/B(x) = \sum_{i=1}^{k} \sum_{j=1}^{n_i} a_{i,j}(x)/B_i^j(x) \tag{2.4}
\]
2.2T6 A complete square free partial fraction decomposition of a regular rational function is unique. [HOR 70]

2.2T7 Let \( R(x) = \frac{A(x)}{B(x)} \) be a regular rational function then

\[
\int R(x) \, dx = S(x) + \sum_{i=1}^{k} d_i \log(x-b_i)
\]  

(2.5)

where \( S(x) \) is a regular rational function and

\[
\sum_{i=1}^{n} d_i \log(x-b_i)
\]

is the transcendental part of integration, \( b_i \) are in complex number field \( \mathbb{C} \) and are distinct roots of \( B(x) \) where \( d_i \in \mathbb{C} \) for \( i = 1, 2, \ldots, k \)

[HAR 16]

For the proof let us write \( B(x) \) as

\[
B(x) = a_1(x-b_1)^{n_1} \ast (x-b_2)^{n_2} \ast \cdots \ast (x-b_k)^{n_k}
\]  

(2.6)

where \( b_i \in \mathbb{C} \)

Using theorem 2.2T7 we can write

\[
R(x) = \frac{A(x)}{B(x)} = \sum_{i=1}^{k} \frac{A_{i,1}(x)}{(x-b_i)} + \frac{A_{i,2}(x)}{(x-b_i)^2}
\]

\[+ \cdots + \frac{A_{i,n_i}(x)}{(x-b_i)^{n_i}}\]

where
\[
\int R(x) \, dx = \sum_{i=1}^{k} \left( \frac{A_{i,1}(x)}{x-b_i} \right) + \sum_{i=1}^{k} \left[ - \frac{A_{i,2}}{(x-b_i)} \right] \\
+ \frac{A_{i,3}}{(x-b_i)^2} \cdots - \frac{A_{i,n_i}(x)}{(n_i-1)(x-b_i)^{n_i-1}}
\]

\[
= \sum_{i=1}^{k} A_{i,1} \log(x-b_i) + S(x)
\]

where \( S(x) \) is a rational function and \( A_{i,1} = d_i \).

2.2T8 If \( R(x) \) is a rational function, the the rational and transcendental parts of \( \int R(x) \, dx \) are unique.

[HOR 70]

2.3 Hermite's Method for Rational Function Integration

Hermite's method [HER 12] for the integration of rational functions can be divided into two parts. In the first part we obtain the complete square free partial fraction decomposition, while in the second part we obtain the rational part of integration using a reduction method. A general algorithm describing Hermite's method is given in Figure 2.1.

In performing the complete square free partial fraction decomposition, we make use of the algorithm RSQDEC to obtain a square free partial fraction decomposition. During the execution of RSQDEC we compute the square free factorization of the denominator using the algorithm PSQFRE
READ AB

If AB non regular put it in this form
$AB = AB^* + R_1$, where $AB^* = A/B$ is
regular rational function and degree
of $R_1 < \text{degree of } B$

Set $R=0$, $S=0$

Factorization algorithm such that
$B = \prod_{i=1}^{k} y_i(x), y_i$ is square
$\text{free polynomial}$

Construct matrix $E$ for performing
partial fraction decomposition

Solve system of linear equation to
 obtain $A_i$ such that
$A/B = \sum_{i=1}^{k} A_i(x)/y_i(x)$

Compute $A_i, j(x)$ such that
$A_i(x)/y_i(x) = \sum_{j=1}^{k} \frac{A_i, j}{w_i y_i(x)}$

$S = A_{I, 1}/(y_1(x)^a), I = 2$

$I = I + 1$

Compute $R_p, S_p$ such that
$\int \sum_{j=1}^{I} A_i, j y_i^j = R_p + \int S_p$

$R = R + R_p/a*w_j, S = S + S_p/a*w_j$

Yes

$I < K$

$R = R + \int R_1$

$S = S$

Figure 2.1 Hermite Algorithm
followed by computing the partial fraction terms using the algorithm MATSFD. Once we have completed these steps, we then proceed to compute the complete partial fraction decomposition using the algorithm PCDEC.

A brief description of these algorithms follows:

Algorithm PSQFRE:-

Input is any polynomial \( B(x) \) while the output is the square free polynomials \( Q_1, Q_2, \ldots, Q_k \) represented by a vector such that

\[
B(x) = Q_1(x)^{n_1} \cdot Q_2(x)^{n_2} \cdots \cdot Q_k(x)^{n_k},
\]

where

\[
n_k > n_{k-1} > \ldots > n_2 > n_1
\]

1) Initialize: set \( Q=0, D=0 \)

2) Obtain the linear term:

set \( E = \text{GCD}(B, dB/dx) \)

If \( E = 0 \) then set \( F = B; \) else \( F = B/E \)

3) Add to the vector \( Q: \)

if \( \text{deg}(D) = \text{deg}(F) \) go to 4), if \( Q \neq 0 \) add \( D/F \) to \( Q \)

4) Test for an end to the algorithm:

if \( E \) is an integer add \( B \) to the vector \( Q \), then end; else set \( B = E, D = F \) and return to step 2).

Let \( n = \text{Deg} (B(x)), n_i = \text{Deg} (B_i(x)) \) such that
\[ B(x) = \prod_{i=1}^{k} B_i(x)^i \]

Our purpose is to obtain \( A_i(x) \) which satisfies theorem 2.2T3 such that

\[ \frac{A(x)}{B(x)} = \sum_{i=1}^{k} \frac{A_i}{B_i(x)}^i \]

This equation can be rewritten by multiplying both sides by \( B(x) \) such that

\[ A(x) = A_1 E_1 + A_2 E_2 + \ldots + A_k E_k \quad (2.7) \]

where

\[ A_i(x) = \sum_{j=0}^{\text{in}_i-1} a_{i,j} x^j \]

\[ E_i(x) = \sum_{j=0}^{n-\text{in}_i} e_{i,j} x^j \]

where \( a_{i,j}, e_{i,j} \in I \).

To compute \( A_i \) we must compute \( a_{i,j} \). This can be obtained by equating the coefficients for the same powers in \( x \) in both sides of equation (2.7).

Before this can be done, the procedure MATSFD constructs a matrix \( E \) composed of the coefficients \( e_{i,j} \) as follows:
Algorithm MATSFD: -

Inputs to this procedure are both $B(x)$ and the resolvers list $(B_1,B_2,\ldots,B_k)$. Output is matrix $E$ shown in Figure 2.2. Matrix $E$ will be employed to compute the partial fraction terms of any rational function.

1) Initialization:
   set $i = 1$

2) Compute the vector $Q$:
   set $E_i = B(x)/B_i(x)$, $Q$ equal to the vector of coefficients $E_i$, placing $Q$ in the first column of the $n_i$ group. Set $j = 2$, $n_i=\text{deg}(B_i(x))^i$

3) Construct the remainder of $n_i$ columns:
   shift downward by one place all the elements in vector $Q$ while placing an element of value zero into first location. Add $Q$ to the matrix in the $j$th column of the $n_i$ group. If $j \neq n_i$, set $j = j+1$ and repeat step 3).

4) Set $i = i+1$. If $i > k$ then end; else return to step 2).

Since $E_i = \sum_{j=0}^{n_i-1} e_{i,j} x^j$ where $e_{i,j} \epsilon I$, the coefficient matrix for the numerator of the partial fraction terms is given in Figure 2.2. This matrix will also be employed when computing the transcendental part.
Figure 2.2 Coefficient MATRIX E
Procedure RSQDEC is now employed to obtain the partial fraction decomposition. From the left-hand side of equation (2.7) we construct a constant vector \( C \) from the polynomial \( A(x) \). From procedure MATSFD we have constructed the coefficient matrix \( E \). Using the vector \( C \) and the coefficient matrix \( E \) we then can proceed to solve a linear system of equations, the solution being the coefficients \( a_{i,j} \). From these the polynomials \( A_i(x) \) can be constructed.

Algorithm RSQDEC:

- **Input** is the rational function \( A(x)/B(x) \), while the output is the terms \( A_i \) and \( B_i \), \( i=1, \ldots, k \) such that
  \[
  A(x)/B(x) = \sum_{i=1}^{k} \frac{A_i(x)}{B_i(x)}
  \]

1) **Factorization:**
   Set \( Q = \text{PSQFRE}(B(x)) \). The result is a linear list (vector) of all the square free polynomials of \( B(x) \).

2) **Construct the coefficient matrix:**
   \( E = \text{MATSFD}(B(x), Q) \). Here we obtain the coefficient matrix given in Figure 2.2.

3) **Construct the constant vector** \( C \):
   Place the coefficients of the numerator \( A(x) \) of the rational function in vector \( C \).
4) Solve the system of linear equations:
Here we solve a system of linear equations
\[ E_\alpha = C \] using the ALTRAN procedure ASOLVE. The
solution \( \alpha \) is a vector listing the coefficients
of \( A_\alpha \). Set \( n_0 = 0 \), \( j=1 \)

5) Construct \( A_j \):
\[
A_j = \sum_{i=n_0}^{n_0+n_j-1} \alpha_i x^{i-n_0}
\]
set \( j=j+1 \)

If \( i = n \), the end; else \( n_0 = n_0 + n_j-1 \) and
repeat this step.

To compute the complete partial fraction decomposi-
tion, we now make use of procedure PCDEC.

Algorithm PCDEC:

Input is two polynomials \( A_\alpha \), \( B_\alpha \) and integer \( i \) such that

\[
A_\alpha/B_\alpha^i = \frac{1}{W} \sum_{j=1}^{i} Y_j(x)/B_\alpha^j(x)
\]

where \( W \) is a constant determined during the computation of
the algorithm. Output is vector \( Y \) and constant \( W \).

1) Initialize variables:
Set \( m = \text{degree } (A_\alpha(x)) \),
\( n = \text{degree } (B_\alpha(x)) \),
\( W = \{\text{LDC } (B_\alpha(x))\}^{m-n+1} \)
(where ldc represents the leading coefficient term),

\[ Y = 0 \]

\[ Q = W A_\alpha(x), \text{ Set } j = 1 \]

2) Compute \( Q' \) and \( Y_j \) such that:

\[ Q = B_\alpha(x) Q' + Y_j \]

If Deg \((Q') < n\), Set \( Y_{j+1} = Q' \) and end; else set \( Q = Q' \).

Set \( j = j+1 \). If \( j > i \) then end; else repeat this step.

Procedure RDEC provides the steps necessary to obtain the complete partial fraction decomposition.

Algorithm RDEC:

Input is the regular rational function \( A(x)/B(x) \).

Output are the terms of the complete partial fraction decomposition such that

\[
A(x)/B(x) = \sum_{i=1}^{k} \frac{1}{W_i} \left( \sum_{j=1}^{i} A_{i,j}(x)/B_i^j(x) \right)
\]

These include an array of the terms \( A_{i,j} \) and vectors for the terms \( B_i \) and \( W_i \).

1) Perform partial fraction decomposition:

Call RSQDEC \((A(x)/B(x))\). Set \( i = 1 \)
2) Perform the complete partial fraction decomposition for \( A_i(x)/B_i(x) \):

Call PCDEC \((A_i(x), B_i(x), i)\) Set \( i = i+1 \)

If \( i > k \) then end; else repeat step 2).

Let us now consider computing the rational part of integration using a reduction method. After computing the complete square free partial fraction decomposition we have the equation

\[
\int \frac{A(x)}{B(x)} \, dx = \sum_{i=1}^{k} \frac{1}{W_i} \sum_{j=1}^{i} \int A_i, j(x)/B_i^j(x) \, dx
\]

What is necessary is to integrate the terms \( A_i, j(x)/B_i^j(x) \) with respect to \( x \) for \( i > 1 \).

Since \( B_i(x) \) is a square free polynomial,

\[
\gcd(B_i(x), \frac{dB_i(x)}{dx}) = 1
\]

From theorem 2.2T2 there exist two polynomials \( C(x) \) and \( D(x) \) such that

\[
C(x) B_i(x) + D(x) \frac{dB_i(x)}{dx} = A_i, 1(x)
\]

for \( i > 1 \).

Then,

\[
\int \frac{A_i, 1(x)}{B_i^1(x)} \, dx = \int \frac{C(x)}{B_i^{i-1}(x)} \, dx + \int \frac{D(x) \frac{dB_i(x)}{dx}}{B_i^i(x)} \, dx
\]
Using integration by parts, we have

\[ \int \frac{A_{i,i}(x)}{B_i^i(x)} \, dx = \int \frac{C(x)}{B_i^{i-1}(x)} \, dx + \int \frac{dD(x)/dx}{(i-1)B_i^{i-1}(x)} \, dx \]

\[ - \frac{D(x)}{(i-1)B_i^{i-1}(x)} \]

which can be written as

\[ \int \frac{A_{i,i}(x)}{B_i^i(x)} \, dx = - \frac{D(x)}{(i-1)B_i^{i-1}(x)} + \int \frac{H(x)}{B_i^{i-1}(x)} \, dx \quad (2.8) \]

where

\[ H(x) = C(x) + \frac{1}{(i-1)} \frac{dD(x)}{dx}, \quad (2.9) \]

Since the \( \deg(C(x)) < \deg(B_i(x)) \) and the \( \deg(\frac{dD(x)}{dx}) < \deg(B_i(x)) \), we find that the \( \deg(H(x)) < \deg(B_i(x)) \)

Now let

\[ A_{i,i-1}^* = A_{i,i-1} + H(x) \quad (2.10) \]

where the \( \deg(A_{i,i-1}^*) < \deg(B_i(x)) \).

Proceeding in the same fashion we reduce by one the exponent of \( B_i(x) \) in \( A_{i,i-1}^*(x)/B_i^{i-1}(x) \) until we arrive at

\[ \int \frac{A_{i,i-1}(x)}{B_i(x)} \, dx \]

which is the transcendental part. Our result is then
\[
\int \frac{A(x)}{B(x)} \, dx = \sum_{i=2}^{k} S_i(x) + \sum_{i=1}^{k} \frac{A_{i,1}(x)}{B_i(x)} \, dx
\]

where \( \sum_{i=2}^{k} S_i(x) \) is the rational part of integration and \( \sum_{i=1}^{k} \frac{A_{i,1}(x)}{B_i(x)} \, dx \) is the transcendental part.

This formulation is implemented by the procedures HERM1 and HERM2. The procedure HERM1 uses the reduction procedures described above.

Algorithm HERM1:

Inputs will be a vector \( A_{i,1}, A_{i,2}, \ldots, A_{i,i} \) obtained from algorithm RDEC, \( B_i(x) \) and the integer \( i \). Output is a pair of polynomials \( R(x) \) and \( S(x) \) such that

\[
\int \sum_{j=1}^{i} \frac{A_{i,j}(x)}{B_i^j(x)} \, dx = R(x) + \int S(x) \, dx
\]

1) Initialize the rational and transcendental parts:
   Set \( R = 0, S = A_{i,i} \)

2) Use the identity discussed in theorem 2.2T2:
   Call PEDCD \( (B_i, dB_i/dx) \) to compute \( C(x), D(x) \) such that
   \[
   C(x) B_i(x) + D(x) \frac{dB_i(x)}{dx} = 1
   \]
   PEGCD is a user defined algorithm. Set \( j=i \).
3) Implementation of theorem 2.2T2:
Call EGCD \((B_i, dB_i/dx, S, C, D)\) to compute \(CC(x)\), \(DD(x)\) and \(W\), an integer such that
\[
W \cdot S = CC(x) \cdot B_i(x) + DD(x) \cdot dB_i(x)/dx
\]
where \(W \in \mathbb{I}\).
\(W\cdot S\) insures that the right hand side of the above equation has coefficients over the integers.

4) Compute the rational part:
Set \(R = R - DD(x)/[W \cdot (j-1) \cdot B_i^{j-1}]\)

5) Compute \(A_{i,j-1}^*\):
Set \(S = A_{i,j-1} + CC(x) + \frac{1}{(j-1)} \cdot \frac{dDD(x)}{dx}\)
Set \(j = j-1.\) If \(j > 1\) return to step 3)

6) Compute the transcendental part:
\(S = S/(W \cdot B_i(x)),\)
then end.

Procedure HERM2 will perform the integration for any regular rational function.

Algorithm HERM2: -

Input is a regular rational function \(A(x)/B(x)\)
while the output is 2 polynomials \(R(x)\) and \(S(x)\) such that
\[
\int \frac{A(x)}{B(x)} \, dx = R(x) + \int S(x) \, dx
\]
Procedure HERM1 calls upon procedure HERM2 and RDEC.

1) Initialize R and S:
   Set \( R = 0, \ S = 0 \)

2) Compute the complete partial fraction decomposition:
   Call RDEC(A/B) to compute the complete partial fraction terms.

3) Initialize the transcendental part:
   Set \( S = \frac{A_1,1}{B_1}(x) \)
   Set \( j = 2 \)

4) Reduction procedures:
   Call HERM1((A_{j,1}, A_{j,2}, \ldots, A_{j,j}), B_{j,j})
   to compute \( R_p \) and \( S_p \) such that
   \[
   \int \sum_{i=1}^{j} \frac{A_{j,i}(x)}{B_{j,i}(x)} \, dx = R_p + \int S_p
   \]

5) Sum the rational and transcendental parts:
   Set \( R = R + R_p/W_j, \ S = S + S_p/W_j \)
   \( W_j \) is obtained from RDEC
   Set \( j = j+1 \)
   If \( j \leq k \) return to step 4); else end.

The purpose of procedure HERM is to act as a supervisor for the integration of any rational function over the integers. If the rational function is not regular HERM converts it to a regular rational function plus a
Algorithm HERM: -

Input is any rational function called $AB = A(x)/B(x)$. Output is the integration of this function.

1) Initialize:
Set $R_1 = 0, AB^* = AB, R = 0, S = 0$

2) Test if $AB$ is a regular rational function:
If degree $(A(x)) < \text{degree} (B(x))$
go to step 4); else compute $A^*(x)$ and $R_1$
such that
$$A(x) = R_1(x)B(x) + A^*(x)$$
Then $AB^* = A^*(x)/B(x)$

3) Integration of polynomial $R_1$:
Set $R = \int R_1 \, dx$ using the ALTRAN system
procedure PINT.

4) Integration of the regular rational function:
Call HERM2 $(AB^*(x))$ to integrate the regular
rational function from which rational and
transcendental parts, $R_x$ and $S_x$ are computed.

5) Compute the final rational and transcendental parts:
Set $R = R + R_x, S = S_x$

A listing for the algorithm HERM is given in
Appendix A.

2.4 Horowitz's Method for Rational Function Integration

By Hermite's algorithm we were able to compute polynomials C and D such that

\[
\int \frac{A(x)}{B(x)} \, dx = \frac{C(x)}{B_2(x) \cdots B_{k-1}(x)} + \int \frac{D(x)}{B_1(x) \cdots B_k(x)} \, dx
\]  

(2.12)

where

\[
\frac{C(x)}{B_2(x) \cdots B_{k-1}(x)}
\]

is the rational part.

Using Hermite's method, we first obtained the partial fraction decomposition as described in Section 2.3 and then apply a reduction process to the partial sums

\[
\sum_{j=1}^{i} A_{i,j} / B_i^j
\]

for \(2 \leq i \leq k\).

Instead of Hermite's method, let us consider equation (2.12) above where \(C(x)\) and \(D(x)\) are undetermined polynomials. Differentiating both sides of equation (2.12) we have
\[
\frac{A(x)}{B(x)} = \frac{C'(B_2 \ldots B_k^{k-1}) - C(B_2 \ldots B_k^{k-1})'}{(B_2 \ldots B_k^{k-1})^2} + \frac{D}{B_1 \ldots B_k}
\]

\[
= \{C'(B_1 \ldots B_k)(B_2 \ldots B_k^{k-1}) - C(B_1 \ldots B_k)
\]

\[
(B_2 \ldots B_k^{k-1})' + D(B_2 \ldots B_k^{k-1})^2\}/
\]

\[
[(B_1 \ldots B_k)(B_2 \ldots B_k^{k-1})^2]
\]

(2.13)

But

\[
(B_2 B_3^{2} \ldots B_k^{k-1})' = (B_3 B_4^{2} \ldots B_k^{k-2}).
\]

\[
\frac{k}{i=2} (i-1)B_2 \ldots B_{i-1}B_i' \]

\[
B_{i+1} \ldots B_k
\]

(2.14)

[HOR 70, pp.103]

Substituting equation (2.14) into (2.13) we obtain

\[
\frac{A(x)}{B(x)} = \{C'(B_1 \ldots B_k) - C(\sum_{i=2}^{k} (i-1)B_1B_2 \ldots B_{i-1}B_i')
\]

\[
B_{i+1} \ldots B_k + D(B_2 \ldots B_k^{k-1})\}/
\]

\[
[(B_1 \ldots B_k)(B_2 \ldots B_k^{k-1})]
\]

(2.15)

where \( B = (B_1 \ldots B_k)(B_2 \ldots B_k^{k-1}) \)
Let $U(x) = \prod_{i=1}^{k} B_i(x)$, $V(x) = \prod_{i=2}^{k} B_i^{i-1}(x)$

$C'.U = \sum_{i=0}^{n-2} e_i x^i$ where $e_i = \sum_{j=0}^{m-2} (j+1)c_{j+1}u_{i-j}$,

$C.W = \sum_{i=0}^{n-2} f_i x^i$ where $f_i = \sum_{j=0}^{m-1} c_jw_{i-j}$,

$D.V = \sum_{i=0}^{n-1} g_i x^i$ where $g_i = \sum_{j=0}^{m-1} d_{i-j}v_j$

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

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

If $H = (c_{m-1}, \ldots, c_0, d_{n-m-1}, \ldots, d_0)$ and $A = (a_{n-1}, \ldots, a_0)$, then $H$ is a unique vector satisfying the equation

$EH = A$

where $E$ is the coefficient matrix given in Figure 2.3.

A flowchart showing the steps necessary in Horowitz's algorithm is given in Figure 2.4. A brief description of these procedures used in Horowitz's algorithm follows:

Algorithm MATX: -

Inputs are the polynomials $U$ and $V$ and vector $(B_1, B_2, \ldots, B_k)$ such that
\[ U = \prod_{i=1}^{k} B_i(x), \quad V = \prod_{i=2}^{k} B_{i-1}(x) \]

The polynomial \( W \) is constructed within this procedure. Output is the coefficient matrix given in Figure 2.3.

1) Initialize:
Set \( m = \deg(V) \), \( n = \deg(U) + m \)

\[
W = -\sum_{i=2}^{k} (i-1) \frac{U}{B_i} \cdot \frac{dB_i}{dx}
\]

2) Construct set of vectors:
Set \( L_V \) to the coefficient of polynomial \( V \),
\( L_U \) to the coefficient of polynomial \( U \) and
\( L_W \) to the coefficient of polynomial \( W \).
Set \( i = 1 \).

3) [Construct the first \( m \) columns of the coefficient matrix \( E \).]
Place column \( L_W \) in the \( m \)th column of the matrix \( E \)
Set \( j = 0 \), \( W = W \cdot X \)
a) Set \( X_1 = (W + (j+1) \cdot U) \cdot X^j \) and vector \( L_W \) to the coefficient of polynomial \( X_1 \), placing vector \( L_W \) in the \((m-1-j)\)th column of matrix \( E \).
Set \( j = j + 1 \)
If \( j < (m-1) \) go to a).
Figure 2.3 Coefficient MATRIX
4) [Construct the n-m columns of matrix E.]

Set $i = m + 1$
Set $j = 0$

b) Set $p = N - j$ Place vector $L_v$ in the $p$th column of matrix $E$, set $j = j + 1$. If $j > (n - m)$ then end; else shift up one place all the elements in vector $L_v$ and place element of value zero on the bottom. Repeat step b).

The purpose of procedure RINTG is to integrate a regular rational function.

Algorithm RINTG:

Input is a regular rational function $A(x)/B(x)$ while the output is the rational part $R(x)$ and the transcendental part $S(x)$.

1) Compute the square free factor of the denominator $B(x)$:

Call PSQFRE ($B(x)$) to obtain the square free polynomials $B_1, B_2, \ldots, B_k$.

2) Compute the polynomials $U$ and $V$:

Set $U = \prod_{i=1}^{k} B_i$, $V = \prod_{i=2}^{k} B_{i-1}$
Read AB=A/B

**AB Rational Function**

If AB non regular put it in this form
AB=AB*+R₁, where AB* is regular rational function, R₁ ∈ I[x]

**Factorization Algorithm**

\[ B = \prod_{i=1}^{k} Y_i(x) \]
\[ U = \prod_{i=1}^{k} Y_i(x) \]
\[ V = \prod_{i=2}^{k} Y_i(x) \]

\[ W = -\sum_{i=2}^{k} (i-1)B_1\cdots B_{i-1}B_i\cdots B_k \]

Construct matrix E for unknown coefficient from equation C*U+C*W+D+V

Construct the constant vector from the coefficient of numerator A.
Solve system of linear equation

\[ \begin{bmatrix} C_{m-1} \\ C_{m-2} \\ \vdots \\ C_0 \\ d_{n-m-1} \\ \vdots \\ d_0 \end{bmatrix} E = \begin{bmatrix} a_{n-1} \\ a_{n-2} \\ \vdots \\ \vdots \\ a_0 \end{bmatrix} \]

\[ C = \sum_{i=0}^{n-1} C_i x^i \]
\[ D = \sum_{i=0}^{n-m-1} d_i x^i \]

\[ R = C/V \quad S = D/V \]

\[ R = R + \int R_1 \quad S = S \]

Write AB,R,S

Figure 2.4 Horowitz Algorithm
Compute \( V_i, U_i, W_i \) such that

\[
V(x) = \sum_{i=0}^{m} V_i x^i = \prod_{i=2}^{k} B_i^{i-1} \tag{2.16}
\]

\[
U(x) = \sum_{i=0}^{n-m} U_i x^i = \prod_{i=1}^{k} B_i \tag{2.17}
\]

\[
W(x) = \sum_{i=1}^{k} \{(i-1)B_1, \ldots, B_{i-1}B_iB_{i+1}, \ldots, B_k\} = \sum_{i=0}^{n-m-1} w_i x^i \tag{2.18}
\]

3) Construct the coefficient matrix \( E \):

Call MATX to construct the coefficient matrix \( E \) given from the equation

\[
A = C'U + CW + DV \tag{2.19}
\]

where

\[
C(x) = \sum_{i=0}^{m-1} C_i x^i,
\]

\[
D(x) = \sum_{i=0}^{n-m-1} d_i x^i
\]

\[
C'(x) = \sum_{i=0}^{m-2} (i+1) C_{i+1} x^i
\]

4) Solve system of linear equation:

Construct constant vector \( F \) from the coefficients of the numerator \( A(x) \). Solve the system of linear equations
using the Altran procedure ASOLVE

5) Compute polynomials $C(x)$ and $D(x)$:

Set $C(x) = \sum_{i=0}^{m-1} h_i x^{m-1-i}$

where the first $m$ elements of $H$ are the coefficients of $C$.

Set $D(x) = \sum_{i=m}^{n-1} h_i x^{n-i-1}$

where $(m+1)$th to $(n)$th elements of $H$ are coefficients of $D$.

6) Rational and transcendental computation:

Set $R = C(x)/(\prod_{i=2}^{k} B_i(x))$

Set $S = D(x)/(\prod_{i=1}^{k} B_i(x))$

Procedure RINTG acts as a supervisor procedure for Horowitz's algorithm. It's primary purpose is to reduce any nonregular rational function into a regular
rational function plus a separate polynomial. Procedure RINTGS calls upon procedure RINTG to perform the integration of the regular rational function. Steps taken by RINTGS are similar to those in procedure HERM.

2.5 Discussion on the Methods and Empirical Results

It is clear from Hermite's algorithm that considerable computation time is needed for complete partial fraction decomposition. This time depends upon the coefficient bound, the coefficient bound being related to the norm of the coefficients of a polynomial, and the order of the denominator of the input function. From the examples the time taken to perform complete partial fraction decomposition in itself is greater than the time taken for the complete Horowitz algorithm.

In the more efficient implementation where modular reduction [HOR 70] is used to compute the complete partial fraction decomposition the execution time is proportional to $O(n^4 \cdot CB^2)$ where $n$ is the degree of the denominator $B$ and $CB$ is proportional to the coefficient bound. However the difference in the computation time of modular reduction over I is approximately equal to direct computation when both $n$ and $CB$ are small. In addition the time taken by the reduction technique due to Hermite's is proportional to the square of the number of the square free polynomials of the denominator [HOR 70, pp. 96].

The method discussed by Horowitz avoids the partial
fraction decomposition and instead requires the solution of a system of linear equations. Execution time of Horowitz's method depends upon having an efficient procedure, such as ASOLVE in ALTRAN, to compute these solutions. In addition, Horowitz's method is not dependent upon the number of square free polynomials of the input denominator B.

Due to the storage of partial fraction decomposition required by Hermite's algorithm, this method requires more storage than Horowitz's algorithm. While the problem is not an academic one, it could become serious for small algebraic systems.

A comparison of execution times for Hermite's and Horowitz's method is given in Figure 2.

2.6 Extension of Rational Function Integration to Multivariate Rational Functions

A multivariate polynomial $F$ can be written as

$$F(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{m} f_i x_1^i$$

where the coefficient $f_i$ is a polynomial in $(n-1)$ variables over the integers,

$$f_i \in \mathbb{Z}[x_2, x_3, \ldots, x_n]$$

and

$$m = \text{degree of } F(x_1, x_2, x_3, \ldots, x_n)$$

with respect to $x_1$.

A multivariate rational function is simply the ratio of two multivariate polynomials. In performing multivariate rational function integration, all operations are performed with respect to main variables (main
indeterminate). All definitions and theorems discussed in section 2.2 concerning polynomials of a single variable with integer coefficients can be extended to multivariate polynomials over the integers. For example, in the case of the square free polynomial factorization, a multivariate polynomial $B(x_1, x_2, \ldots, x_n)$ can be factored into $B_1, B_2, \ldots, B_k$ such that

$$B(x_1, \ldots, x_n) = a * B_1(x_1, \ldots, x_n)^{n_1} * B_2(x_1, x_2, \ldots, x_n)^{n_2} * \ldots * B_k(x_1, x_2, \ldots, x_k)^{n_k}$$

where the degree $(B_i(x_1, x_2, \ldots, x_n))$ is with respect to $x_1$ and is greater than zero and $a \in I[x_2, x_3, \ldots, x_n]$. The only change required is in the procedure PSQFRE where derivatives are taken with respect to $x_1$ instead of $x$.

In the same fashion it can be shown that the integration of the multivariate rational function $A(x_1, x_2, \ldots, x_n)/B(x_1, x_2, \ldots, x_n)$ can be written in the form

$$\int \frac{A}{B} \, dx_1 = \frac{C(x_1, \ldots, x_n)}{\gamma B_2(x_1, \ldots, x_n) B_3(x_1, \ldots, x_n)^2 \ldots B_k(x_1, \ldots, x_n)^{k-1}} + \int \frac{D(x_1, \ldots, x_n)}{\gamma B_2(x_1, \ldots, x_n) B_3(x_1, \ldots, x_n) \ldots B_k(x_1, x_2, \ldots, x_n)} \, dx_1$$

where $B(x_1, x_2, \ldots, x_n) = \alpha \prod_{i=1}^{k} B_i(x_1, \ldots, x_n)$

and $\alpha, \gamma \in I[x_2, x_3, \ldots, x_n]$. 
The method discussed in section 2.4 due to Horowitz applies as well to multivariate rational functions. In the case of matrix E given in Figure 2.3 the coefficients of the matrix will be polynomials in \((n-1)\) variables. Again when solving exactly the system of linear equations with multivariate polynomials as coefficients, the ALTRAN procedure ASOLVE is used.

Examples are given in Figure 2.5.
Example 1

\[ AB \]
\[ \frac{X^2 + X + 1}{(X + 1)^2(X + 2)} \]

\[ R \]
\[ -1 \]
\[ (X + 1) \]

\[ S \]
\[ \frac{X - 1}{(X + 1)(X + 2)} \]

Example 2

\[ AB \]
\[ \frac{1}{(X - 3)^3(X - 2)^3(X - 1)^2(X^2 + 1)} \]

\[ R \]
\[ \frac{37X^4 - 227X^3 + 342X^2 + 148X - 400}{400(X - 3)^2(X - 2)^2(X - 1)} \]

\[ S \]
\[ \frac{37X^3 + 138X^2 + 33X + 142}{400(X - 3)(X - 2)(X - 1)(X^2 + 1)} \]

Example 3

\[ AB \]
\[ \frac{X^2 + 2X + 2}{(X + 1)^3(X + 2)^2(X + 3)} \]

\[ R \]
\[ \frac{13X^2 + 30X + 16}{4(X + 1)^2(X + 2)} \]

\[ S \]
\[ \frac{13X + 34}{4(X + 1)(X + 2)(X + 3)} \]

Figure 2.5 (a) Examples on Univariate Rational Function Integration
Example 4

\[ \frac{1}{((x + 1) * (x^3 + 1))^3} \]

\[ \frac{-(28x^5 - 12x^4 - 40x^4 + 49x^3 - 15x^2 - 64x + 18)}{(162 * (x + 1))^3 * (x^2 - x + 1)^2} \]

\[ -2 * (7x - 20) / (81 * (x + 1) * (x^2 - x + 1)) \]

Example 5

\[ \frac{1}{((x^4) * (x + 1))^3 * (x + 3)^2 * (x^2 + 2)} \]

\[ \frac{-(5279x^5 + 23791x^4 + 25640x^3 + 4884x^2 - 1254x + 396)}{(7126x^3) * (x + 1)^2 * (x + 3)} \]

\[ -2 * (5279x^3 + 15908x^2 + 10684x + 31636) / (7126x * (x + 1) * (x + 3) * (x^2 + 2)) \]

figure 2.5 (b) Examples on Univariate Rational Function Integration
\[
\begin{align*}
\text{v AB} & \quad \quad \frac{x(1)^*5}{(x(1) + x(3))^{*2}} - \frac{(x(1)^*2 + x(2) + x(3))^*2}{(x(1) + x(3))^{*2}} \\
\text{v R} & \quad \quad (x(1)^*3x(2) + x(1)^*3x(3)^{*2} + x(1)^*3x(3) - 3x(1)^*2x(2)x(3) - 3x(1)^*2x(3)^*3 - 3x(1)^*2x(3)^{*2} - 4x(1)^*x(2)^*x(3)^*2 - 4x(1)^*x(3)^*4 - 4x(1)^*x(3)^{*3} + 4x(2)^*x(3)^*4 - 16x(3)^*5 + 4x(3)^*4) / (2 + (x(1) + x(3))^{*2}) \\
\text{v S} & \quad \quad (-x(1)^*2x(2)^{*2} - 2x(1)^*2x(2)^*x(3)^*3 + x(1)^*2x(2)^*x(3)^{*2} - 3x(1)^*2x(3)^*4 - 3x(1)^*2x(3)^{*3} + 6x(1)^*x(2)^*x(3)^{*3} + 2x(1)^*x(2)^*x(3)^*2 - 6x(1)^*x(3)^*4 + x(1)^*x(3)^{*3} - 4x(2)^*2x(3)^*2 + 3x(2)^*x(3)^{*4} - 6x(2)^*x(3)^{*3} + 3x(2)^*5 - 4x(3)^{*4}) / \\
& \quad \quad ((x(1) + x(3))^* (x(2) + x(3)^{*2} + x(3))^*)^* \\
& \quad \quad ((x(1)^*2 + x(2) + x(3))^{*2} + x(3))^{*3}) \\
\end{align*}
\]

Figure 2.6 Example on Multivariate Rational Function Integration Using Horowitz method
<table>
<thead>
<tr>
<th>HERMITE</th>
<th>HOROWITZ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete partial fraction</td>
<td>final time</td>
</tr>
<tr>
<td>19.2</td>
<td>22.828</td>
</tr>
<tr>
<td>55.0</td>
<td>68.297</td>
</tr>
<tr>
<td>6.51</td>
<td>8.169</td>
</tr>
<tr>
<td>45.9</td>
<td>58.756</td>
</tr>
<tr>
<td>67.1</td>
<td>73.20</td>
</tr>
</tbody>
</table>

Table 1
Comparison between Hermite and Horowitz methods
CHAPTER 3
POLYNOMIAL FACTORIZATION OVER INTEGERS

The purpose of this chapter is to discuss polynomial factorization. We will begin with a brief history followed by a discussion of Berlekamp's [BER 67] and Zassenhaus [ZAS 69] algorithms followed by a description for univariate and multivariate factorization over integers [WAN 73].

3.1 Introduction to Factoring Problem

In the area of symbolic algebraic manipulation, polynomial factoring is an important operation not only as an operation in itself but also in the operation of symbolic integration and that of solving polynomial equations. Because of this importance the goal of many researchers has been to obtain an efficient algorithm for factoring polynomials.

One of the first methods employed to factor polynomials was provided by Kronicker's algorithm [VAN 53, pp.77]. This technique involved finding a set of integral factors for the given polynomial to be factored, choosing an element from the set of integral factors, computing by interpolation a unique polynomial and testing to see if this unique polynomial could be divided into the given polynomial. If the division was exact a factorization was
found and the method could be applied recursively to the two factors. Else the unique polynomial would be discarded and another element from the set of integral factors would be chosen.

When the degree and number of coefficients of the given polynomial to be factored are small, Kronecker's algorithm performs well. For a large number of coefficients considerable time is required to factor the given polynomial into primes. If the degree of the given polynomial is large, then an enormous number of possible choices must be made in finding unique polynomials. An increase in either the number of coefficients or the degree of the given polynomial causes an exponential growth in the computing time. Attempts to decrease the computation time of Kronecker's algorithm have not relieved the basic problem of exponential growth. Because of this alternative methods have been developed for performing factorization.

One of these more efficient algorithms has been mod p factorization by Berlekamp [BER 68]. Berlekamp's algorithm has paved the way for algorithms based on mod p factorizations rather than on integer factorizations.

At the suggestion of Zassenhaus [ZAS 69], one can combine Berlekamp's algorithm and Hensel's Lemma to obtain a practical method of factoring polynomials with integer coefficients. This of course has been successfully
demonstrated by Musser [MUS 71] and has been extended to multivariate factorization by Wang [WAN 73].

3.2 Factoring Polynomials over Finite Fields

Let \( U(x) \) be a square free polynomial in the Euclidean Domain \( \mathbb{Z}_p[x] \). Our goal is to find a set of irreducible factors

\[
F_1(x), F_2(x), \ldots, F_r(x)
\]

such that

\[
U(x) = \prod_{i=1}^{r} F_i(x) \pmod{p}
\]

where each \( F_i(x) \) is a distinct relatively prime polynomial over \( \mathbb{Z}_p[x] \). Berlekamp's algorithm [BER 67] is the basis for much of this work and is briefly outlined for the benefit of the reader.

Berlekamp's technique is to make use of the Chinese remainder theorem which is valid for polynomials as well as integers. If the set of residues \((s_1, s_2, \ldots, s_r)\) are any \( r \)-tuple of integers over the integer field \( \mathbb{Z}_p \), the Chinese remainder theorem implies that there exist a unique polynomial \( V(x) \) such that

\[
V(x) \equiv s_i \pmod{F_i(x)}
\]

for \( i=1, \ldots, r \).

If \( V(x) \) can be found, then we can also obtain the factors \( F_i(x) \) of \( U(x) \) for if \( r \geq 2 \) and \( s_i \nmid s_j \), \( i \neq j \), we will
find that \( \text{gcd} \left( U(x), V(x) - s_i \right) \) (where gcd stands for the greatest common-divisor) is divisible by \( F_i(x) \) but not by \( F_j(x) \).

Since we can obtain further information about the factors of \( U(x) \) from solutions \( V(x) \) of equation (3.2), let us consider equation (3.2) more closely. In the first instance, the polynomial \( V(x) \) satisfies the condition that

\[
V(x)^p = s_j^p \pmod{F_j(x)}
\]

\[
= s_j \equiv V(x) \pmod{F_j(x)} \tag{3.4}
\]

for \( j = 1, \ldots, r \) \([\text{KNU } 69, \text{ pp. } 382]\).

Therefore,

\[
V(x)^p \equiv V(x) \pmod{U(x)} \tag{3.5}
\]

where the \( \deg(V(x)) < \deg(U(x)) \).

In the second instance,

\[
V(x)^p - V(x) = (V(x) - 0)(V(x) - 1) \ldots (V(x) - (p-1)) \tag{3.6}
\]

is an identity for any polynomial \( U(x) \) when working in modulo \( p \). If \( V(x) \) satisfies equation (3.5), then it follows that \( U(x) \) divides the left hand side of equation (3.6). Thus every irreducible factor of \( U(x) \) must divide one of the \( p \) relatively prime factors of the right side of equation (3.6) with all solutions of equation (3.5) having the form of equation (3.4) for some \( s_1, s_2, \ldots, s_r \). In this case
there are exactly $p^r$ solutions of equation (3.4).

Solutions to equation (3.4) thus provide the basis to the factorization of $U(x)$. If we consider the degree of $U(x)$ to be equal to $n$, we can construct an $nxn$ matrix $Q$ of row vectors $q_k$ where

$$x^{pk} \equiv \sum_{i=0}^{n-1} q_{k,i} x^i \pmod{U(x)}$$

(3.7)

Then

$$V(x) = \sum_{j=1}^{n-1} v_j x^j$$

is a solution to equation (3.4) if and only if

$$V Q = V \text{ or } V (Q-I) = 0$$

where $\tilde{V}$ is the coefficient vector of $V(x)$. This reduces the problem to finding null space vectors and thus a set of irreducible vectors (independent vectors). The number of these irreducible vectors is equal to the number of prime factors of $U(x)$ over $\mathbb{Z}_p[x]$.

To obtain the residues $s_j$, different techniques have been discussed by Berlekamp [BER 67] and Collins [COL 69]. In both successive elements of the finite integer field are searched using the gcd $(U(x), V(x) - s_j)$ divisible by $F_j$. In order to obtain all the irreducible polynomial this search is repeated until the number of irreducible polynomials are equal to the number of irreducible vectors.
3.3 Zassenhaus Algorithm using Hensel's Lemma

Much of what is to be discussed in this section is based upon the studies of Musser [MUS 71].

Zassenhaus has suggested that by using Hensel's Lemma one can construct a factorization mod $p^j$ from a given factorization mod $p$. The algorithm to be discussed is for two factors and is simple to extend to more than two.

Let $p_1 = p$ and let

$$U(x) = F(x)G(x) \mod p_1 \quad (3.8)$$

There exist two polynomials $C(x), D(x) \in Z_{p_1}[x]$ such that

$$C(x)F(x) + D(x)G(x) = 1 \mod p_1 \quad (3.9)$$

Since $(U(x)-F(x)G(x))$ is divisible by $p_1$ from equation (3.8), we can compute

$$T(x) = (U(x)-F(x)G(x))/p_1 \quad (3.10)$$

where the $\deg (T(x)) \leq \deg (U(x))$

Thus we can write the identity

$$C_1(x)F(x) + D_1(x)G(x) = T(x) \mod p_1 \quad (3.11)$$

Using Hensel's Lemma let

$$F_n(x) = F(x) + p_1D_1(x) \quad (3.12)$$

$$G_n(x) = G(x) + p_1C_1(x) \quad (3.13)$$

Then,
\[ U(x) \equiv F_n(x)G_n(x) \pmod{p_1^2} \quad \text{(3.14)} \]

To prove this, multiply equation (3.12) and (3.13) to obtain

\[
F_n(x)G_n(x) = F(x)G(x) + p_1(F(x)C_1(x) + G(x)D_1(x)) \\
+ p_1^2 D_1(x)C_1(x) \\
= F(x)G(x) + p_1T + p_1^2 C_1(x)D_1(x) \\
= U(x) + p_1^2 D_1(x)C_1(x) \\
\equiv U(x) \pmod{p_1^2}
\]

where \( p_1^2 D_1(x) C_1(x) \) vanishes.

Since we wish to continue computation until we obtain a \( F_n(x) \) and \( G_n(x) \) such that

\[
U(x) \equiv F_n(x)G_n(x) \pmod{p_j} \quad \text{(3.15)}
\]

let \( p_1 = p_1^2 \) if \( p_1^2 < p_j \), else \( p_1 = p_j/p_1^2 \).

Here \( F \) and \( G \) become \( F_n \) and \( G_n \) respectively and we return to equation (3.10).

Much of what is to be discussed in the following two sections is based upon a paper written by Wang [WAN 73].
3.4 Factoring a Univariate Polynomial over the Integers

The following algorithm to be discussed involves the factorization of a square free primitive polynomial. If the given polynomial fails to be a primitive polynomial, that is where all of its coefficients are relatively primed, then the given polynomial is divided by its content. That is

$$pp(U(x)) = U(x)/\text{cont}(U(x))$$

where the content of $U(x)$ is the gcd of the coefficients. In the case that the given polynomial is not square free, we can obtain a square free polynomial from the equation

$$D = \gcd(U(x), d(U(x))/dx)$$

where

$$U(x) = \frac{U(x)}{D},$$

$D$ being a factor. Separate factorization must be done for $D$ and $U/D$.

In the process of choosing a prime number $p$, we should satisfy both the previous conditions, i.e.,

$$U(x) = \hat{U}(x) \ [\text{mod } p]$$

where $\hat{U}(x)$ is a square free prime polynomial over $\mathbb{Z}_p[x]$, $\hat{U}(x)$ having the same degree of $U(x)$. Since Wang's algorithm requires that we construct factors over modulo $p^{2^j}$, we can estimate the integer $j$ from computing a coefficient bound $B$. The coefficient bound $B$ for the
polynomial $U(x)$ where

$$U(x) = \sum_{i=1}^{n} u_i x^i, \quad u_i \in \mathbb{Z}_p[x]$$

satisfies the relationship

$$B > 2 \max\{|u_n|, |u_{n-1}|, \ldots, |u_0|\}$$

Thus, $j$ is the smallest integer which satisfies the relationship $B < p^{2^j}$.

Factors of a univariate polynomial over the integer can be constructed by the following steps.

1) First factor over a finite field (modulo $p$) using algorithm discussed in section 3.2. From that we obtain

$$U(x) = F_1(x) F_2(x) \ldots F_r(x) \pmod{p}$$

2) Then applying Zassenhaus's algorithm we can then find

$$U(x) = \hat{F}_1(x) \hat{F}_2(x) \ldots \hat{F}_r(x) \pmod{p^{2^j}}$$

3) Using an algorithm TRUEFACTOR, we can find all the irreducible polynomials over the integers as follows. If $U(x)$ is a monic polynomial then $\hat{F}_1(x), \hat{F}_2(x), \ldots, \hat{F}_r(x)$ are all monic polynomials. Else we calculate polynomials $H_1(x), H_2(x), \ldots, H_r(x)$ using the relationship
\[ H_i(x) = \text{ldc}(U(x)) 1 \text{dc}(\hat{F}_i)^{-1} F_i \pmod{p^{2^j}} \]
\[ = \hat{F}_i \prod_{j=1}^{r} 1 \text{dc}(\hat{F}_i) \pmod{p^{2^j}} \]

where ldc stands for the leading coefficient.

If \( F_i \) or \( H_i \) divides \( U \), \( F_i \) or \( \text{pp}(H_i) \) will be an irreducible factor of \( U \). Otherwise, the irreducible factor will be equal to the product of two or more of \( F_i \) or \( H_i \) until all the irreducible polynomials are found.

3.5 Multivariate Polynomial Factorization over the Integers

In this section we will discuss briefly Wang's algorithm for factorization of multivariate polynomials. It is assumed that the given polynomial is both square free and primitive. If not, the content and primitive part are factored separately. The polynomial \( U \) is now a function of \( x_1, x_2, \ldots, x_n \).

The first step is to begin with variable substitution. We first wish to find a set of integers \( \{a_2, a_3, \ldots, a_n\} \) such that \( U(x) = U(x_1, a_2, \ldots, a_n) \) is a square free polynomial with degree equal to the degree of \( U(x) \). Values of \( a_i \) are found by trial and error. First choices for the integers \( a_i \) should be 0, 1 and -1 since they usually make the coefficients of \( U(x) \) small in size. Since each \( a_i \) which
is non zero could cause some intermediate expression growth when using the extended Zassenhaus algorithm, it is more than desirable to use as many zeros as possible for substitution. If the zeros can not satisfy our square free conditions, we change one of the zero variables to $\pm K$ where $K = 1, 2, 3, \ldots$ until our square free conditions are satisfied. At this point

$$UI(x_1) = U(x_1, a_2, a_3, \ldots, a_n) = \sum_{i=0}^{m} UI_i x_1^i$$

where $UI_i \in \mathbb{Z}, \ i=0,1,\ldots,m$

The second step is factoring the polynomial $UI(x_1)$ over the integers using methods discussed in section 3.4. At this point

$$UI(x_1) = F_1(x_1) F_2(x_1) \ldots F_r(x_1)$$

The third step involves the construction of multivariate factors. If $A$ is the set of elements $\{a_2, a_3, \ldots, a_n\}$ over $\mathbb{Z}[x_2, x_3, \ldots, x_n]$, then the ideal generated by $A$ can be defined as $\sum r_i \alpha_i$ where

$$r_i \in \mathbb{Z}[x_2, x_3, \ldots, x_n],$$

$i=1,2,\ldots,n$ [VAN 53, pp.49]. If $k$ is any integer greater than zero then $A^k$ is defined as the ideal generated by all the polynomials of the form

$$\prod_{i=2}^{n} a_i^{c_i}$$

where $c_i > 0$
and

\[ \sum_{i=2}^{n} c_i = k \]

If given two polynomials \( F \) and \( G \), their main variable being \( x_1 \) and their coefficient over \( \mathbb{Z}[x_2, x_3, \ldots, x_n] \), then

\[ F = G \pmod{A^k, p} \]

if

\[ F \equiv G \pmod{A^k, p} \]

and the degree \( F \) in \( x_2, x_3, \ldots, x_n \) is less than \( k \). At this point we can define the ideal \( S \) as \( (x_2-a_2, x_3-a_3, \ldots, x_n-a_n) \) such that

\[ U(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{r} F_i \pmod{S} \]

where \( F_i \) are the true factors of \( U(x_1, a_2, a_3, \ldots, a_n) \) over the integers given by the second step. To construct the multivariate factors for \( U \) we must compute \( \hat{F}_i \) such that

\[ U(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{r} \hat{F}_i \pmod{S^k, p^{2^j}}, \]

this being an extension to the Zassenhaus algorithm. For two factors, let

\[ U(x_1) = F(x_1)G(x_1) \pmod{S} \]

If we let \( y_i = x_i - a_i \), \( i=2,3,\ldots,n \)

then

\[ V(x_1, y_2, \ldots, y_n) = U(x_1, y_2+a_2, \ldots, y_n+a_n) \pmod{p^{2^j}} \]
and
\[ R_1 = F(x_1)G(x_1) - V(x_1,y_2,\ldots,y_n) \]
\[ = W_1 + R_2 \]
where
\[ W_1(x_1,y_2,\ldots,y_n) \equiv R_1 \pmod{(S^2,p^{2^j})} \]
and
\[ R_2 = 0 \pmod{(S^2,p^{2^j})} \]

Since \( F(x_1) \) and \( G(x_1) \) are relatively prime polynomials, we can find \( \alpha_i(x_1) \) and \( B_i(x_1) \in \mathbb{Z}_p^{2^j}[x_1] \) such that
\[ \alpha_i(x_1) F(x_1) + B_i(x_1) G(x_1) = x_1^i \]
where \( i = 0,1,\ldots,n \) and \( n \) is equal to the degree of \( U(x_1) \).

Since any polynomial can be represented by
\[ T(x_1,y_2,\ldots,y_n) = \sum_{i=0}^{m_1} t_i x_1^i \]
where
\[ t_i \in \mathbb{Z}_p^{2^j}[y_2,y_3,\ldots,y_n]. \]

Then,
\[ T(x_1,y_2,\ldots,y_n) = \sum_{i=0}^{m_1} t_i [\alpha_i F + \beta_i G] \]
\[ = F(x_1)\left( \sum_{i=0}^{m_1} t_i \alpha_i(x_1) \right) \]
\[ + G(x_1)\left( \sum_{i=0}^{m_1} t_i \beta_i(x_1) \right) \]
Now
\[ W_i(x_1, y_2, \ldots, y_n) = F(x_1) \times (\sum_{i=0}^{j} w_i \alpha_i(x_1)) \]
\[ + G(x_1) \times (\sum_{i=0}^{j} w_i \beta_i(x_1)) \]

where \( j \) is the degree \((W_1(x_1))\) and

\[ w_i \in \mathbb{Z}_p 2^j(y_2, \ldots, y_n) \]

for \( i=0, 1, \ldots, j \)

To continue as was done in Zassenhaus using Hensel's Lemma (section 3.3), we can find \( F_n \) and \( G_n \) such that

\[ F_n = F - \sum_{i=0}^{j} w_i \beta_i(x_1) \]
\[ G_n = G - \sum_{i=0}^{j} w_i \alpha_i(x_1) \]

What we now wish to prove is that

\[ V(x_1, y_2, y_3, \ldots, y_n) = F_n G_n \quad \text{[mod } S^2\text{]} \]

where\( F_n \) and \( G_n \) \in \( \mathbb{Z}_p 2^j(x_1, y_2, \ldots, y_n) \)

\[ F_n G_n = F G - [F \sum_{i=0}^{j} w_i \alpha_i + G \sum_{i=0}^{j} w_i \beta_i] \]
\[ + [\sum_{i=0}^{j} w_i \alpha_i \cdot \sum_{i=0}^{j} w_i \beta_i] \]
\[ = FG - W_1 + \left[ \sum_{i=0}^{j} w_i \alpha_i \cdot \sum_{i=0}^{j} w_i \beta_i \right] \]

But

\[ \sum_{i=0}^{j} w_i \alpha_i \cdot \sum_{i=0}^{j} w_i \beta_i \equiv 0 \pmod{S^2} \]

Thus,

\[ V(x_1, y_2, \ldots, y_n) = FG - W_1 \pmod{S^2} \]

Allowing \( F_n G_n = V(x_1, y_2, \ldots, y_n) \pmod{S^2} \) one repeats the calculations letting \( F_n \) and \( G_n \) for \( F \) and \( G \) respectively until \( R_2 = 0 \) or until \( K \) is equal to one plus the degree of \( U \) in \( x_2, x_3, \ldots, x_n \). To extend this algorithm for more than two factors we can employ the following simple technique. First, let

\[ U(x_1, y_2, \ldots, y_n) = F_1 F_2 \ldots F_r \pmod{S} \]

Let

\[ G = F_2 F_3 \ldots F_r \pmod{S} \]

\[ U(x_1, y_2, \ldots, y_n) = F_1 G \pmod{S} \]

Using the extended Zassenhaus algorithm we can obtain

\[ U(x_1, y_2, \ldots, y_n) = \hat{F}_1 \hat{G} \pmod{S^k} \]

It can be shown that

\[ \hat{G}(x_1, y_2, \ldots, y_n) = F_2 F_3 \ldots F_k \pmod{S} \]
Set \( U = \hat{G} \) such that

\[
U = \prod_{i=2}^{r} F_i \pmod{S}
\]

\( = F_2 G \pmod{S} \)

and repeat the Zassenhaus algorithm to obtain \( \hat{F}_2 \) and \( \hat{G} \).

Continue this process until we obtain \( \hat{F}_r \).

In the fourth step we apply the algorithm called TRUEFACTORS to the polynomial

\[
U(x_1, x_2, \ldots, x_n) = \hat{F}_1(x_1, x_2, \ldots, x_n) \ldots \hat{F}_r(x_1, \ldots, x_n) \pmod{(S^k, p^{2^j})}
\]

to obtain the actual factors. Here the computation is more complex since we are dealing with two type modulo \([S^k, p^{2^j}]\) instead of modulo \([p^{2^j}]\).
3.6 Implementation of Wang's Algorithm for Factoring Multivariate Polynomials

There are eight major steps in Wang's algorithm for computing the irreducible factors of a multivariate polynomial \( U \) over \( \mathbb{Z} \). A flowchart designating these steps is given in Figure 3.1.

The first procedure \textsc{VSUBT} substitutes a set of integers \( \{a_2, a_3, \ldots, a_n\} \) into \( U \) such that

\[
U_I(x) = U(x, a_2, \ldots, a_n)
\]

is square free and the degree of \( U_I \) is equal to the degree of \( U \).

In the second step we make use of the Altran function \textsc{HPRIME} to return a prime number \( p \) larger than its argument. If \( U_I \text{ [modulo } p\text{]} \) is a square free polynomial with its degree equal to the degree of \( U \), we go to step 3. Else we repeat this step to obtain a larger prime number \( p \) using function \textsc{HPRIME}.

In the third step we obtain an irreducible polynomial over \( \mathbb{GF}(p) \) using a procedure called \textsc{PPOINT}. This procedure obtains a set of factors \( Z_1, Z_2, \ldots, Z_T \) such that

\[
U_I = Z_1 * Z_2 * Z_3 * \ldots * Z_T \quad \text{[modulo } p\text{]}
\]

It is in this step that we employ Berlekamp's algorithm described earlier.

In the fourth step we compute the coefficient
bound using a procedure called CBOUND. It is from this
procedure that we compute the modulus PQ equal to $p^{2^j}$.

In step five, procedure PFCI performs the
Zassenhaus algorithm from which we compute an array $F$
such that

$$UI = F_1 \times F_2 \times \ldots \times F_T \quad [\text{modulo PQ}]$$

In step six we obtain the univariate factors
using the procedure TRUFAC. The results are factors over
the integrals, that is,

$$UI = H_1 \times H_2 \times \ldots \times H_r$$

where $r \leq T$.

In step seven we apply the extended Zassenhaus
algorithm to obtain the multivariate factors $Y_1, Y_2, \ldots, Y_r$
such that

$$U = Y_1 \times Y_2 \times \ldots \times Y_r \quad [\text{modulo } (PQ, S^h)]$$

In the final step, we again apply the procedure
TRUFAC to obtain the actual factors $FAC_1, FAC_2, \ldots, FAC_L$
where

$$U = FAC_1 \times FAC_2 \times \ldots \times FAC_L$$

The result is expressed as a vector.

The following algorithms listed in the flowchart
of Figure 3.1 will briefly be discussed.
Algorithm VSUBT:

Input is the multivariate polynomial $U$ of $n$ variables. Output is an array of integers used for substitution in $U$. $UI$ is a univariate polynomial such that

$$UI = U(x_1, a_2, \ldots, a_n)$$

1) Initialization:
   Set $K_1 = 1$, $M = 1$, $i = 1$

2) Test the leading and trailing coefficient term of $U$:
   If a variable or variables of the set 
   \{x_2, x_3, \ldots, x_n\} can be factored from the leading coefficient term of $U$, assign a value of $K_1$ to the variable or variables. Do the same for the trailing coefficient term except that
   the assigned value will be $(K_1+1)^i \mod 5$ instead of $K_1$. Set the remainder of the variables equal zero. These values correspond with the elements of the vector $A$.

3) Substituting into $U$ and testing a square free polynomial and a nonvanishing leading coefficient term of $U$:
   Set $UI = U(x_1, a_2, \ldots, a_n)$. If degree $(UI) =$ degree $(U)$ and $UI$ is a square free polynomial, set $A = (a_2, a_3, \ldots, a_n)$ and end.
4) Else reinitialize the set \{x_2,x_3,\ldots,x_n\}:
   If \(x_i\) is not one of the factors of the leading
   or trailing coefficient term of \(U\), set \(a_i\)
   equal to zero.

5) Set \(x_{i+1}\) to a new value:
   Set \(j = i+1\)
   From \(i = j\) to \(n\), do; if \(a_i = 0\), set \(a_i = K_1\)
   and return to step 3, else continue

6) Define new values for \(K_1\):
   If \(K_1 > 0\) set \(K_1 = -K_1\) and go to step 7),
   else set \(K_1 = -K_1 + 1\)  If \(K_1 < (M+5)\) go to step 7)
   else set \(M = M + 1\), \(K_1 = M\) go to step 2).

7) Initialize for another trial:
   Set \(j = i = 1\) and return to step 4.

Algorithm PFOINT:

This procedure is a supervisor program for the
factorization of polynomials over a finite field. Input
is the univariate polynomial \(U\) and prime number \(p\).
Output is an array \(Z\) containing the irreducible polynomials
\(Z_1,Z_2,\ldots,Z_m\) over the GF[\(p\)] such that
\[ UI = Z_1 * Z_2 * \ldots * Z_m \pmod{p} \]
where \(m\) is the number of irreducible polynomials.
1) Compute $x^{p^i}$ [modulo UI]:

Call CPBQ (UI, p) to compute

$q_i ≡ x^{p^i}$ [modulo UI] for $i=0$ to degree $(UI)-1$.

2) Construct the Q matrix:

Place the coefficient of polynomial $q_i$ in the

ith row of the matrix Q for $i=0$ to degree

$(UI)-1$.

3) Compute the independent vector:

Call NULLSP(Q, p) to compute the independent vectors from which the corresponding factors $V$ can be constructed.

4) Compute the irreducible polynomials:

Call BRLKPF(UI, V, p, r), where r is the number of factors of independent vectors $V$, to obtain irreducible polynomials $Z$ over $GF[p]$ such that

$UI = Z_1*Z_2*...*Z_n$ [mod p]

where $m=r$.

Algorithm CPBQ:

Input is UI, the univariate polynomial over $GF[p]$, p prime number. Output is Q array of polynomials such that

$Q_i = x^{p^i}$ [modulo UI]
1) Initialization:
Set \( K = \lfloor \log_2 p \rfloor \), \( L=2^k \), \( M=P-L \), \( B=x \),
\( j=\deg (UI) \)
where \( \lfloor \log_2 p \rfloor \) is the greater integer less than or equal to \( \log_2 p \).

2) Compute \( x^p \):
Set \( B = \rem(B^2,UI)p \) If \( M<L \) go to step 3, else \( M=M-L \), \( B = \rem(x\cdot B,UI)p \)

3) Test for iterating on \( L \):
Set \( L = L/2 \). If \( L \nmid 0 \) return to step 2.

4) Compute \( x^{p^i} \):
Set \( C=1 \), \( Q_1=1 \) and for \( i=2, \ldots, j \) do:
\( C = \rem(B\cdot C,UI)p \),
\( Q_i = C \)
continue.

When computing the function \( \rem \), calculations are performed modulo \( p \).

Algorithm NULLSP: [KNU 69]
Input is the Q matrix obtain from procedure CPBQ, \( p \) prime number. Output is \( V \) an array of polynomials computed from the independent vectors while \( r \) is the number of independent vectors.

1) Initialization:
Set vector \( C = -1 \).
Set \( r=1 \), \( V_1=1 \) and \( n=\text{matrix order} \)
Set \( k = 1 \). for \( i=1 \) to \( M \), set \( q_{i,i} = q_{i,i-1} \)

2) Scan the \( k \)th row of matrix \( Q \) for dependence:
   If there is some \( j \) in the range \( 0 \leq j \leq n \) such that \( q_{k,j} \neq 0 \) and \( C_j < 0 \), then do;
   Multiply column \( j \) by \(-1/q_{k,j}\)
   Add \( q_{k,i} \) times column \( j \) to column \( i \) for all \( j \).
   Set \( C_j = k \), \( k=k+1 \)
   If \( k>n \) end; else repeat this step; else,

3) Compute polynomials from independent vector:
Set \( r = r+1 \)
For \( j=1,\ldots,n \) construct vector \( B \) such that

\[
B_j = \begin{cases} 
1 & \text{if } j=k \\
q_{k,s} & \text{if } C_s=j>1 \\
0 & \text{otherwise}
\end{cases}
\]

Set \( V_r = \sum_{l=0}^{n-1} B_{l+1} x^l \)
Set \( k = k+1 \)
If \( k>n \) then end, else return to step 2.

Algorithm BRLKPF:

Input is \( U \) the univariate polynomial over \( GF[p] \),
\( V \) an array containing polynomial factors computed from
the procedure NULLSP, \( p \) prime number and \( r \) the number of
factors. Output is \( T \) a vector containing the irreducible
polynomials $T_1, T_2, \ldots, T_r$ such that

$$UI = T_1 * T_2 * \ldots * T_r \quad \text{[modulo p]}$$

1) Initialization:
Set vector $S = T = 0$, 
\(i = 1, S_1 = UI, k = m = 0\)

2) Employ another factor from $V$:
\(i = i + 1, VI = V_1\)

3) Employ another polynomial from $S$:
Set $k = k + 1$ 
For $j = 0$ to $p-1$ do:
If degree $(S_k) = 0$ go to step 4,
else $G = \text{GCD}(VI-j, S_k)_p$
(this operation being performed modulo $p$)
If degree $(G) = 0$ continue do loop
If degree $(G) = \text{degree} (S_k)$ go to step 4,
else place $G$ in vector $T$ in proper position depending upon its proper degree. (Call procedure ORDPOL)
Set $m = m + 1$, $S_k = (S_k/G)_p$
If $m = r$ go to step 6,
If $S_k = 0$ go to step 5 else continue do loop

4) Push the remainder onto the stack:
Set $S_k$ in $T$ (call procedure ORDPOL)

5) Test stack $S$ before trying another factor:
If there is an element in vector $S$ not used
in step 3, return to step 3; else set \( S = T \) and return to step 2.

6) Terminate the algorithm:
Place \( S_k \) and any element in the vector \( S \) not employed in step 3 in vector \( T \) and end.

In step 3 we have employed a procedure called \( \text{CPGCDl} \) to obtain the monic greatest common divisor of \( A \) and \( B \) over \( GF[p] \), that is \( \text{GCD}(VI-j, S_k)_p \).

Algorithm \( \text{CPGCDl} \):

Input is the prime number \( p \), two univariate polynomials \( A \) and \( B \) over \( GF[p] \). Output is a univariate polynomial \( C \) which is the monic greatest common divisor of \( A \) and \( B \) over \( GF[p] \).

1) Compute the remainder:
Set \( R = \text{rem}(A, B)_p \)
\( r_1 = \text{denominator} \ (R) \)
where \( r_1 \in I \)

2) Modify the remainder \( R \) to be in \( GF[p] \):
Set \( C = R \ r_1^{-1} \ [\text{modulo} \ p] \)
\( A = B, B = C \)

3) Test to terminate the algorithm:
If \( B \neq 0 \), return to step 1); else
\( A = A*(\text{ldc} \ A)^{-1} \ [\text{modulo} \ p] \)
and end.
Algorithm CBOUND:

Input is the univariate polynomial $UI$ over $\mathbb{I}[x]$ of degree $m$ and the prime number $q$. Output is the integer $j$ from which the modulus $q^{2^j}$ can be computed.

1) Search for the maximum absolute coefficient of $UI$:
Set $X_{\text{MAX}} = \text{MAX}(C, m+1)$ where $\text{MAX}$ is a function that searches the coefficient vector $C$ of $UI$ for the maximum absolute value. Set $j=1$.

2) Compute modulus $q^{2^j}$:
If $3 \cdot \text{lcd}(UI) \cdot X_{\text{MAX}} < q^{2^j}$ then end; else set $j = j+1$ and repeat this step.

Algorithm PFCI:

Input is $UI$ univariate polynomial, $p$ prime number, modulus $PQ$ equal to $p^{2^j}$, $G$ an array of polynomials over $\mathbb{GF}[p]$ such that $UI = G_1 \ast G_2 \ast \ldots \ast G_T$ [modulo $p$] where $T$ is the number of irreducible polynomials. Output is the vector $F$ containing polynomials $F_1, F_2, \ldots, F_T$ such that

$$UI = F_1 \ast F_2 \ast \ldots \ast F_T$$ [modulo $PQ$]

1) Initialization:
Set $U_p = UI$ [modulo $p$]
For $l=1$ to $T$ do:
$$G_p(l) = G(l)$$ [modulo $p$]
2) Iterate over index i:
   For i=1 to T-1, do:
   \[ B_p = \text{rem}(U_p, G_p(i)) \]
   Call PEGCDX(\( G_p(i), B_p \)) to compute \( S_p \) and \( T_p \)
   such that
   \[ G_p(i)S_p + B_pT_p = 1 \quad [\text{modulo } p] \]

3) Zassenhaus algorithm:
   Call PFH1(\( U_I, p, PQ, G_p(i), B_p, S_p, T_p \))
   to compute \( A, B \) such that
   \[ U_p = A \cdot B \quad [\text{modulo } PQ] \]
   Set \( F_i = A, U_I = B, C_p = B_p \) and continue to iterate over the index \( i \).

4) Terminate algorithm:
   Set \( F_T = U_I \cdot (\text{ldc } U_I)^{-1} \) \[\text{modulo } PQ\]
   and end.

Algorithm PFH1: [MUS 71, pp.130]

This procedure is the Zassenhaus algorithm described earlier. Input is \( U_I \) the univariate polynomial, \( p \) prime number, \( PQ \) the modulus \( p^{2j} \), \( G_p(i) \) and \( B_p \) univariate polynomials such that

\[ UI = G_p(i)B_p \quad [\text{modulo } p] \]

\( S_p, T_p \) univariate polynomials such that
\[
G_p(i)S_p + B_pT_p = 1 \quad \text{[modulo PQ]}
\]

Output are two univariate polynomials \(A\) and \(B\) over \(GF[PQ]\) such that

\[
UI \equiv A \cdot B \quad \text{[modulo PQ]}
\]

1) Initialization:
Let \(G_p(i), B_p, S_p, T_p\) be polynomials over \(GF[p]\) using the ALTRAN function MREDPO.
Set \(Q = p\)

2) Test to terminate the algorithm:
If \(Q = PQ\) then end

3) Compute polynomials \(Y\) and \(Z\). These will be used in Hensel's Lemma:
Set \(W = \frac{(UI - G_p(i)B_p)}{Q}\)
If \(Q^2 < PQ\) call procedure PSEQT\((Q,G_p(i), B_p, S_p, T_p, W)\) to compute \(Y\) and \(Z\) such that
\[
W = G_p(i)Y + B_pZ \quad \text{[modulo Q]}
\]
then go to step 4);
else \(QT = PQ/Q\) and set \(AT, BT, ST, TT\) to \(G_p(i), B_p, S_p, T_p\) [modulo QT] respectively. Here again the ALTRAN function MREDPO is employed.
Call PSEQT\((QT,AT,BT,ST,TT,W)\) to compute \(Y\) and \(Z\) such that
\[
W = ATY + BTZ \quad \text{[modulo QT]}
\]
4) Hensel's Lemma:

Compute $A_s$ and $B_s$ such that

$$A_s = QZ + G_p(i)$$

$$B_s = QY + B_p$$

If $Q^2 > PQ$ then end.

5) Recompute $S_p$ and $T_p$:

Set $TM = (A_s S_p + B_s T_p - 1)/Q$

Call PSEQT ($Q, G_p(i), B_p, S_p, T_p, W$) to compute

$AT$ and $BT$ such that

$$TM = G_p(i)AT + B_p BT \pmod Q$$

Set $S_p = S_p - Q \cdot AT \pmod Q$

$T_p = T_p - Q \cdot BT \pmod Q$

Set $Q = Q^2$, $G_p(i) = A_s$, $B_p = B_s$

and return to step 2).

Algorithm EXZH:

Input is the modulus number $PQ$, $U$ the given

multivariate polynomial of degree $M$, $A$ an array of integers

obtained from VSUBT, $H$ an array of polynomials $H_1, H_2, \ldots, H_{ir}$

such that

$$U(x_1, a_2, \ldots, a_n) = H_1 * H_2 * \ldots * H_{ir},$$

$ir$ is the number of polynomials in $H$. Output is the vector

$Y$ of polynomials such that
\[ U = Y_1 Y_2 \cdots Y_{ir} \pmod{(S^j, PQ)} \]

where \( j \) is the power of the ideals of \( S \).

1) Initialization:
   
   Set \( Y = 0 \),
   
   \[ W = U(x_1, x_2 + a_2, \ldots, x_n + a_n) \pmod{PQ} \]
   
   Set \( i = 1 \)

2) Compute all multivariate factors:
   
   Set \( F = H_1, \quad G = \prod_{i=1}^{ir} H_i \pmod{PQ} \)
   
   \( M = \deg(FG, x_1) \)
   
   For \( i = 0 \) to \( (m-1) \) do:
   
   (a) Call PEGCDX(\( F, G, i \)) to compute
   
   \( \hat{\alpha}_i, \hat{\beta}_i \) such that
   
   \[ \hat{\alpha}_i F + \hat{\beta}_i G = j j \cdot x^i \]
   
   (b) Multiplying both sides by \( j j^{-1} \pmod{PQ} \)
   
   we obtain \( \alpha_i \) and \( \beta_i \) such that
   
   \[ \alpha_i F + \beta_i G = x^i \pmod{PQ} \]
   
   (c) Continue the do loop on \( i \)
   
   Set \( j = 2, \quad R_1 = (FG - W) \pmod{PQ} \)

3) Compute \( \hat{F} \) and \( \hat{G} \):
   
   Set \( W_1 = R_1 \pmod{S^j} \) using the procedure MDSRPK.
   
   Set \( \hat{F} = (F - \Sigma \beta_i W_i) \pmod{PQ} \)
\[ \hat{G} = (G-\Sigma a_i w_i) \pmod{PQ}, \]

The index over summation depending on the degree of \( W_1 \) where \( W_1 = \Sigma w_i x_1^i \)

4) Test for the termination of the two factor algorithm:
Set \( R_1 = (F\hat{G}-W) \pmod{PQ} \)
If \( R_1 \neq 0 \) or \( j < (1 + \deg U \text{ in } x_2, \ldots, x_n) \)
Set \( F = \hat{F}, \; G = \hat{G}, \; j = j + 1 \) and return to step 3);
else set \( Y_1 = \hat{F}, \; W = G, \; l = l + 1. \) If \( L \geq ir \) then set
\( Y = Y(x_1, x_2-a_2, \ldots, x_n-a_n) \) and end; else return to step 2).

Algorithm MDSRPK:
The purpose of this procedure is compute any polynomial \( \pmod{S^j} \) where \( S \) is the ideal. Input is \( R_1 \) a multivariate polynomial, and an integer \( j \). Output is a multivariate polynomial \( W_1 \) such that
\[ W_1 = R_1 \pmod{S^j} \]

1) Initialization:
Set \( W_1 = 0, \; m = \deg (R_1, x_1) \)
\( j = m \)

2) Scan the terms for the coefficient of \( x_1^j \):
Set \( F_1 = \text{coeff}(W_1, x_1^j), \; H = 0 \)
3) Test the first term of \( F_1 \):
Call procedure \( \text{EXPOWR}(F_1) \) to obtain the first term \( F_x \) after expanding and placing \( F_1 \) in canonical form. From procedure \( \text{EXPOWR} \) we obtain the list of exponents of the variable \( F_x \) as a vector \( D \).
If the sum \( \sum_{i=2}^{n} d_i \) is less than \( j \), add \( F_x \) to \( H \).
Set \( F_1 = F_1 - F_x \prod_{i=2}^{n} x_i^{d_i} \) Set \( LK = \text{maximum sum if } LK = 0 \)
If \( F_1 \neq 0 \) repeat this step.

4) Construct \( W_1 \):
\[
W_1 = W_1 + H x_1^j
\]
If \( j < 0 \) then end;
else return to step 2).

The purpose of procedure \( \text{EXPOWR} \) is to place the function \( F_1 \) in canonical form after expansion such that values of the exponents of the variables in the first term can be obtained.

For example, let
\[
F_1 = x_2^3 x_3^4 x_4^2 + x_2^3 x_3^5 x_4 + x_2^2 x_3^7 x_4^5
\]
after expansion. Placing \( F_1 \) in canonical form,
\[
F_1 = x_2^3 x_3^5 x_4^1 + x_2^3 x_3^4 x_4^2 + x_2^2 x_3^7 x_4^5
\]
READ U
U is Multivariate polynomial

I/P U (x_1, x_2, ..., x_n)
VSUBT
O/P (a_2, a_3, ..., a_n), UI (x_1, a_2, ...)

choose the Prime number P

I/P UI, P
PPOINT
O/P Z such that UI = Z_1 * Z_2 * ... Z_T [MOD p]

I/P UI, P
CBOUND
O/P PQ = p^{2^J}

I/P UI, P, PQ, Z
PFCI
O/P F such that UI = F_1 * F_2 * ... F_T [MOD PQ]

I/P UI, PQ, h=1, F
TRUFAC
O/P H such that UI = H_1 * H_2 * ... H_r

I/P PQ, H, U
EXZH
O/P h, Y such that U = Y_1 * Y_2 * ... Y_r [MOD S^h]

I/P PQ, h, Y
TRUFAC
O/P FAC such that U = FAC_1 * FAC_2 * ... FAC_L

WRITE FAC

Figure 3.1 Multivariate polynomial Factorization algorithm.
First term will be $x_2^3x_3^5x_4^1$ while the second and third terms will be $x_2^3x_3^4x_4^2$ and $x_2^2x_3^7x_4^5$ respectively.

Algorithm EXPOWR:

Input will be $F_1$, multivariate polynomial in $(n-1)$ variables $x_2, x_3, \ldots, x_n$. Output is $F_x$, the first term of the polynomial $F_1$ after expanding and placing it in canonical form. $D$ is a vector containing the values of exponents of the variables $x_2, x_3, \ldots, x_n$ in $F_x$.

1) Initialization:
Set $D = 0$, $i = 2$

2) Compute the first term of $F_1$ after placing it in canonical form:
Set $D_i = \text{degree } (F_1, x_i)$,

$F_1 = \text{lcd } (F_1)$ with respect to $x_i$.
If $F_1 \neq 0$, set $i = i + 1$
If $i \leq n$ (n being the number of variables)
repeat this step.
Else set $F_x = F_1$ and end.

Procedure TRUFAC [WAN 73] obtains the true factors of the given multivariate polynomial $U$ over $I[x_1, x_2, \ldots, x_n]$ after computing the factors in modulo $(S_j, PQ)$. 
Algorithm TRUFAC:

Input is the given multivariate polynomial $U$, $j$ is the power of the ideal $S$, $Y$ is a vector of polynomials such that

$$U = Y_1^j Y_2^j \ldots Y_{ir}^j \quad \text{[modulo $(PQ, S^j)$]}$$

$PQ$ being the modulus number, $ir$ is the number of irreducible polynomials in vector $Y$. Output is $FAC$, a vector containing the multivariate polynomials such that

$$U = FAC_1^i FAC_2^i \ldots FAC_{iT}^i$$

where $iT \leq ir$

1) Obtain the direct true factors:
   For $i = 1$ to $ir$ do:
   Set $US = \text{ldc}(U, x_1) \cdot U$,
   $$Z = \prod_{l=1}^{ir} \text{ldc}(Y_l, x_1) \quad \text{[modulo $(S^j, PQ)$]}$$
   If the $\text{rem}(US, Z) = 0$, place $pp(Z)$ on the list $FAC$,
   Set $U = U/pp(z)$ and continue executing the loop;
   else set $Y_i$ on a vector $L$ and continue

2) Test for special case and initialization:
   At this point we have two vectors, $L$ and $FAC$.
   If $L$ is an empty vector, then end.
If $L$ contains less than four elements, place $U$ on vector $FAC$ and end;
else set $M=1$, $r =$ to a number of nonzero elements in vector $L$, $u_1 = \text{degree}(U,x_1)/2$, $US = U \cdot \text{idc}(U,x_1)$

3) Increase by one the combination of polynomials in one true factor: Set $M = M+1$

4) Test for termination:
   If $U=1$ then end.
   If $M \neq r-1$ or $M > u_1$ place $U$ on vector $FAC$ and end

5) Select combination of polynomials:
   Call procedure $LLIST$ to obtain $E$
   a multiplication of $M$ polynomials chosen from vector $L$ with their degree not exceeding $u_1$.
   Also we obtain $EE$ the multiplication of the leading coefficients of the remaining $(r-m)$ polynomials in $L$. If $E=0$ then there are no combinations that can be found. Thus place $U$ on the vector $FAC$ and end.
   If all combinations of $M$ polynomials from $L$ have been chosen, return to step 3).

6) Test the combination chosen from vector $L$:
   Set $Z = E \cdot EE \pmod{(PQ,S^j)}$
If \(\text{rem}(\text{US},Z) = 0\) place \(\text{pp}(Z)\) on the vector \(\text{FAC}\) and delete all the polynomials that are used to construct \(E\) from vector \(L\), set \(U = U/\text{pp}(Z)\), 
\[u_1 = \frac{\deg(U, x_1)}{2},\]
\[r = r - m, U = U \cdot \text{lcc}(\text{pp}(U), x_1)\]
Delete from vector \(L\) any polynomial with degree greater than \(u_1\).
Return to step 4);
else return to step 5) to select an alternate combination.

The purpose of procedure \(\text{LLIST}\) is to choose \(m\) polynomials from a vector \(L\) containing \(r\) polynomials. The degree of the multiplication of \(m\) polynomials with respect to \(x_1\) must be less than \(u_1\).

Algorithm \(\text{LLIST}\):

Input is the integer \(u_1\), a vector \(L\) containing \(r\) factors and \(m\) the number of combinations required to construct one true factor. There is an external integer \(\text{IH}\) used to indicate all possible alternative combinations of \(m\) out of \(r\) factors.

Output \(E\) is the multiplication of \(m\) polynomials chosen from vector \(L\) in which its degree is less than \(u_1/2\) and \(EE\) the multiplication of the leading coefficient of the remaining \((r - m)\) polynomials in the vector \(L\).
1) Initialization:
Set $E = 0$, $EE = 1$
If vector $C \neq 0$ (C being declared as an external array) go to step 2);
else set $c_i = i$ for $i = 1$ to $m$. Go to step 3).

2) Compute the indices of the $m$ polynomials:
Call procedure XPOINT ($m, ir, m$) to compute the indices of the $m$ polynomials and place these indices in vector $C$.

3) Test the $m$ combination:
Set $E =$ multiplication of the $m$ polynomials
If degree $(E, x_1) < u_{1/2}$, $j = 1$ and go to step 4);
else set $IH = IH - 1$.
If $IH = 0$ then end;
else go to step 2) to obtain an alternate combination of indices to compute $E$ in step 3).

4) Compute $EE$:
If $j \notin$ any one of indices of the $m$ polynomials of $L$ that construct $E$ then set
$$EE = EE \cdot ldc(L_j, x_1)$$
Set $j = j + 1$
If $j > ir$ then end;
else repeat this step.

The last procedure XPOINT is a recursive procedure
that allows each of indices (elements) of vector C to point to one of polynomials in vector L.

Algorithm XPOINT:

Input is the integer m which indicates to a pointer that its value is to be changed. ir is the number of factors in vector L and M is the number of polynomials required in one true factor.

Output is the vector C (declared as an external array) having the values of its elements recomputed.

1) Test on vector C:
   For j = 1 to M do:
   If \( C_j \neq (ir-m+j) \) go to step 2);
   else continue for loop.

2) Test for pointer \( m_1 \):
   If \( C_{m_1} \neq ir-m+m_1 \), set \( C_{m_1} = C_{m_1} + 1 \)
   and end.

3) Change pointer \( m_1-1 \):
   Call procedure XPOINT \((m_1-1, ir, m)\)
   to change pointer \( C_{m_1-1} \).
   Set \( C_{m_1} = C_{m_1-1} + 1 \)
   and end.
For example, consider

\[ L = ((x_1^2 + 5x_1 + 3), (4x_1^5 + 3x_1^2 + 2), (3x_1^3 + 4), (2x_1^2 + 7x + 3), (5x_1^3 + 7x_1 + 4)) \]

where we will use these polynomials to construct the true factor terms. While in this example \(ir=5\), the number of combinations required to construct one true factor is chosen to be equal to 3. Let \(u_1 = 15/2\), where the value 15 comes from the degree \(U\) where we have divided \(U\) by all direct true factors. The external integer \(IH = C(ir, m)=10\) is all the possible combinations of having \(m\) factors out of \(ir\) polynomials. After calling procedure LLIST, the vector \(C\) is initially zero. Thus, \(c_1=1, c_2=2, c_3=3\) and now the combination for the new factor \(E\) is formulated. First \(E = L_1 \cdot L_2 \cdot L_3\) where \(\deg E = 10\). Since \(\deg(E) > u_1\), we attempt to apply another combination of \(m\) elements of \(L\) out of \(ir\) factors. Set \(IH=9\) and call procedure XPOINT \((3, 5, 3)\) to obtain the new indices for vector \(C\). These indices are \(c_1=1, c_2=2, c_3=4\). Compute \(E = L_1 \cdot L_2 \cdot L_4\) where \(\deg(E) = 9 > u_1\). Set \(IH=8\) and again call procedure XPOINT \((3, 5, 3)\). We return from this procedure with new indices for vector \(c\), \(c_1=1, c_2=2, c_3=5\). Again we compute \(E = L_1 \cdot L_2 \cdot L_5\) with \(\deg(E) = 10 > u_1\). Set \(IH=7\), call the procedure XPOINT \((3, 5, 3)\) for a third time. In executing XPOINT, \(c_3=5\). But we have tested for possible combinations
(1,2,3), (1,2,4) and (1,2,5). Thus we must change pointer $c_2=3$ and $c_3=4$ such that we have the combination (1,3,4).
This is of course performed recursively. Returning from XPOINT, we compute $E = L_1 \cdot L_3 \cdot L_4$ where $\deg(E) = 7 < u_1$.

In LLIST we compute $EE = [ldc(L_2,x_1) \cdot ldc(L_5,x_1)] = 20$
and return to procedure TRUFA.

In TRUFA we will test to see if the given combination $L_1 \cdot L_3 \cdot L_4$ can lead us to a true factor.
Example 1

**INPUT POLYNOMIAL**

**U**

\[ x(1)^3 + x(2)^3 \]

**OUTPUT IS THE FACTORS OF THE INPUT POLYNOMIAL SUCH THAT**

**U=F(1) \& F(2) \& \ldots \& F(J)**

**F(1)**

\[ x(1) \times x(2) \]

**F(2)**

\[ x(1)^2 - x(1)x(2) + x(2)^2 \]

Example 2

**INPUT POLYNOMIAL**

**U**

\[
\begin{align*}
x(1)^4 & - x(1)^3x(3) + 3x(1)^2x(2)^2 + x(1)x(2)x(3) - 3x(1)^2x(2)x(3) - 13x(1)^2 + x(1)x(2)^2 + 15x(1)x(2)x(3) + 15x(1)x(2)x(3) + 6x(1) - x(2)x(3) - 2x(2)^2 - 15x(2)x(3) = 30
\end{align*}
\]

**OUTPUT IS THE FACTORS OF THE INPUT POLYNOMIAL SUCH THAT**

**U=F(1) \& F(2) \& \ldots \& F(J)**

**F(1)**

\[ x(1)^2 - x(1)x(3) + x(2)x(3) + 2 \]

**F(2)**

\[ x(1)^2 + 3x(1) - x(2)^2 = 15 \]

Figure 3.2(a) Example on Multivariate Polynomial factorization
Example 3

**INPUT POLYNOMIAL**

U

\[ x(1)\times x(4) + x(1)\times x(2)\times x(3) + x(1)\times x(2) + x(1)\times x(4) + x(1)\times x(3)\times x(5) + x(1)\times x(2)\times x(3) + x(1)\times x(2)\times x(5) + x(1)\times x(3)\times x(4) + x(1)\times x(3)\times x(5) + x(1)\times x(2) \times x(3) \times x(4) + x(1)\times x(2) \times x(3) \times x(5) + x(1)\times x(2) \times x(4) \times x(5) + x(2) \times x(3) \times x(4) \times x(5) + x(2) \times x(3) \times x(4) \times x(5) + x(3) \times x(4) \times x(5) \]

**OUTPUT** is the factors of the input polynomial such that

\[ F(1) = x(1) \times x(5) \]
\[ F(2) = x(1) \times x(3) \times x(4) \]
\[ F(3) = x(1) \times x(2) \times x(3) \times x(4) \]

Example 4

**INPUT POLYNOMIAL**

\[ U = x(1) \times x(4) + x(1) \times x(2) \times x(3) + x(1) \times x(2) + x(1) \times x(4) + x(1) \times x(3)\times x(5) + x(1)\times x(2)\times x(3) + x(1)\times x(2)\times x(5) + x(1)\times x(3)\times x(4) + x(1)\times x(3)\times x(5) + x(1)\times x(2) \times x(3) \times x(4) + x(1)\times x(2) \times x(3) \times x(5) + x(1)\times x(2) \times x(4) \times x(5) + x(2) \times x(3) \times x(4) \times x(5) + x(2) \times x(3) \times x(4) \times x(5) + x(3) \times x(4) \times x(5) \]

**OUTPUT** is the factors of the input polynomial such that

\[ U = F(1) \times F(2) \times \cdots \times F(J) \]

\[ F(1) = x(1) \times x(4) \]
\[ F(2) = x(1) \times x(3) \]
\[ F(3) = x(1) \times x(2) \times x(3) \]

Figure 3.2(b) Example on multivariate polynomial

Factorization
CHAPTER 4
INTEGRATION OF TRANSCENDENTAL PART

In this chapter we will briefly discuss the integration of the transcendental part of the rational function integration performed over \( I[x_1, x_2, \ldots, x_n] \) field. The implementation of this discussion will also be given.

4.1 Introduction to the Basic Problem of Integrating the Transcendental Part

As a result of section 2.2 the integration of a regular rational function \( Q \) can be given

\[
\int Q(x)dx = R(x) + \int S(x)dx
\]

where it has been shown that

\[
\int S(x)dx = \int S_0(x)/T(x)dx
\]

is purely transcendental and where \( R(x), S_0(x), T(x) \in F[x] \), \( T(x) \) being a square free polynomial and degree \((S_0(x))<\text{degree}(T(x))\). [HAR 12]

The integration of the transcendental part can be obtained explicitly using only ring operations in \( F[x] \) if and only if the following relation holds:

\[
S_0(x) = c \frac{dT(x)}{dx} \quad (4.1)
\]

where \( c \) is a constant. The proof due to Tobey [TOB 67,
\[ \int S(x)dx = \int S_0(x)/T(x)dx = \sum_{i=1}^{t} c_i \log V_i \quad (4.2) \]

where the \( V_i \) are distinct irreducible polynomials in \( F(x) \) and \( c_i \in F \). Differentiating both sides of equation (4.2) we have

\[ S_0(x)/T(x) = \sum_{i=1}^{t} c_i \frac{1}{V_i} \frac{dV_i}{dx} \quad (4.3) \]

or

\[ S_0(x) \prod_{i=1}^{t} V_i = T(x) \sum_{i=1}^{t} c_i \frac{dV_i}{dx} \prod_{j=1}^{t} V_j \]

Since \( V_i \) is relatively prime to

\[ c_i \frac{dV_i}{dx} \prod_{j=1}^{t} V_j, \]

then \( V_i \) divides \( T(x) \) for all value of \( i \). But, \( T(x) \) divides

\[ S_0(x) \prod_{i=1}^{t} V_i \quad \text{and} \quad \gcd \left( T(x), S_0(x) \right) = 1. \]

Hence \( T(x) \) divides \( \prod_{i=1}^{t} V_i \), which can be written as

\[ T(x) = k \prod_{i=1}^{t} V_i \]

where \( k \) is a constant. Now if \( t=1 \), which is the case when \( T(x) \) is an irreducible polynomial over \( F[x] \), then

\[ T(x) = k V_1(x) \]
Substituting this into equation (4.3), we obtain

\[ S_0(x) \cdot V_1(x) = k \cdot V_1(x) \cdot c_1 \cdot \frac{dV_1}{dx} \]

or

\[ S_0(x) = k \cdot c_1 \cdot \frac{dV_1}{dx} = c_1 \cdot \frac{dT}{dx} \]

With this test we are able to integrate the transcendental part using limited precision rational field arithmetic.

4.2 **Algebraic Extension Field K of F**

Any field \( K \) which contains field \( F \) as a subfield is an extension of \( F \). If \( a_1, a_2, \ldots, a_n \) are elements of \( K \), then \( F[a_1, a_2, \ldots, a_n] \) is the set of elements in \( K \) which can be expressed as the quotients of polynomials in \( a_1, a_2, \ldots, a_n \) with coefficients in \( F \). If \( a_i \in K \) is a root of polynomial \( U(x) \in F[x] \), then \( a_i \) is algebraic with respect to the field \( F \). Kronecker [ART 59] proved that there exist an extension field \( K \) of field \( F \) in which a polynomial \( U(x) \in F[x] \) with roots \( a_1, a_2, \ldots, a_n \) can be completely factored, if not in \( F[x] \), in some \( K[x] \) where \( F<K<F[a_1, a_2, a_3, \ldots, a_n] \). Both the Kronecker theorem and equation (4.1) indicate the nature of the minimum algebraic extension field of \( F \) within which the transcendental part of the integral \( S(x) \) may be computed. However a
theoretical difficulty remains in determining the optimal extension of the field $R$ in which the transcendental part $S(x)$ of $F[x]$ may be computed.

An example due to Tobey [TOB 67] presents us with the nature of the problem. Consider

$$S(x) = \frac{7x^{13} + 10x^8 + 4x^7 - 7x^6 - 4x^5 - 4x^2 + 3x + 3}{x^{14} - 2x^8 - 2x^7 - 2x^4 - 2x^3 - x^2 + 2x + 1}$$

where the integration of $S(x)$ over $R[x]$ rational field is unobtainable using test discussed earlier. However in $R(\sqrt{2})[x]$,

$$S(x) = \frac{1 - \sqrt{2} (7x^6 + 2\sqrt{2}x + \sqrt{2} - 1)}{2 (x^7 + \sqrt{2}x^2 + (\sqrt{2} - 1)x - 1)} + 1 + \sqrt{2} \frac{(7x^6 - 2\sqrt{2}x - \sqrt{2} - 1)}{2 (x^7 - \sqrt{2}x^2 - (\sqrt{2} + 1)x - 1)}$$

and

$$\int S(x) \, dx = \frac{1 - \sqrt{2}}{2} \log(x^7 + \sqrt{2}x^2 + (\sqrt{2} - 1)x - 1) + \frac{1 + \sqrt{2}}{2} \log(x^7 - \sqrt{2}x^2 - (\sqrt{2} + 1)x - 1) + \text{constant}$$

$$= \frac{1}{2} \log(x^{14} - 2x^8 - 2x^7 - 2x^4 - 4x^3 - x^2 + 2x + 1) + \frac{1}{\sqrt{2}} \log\left(\frac{x^7 - \sqrt{2}x^2 - (\sqrt{2} + 1)x - 1}{x^7 + \sqrt{2}x^2 + (\sqrt{2} - 1)x - 1}\right) + \text{constant}$$

While the fact that $\int S(x) \, dx$ can be calculated over
R(\sqrt{2})[x], it is not clear how one can obtain the monic irreducible polynomial \( x^2 - 2 \) as the one to determine the algebraic extension field.

While Tobey has discussed the mathematical techniques of extending the integration of the transcendental part over the extended rational field, he did not describe any precise algorithms for integration of the transcendental part over an optimal extended field \( R[\alpha] \) where \( \alpha \) is an element of the irrational number set. An attempt has been made in this project to deal with only a part of this problem. This is to be discussed in the next section.

4.3 Nature of the Problem Solved in ALTRAN

Because ALTRAN is incapable of performing operations over irrational arithmetic, that is square roots, cubes roots, etc. of integers, we are constrained to perform the integration of the transcendental part \( S(x) = S_0(x)/T(x) \) in the following ways:

a) \( S_0(x) = k \frac{dT(x)}{dx} \) where \( k \) is a constant.

Then,

\[ \int S(x)dx = k \log T(x) \]

b) \( T(x) \) is a polynomial of second order \( (ax^2+bx+c) \). Then
\[ \int S(x)\,dx = \int \frac{a_0 x + b_0}{a_1 x^2 + b_1 x + c_1} \, dx \]

\[ = \int \frac{a_0 (2a_1 x + b_1)}{2a_1 (a_1 x^2 + b_1 x + c_1)} \, dx \]

\[ + \int \frac{a_0 b_1}{a_1 x^2 + b_1 x + c_1} \, dx \]

\[ = \frac{a_0}{2a_1} \log(a_1 x^2 + b_1 x + c_1) \]

\[ + \left( \frac{2a_1 b_0 - a_0 b_1}{2a_1} \right) \int \frac{dx}{(x + \frac{b_1}{2a_1})^2 + \left( \frac{c_1}{a_1} - \frac{b_1^2}{4a_1^2} \right)} \]

\[ = \frac{a_0}{2a_1} \log(a_1 x^2 + b_1 x + c_1) \]

\[ + \frac{2a_1 b_0 - a_0 b_1}{a_1 \sqrt{4a_1 c_1 - b_1^2}} \arctan^{-1} \]

\[ \frac{(2a_1 x + b_1)}{\sqrt{4a_1 c_1 - b_1^2}} \quad (4.5) \]

Special format statements have been used in ALTRAN to represent the arguments of log, square root and \( \arctan^{-1} \) functions.
4.4 Implementation

Consider $S(x) = S_0(x)/T(x)$ where $T(x)$ is a square free polynomial. Making use of Wang's algorithm [WAN 73] described in Chapter 3, we perform the factorization of the polynomial $T(x)$ over the integers such that

$$T(x) = \left( \prod_{i=1}^{n_1} V_{1,i} \right) \left( \prod_{i=1}^{n_2} V_{2,i} \right) \left( \prod_{i=1}^{n_3} V_{3,i} \right)$$

(4.6)

where $V_{1,i}$, $V_{2,i}$ and $V_{3,i}$ are polynomials of degree one, two and higher than two respectively. In addition $V_{3,i}$ is an irreducible polynomial over the integers and if $n_j = 0$ the complete term $\left( \prod_{i=1}^{n_j} V_{j,i} \right)$ will vanish.

Using the partial fraction decomposition algorithm discussed in section 2.2, we can obtain

$$S(x) = \sum_{i=1}^{n_1} \frac{A_{1,i}}{V_{1,i}} + \sum_{i=1}^{n_2} \frac{A_{2,i}}{V_{2,i}} + \sum_{i=1}^{n_3} \frac{A_{3,i}}{V_{3,i}}$$

(4.7)

where the degree $A < \text{degree } V$ (Theorem 2.2T4).

Then,

$$\int S(x)dx = \sum_{i=1}^{n_1} \alpha_i \log V_{1,i} + \sum_{i=1}^{n_2} \beta_i \log V_{2,i}$$

$$+ \sum_{i=1}^{n_2} \gamma_i \arctan^{-1} F_i + \sum_{i=1}^{n_3} \psi_i \log Z_i$$

$$+ \int_{j=1}^{n_{3,2}} \frac{A_{3,j}}{V_{3,j}} dx$$

$$+ \int_{j=1}^{n_{3,2}} \frac{A_{3,j}}{V_{3,j}} dx$$
where $Z_i = V_{3,1}$ If $A_{3,i} = \psi_i V_{3,1}$, $C$, $\alpha_i$, $\beta_i$, $\psi_i$ are constants over the rational field while $\gamma_i$ is a constant over the irrational field, $n_3 = n_{3,1} + n_{3,2}$ then $F_i$ can be computed as shown in equation (4.5).

The first step is call the supervisor procedure MVFOI to perform the factorization of the denominator of the transcendental part over the integers. It is clear from previous discussions that the factored polynomial has only a multiplicity of one such that

$$T(x) = T_1(x) T_2(x), \ldots, T_r(x)$$

In the second step we call procedure PFDEC to compute the partial fraction decomposition. It is in this procedure that we have a modified set of instructions similar to procedure MATSFD. In this procedure we divide $T(x)/T_i(x)$ instead of $T_i(x)^i$. It is in this procedure that we compute the coefficient matrix for the polynomials $A_{j,i}$, $j=1,2,3$ followed by solving a system of linear equations to obtain a vector $A$ such that

$$S_0(x)/T(x) = \sum_{i=1}^{r} \frac{A_i(x)}{T_i(x)}$$

where degree $(A_i)$<degree $(T_i)$.

In the third step we perform integration using a pattern matching procedure.
This procedure is called TRPT and performs a test to indicate if it is possible to integrate the given transcendental part over the rationals using equations (4.1) and (4.5). The last equation enables us to obtain the \( \arctan^{-1} \) term over the irrationals.

Algorithm TRPT:

Input is the transcendental part \( A_i/T_i \) where \( T_i \) is the irreducible polynomial over \( \mathbb{I} \). Output is the coefficients \( C_{01} \) for the logarithmic term, \( C_{02} \) for the inverse arctan function, arguments \( X_{LN} \) of the logarithmic function, \( X_{ART} \) of the inverse arctan function and \( Z \) and integer. In practice \( C_{02} \) and \( X_{ART} \) are divided by the square root of \( Z \).

1) Initialization:
Set \( Z = 0, \ C_{01} = 0, \ C_{02} = 0, \ X_{LN} = 0, \ X_{ART} = 0 \)

2) Apply the test \( C \, \text{dt/dx}: \)
If \( \deg \left( \frac{A_i}{(dT_i/dx)} \right) \neq 0 \)
then go to step 4).

3) Compute the coefficient and argument of the logarithmic function:
Set \( C_{01} = \frac{A_i}{(dT_i/dx)} \)
\( X_{LN} = T_i \)
then end.
4) Compute the coefficient and argument of the inverse arctan function:
If deg \((T_i)>2\) then end;
else set \(M = \text{lde}(a_i)\),
\(N_1 = \text{coeff}(A_i,x^0)\)
\(C = \text{lde}(T_i)\)
\(p = \text{coeff}(T_i,x^1)\)
\(q = \text{coeff}(T_i,x^0)\)
Set \(CO1 = M/(2\cdot C)\)
\(XLN = T_i\)
\(CO2 = (2\cdot N_1 \cdot C - p \cdot M)/C\)
\(XART = a \cdot x \cdot C + p\)
\(Z = 4 \cdot q \cdot C - p^2\),
then end.

The purpose of procedure INTRPT is to act as the supervisor for the integration of any transcendental part.

Algorithm INTRPT:
Input is the transcendental function \(S_0(x)/T(x)\)
while output is the coefficients \(COEF1\), \(COEF2\), the arguments of logarithmic and inverse arctan function XLOG and XARTN respectively, the integer XS the square root of which divides \(COEF2\) and XARTN. In addition there exist as output a vector L containing all terms not possible to integrate over the rational field.
1) Compute the irreducible polynomials:
Call procedure MVFOI \((T(x))\) to compute
\(PT_1, PT_2, \ldots, PT_r\), the irreducible terms such that
\[ T(x) = \prod_{i=1}^{r} PT_i \]
Output is represented as a vector called \(PT\).

2) Compute the partial fraction terms:
Call procedure PFDEC\((T(x), PT, S_0(x))\) to compute
\(A_i\) such that
\[ S_0(x)/T(x) = \sum_{i=1}^{r} A_i/T_i \]
Set \(i=1\).

3) Compute coefficients and arguments:
Call procedure TRPT\((A_i/T_i)\)
Obtain the terms \(C01_i, C02_i, XLOG_i, XARTN_i, XS_i\)
If \(COEF1_i\) and \(COEF2_i = 0\) then add \(A_i/T_i\) to vector \(L\). This indicates that integration is not possible over the rational field.
Set \(i=i+1\), if \(i>r\) then end; else repeat this step.

With regard to multivariate transcendental part, our definitions and theorems discussed earlier in this chapter can be extended. Our constant of integration over
the rationals in our previous equation will now be a multivariate polynomial over the rationals. For example, equation (4.1) can be modified to read

\[ S_0(x_1, x_2, \ldots, x_n) = C \times dT(x_1, x_2, \ldots, x_n)/dx_1 \]

where \( C \in R [x_2, x_3, \ldots, x_n] \), continuing in the same fashion for the other equations. In addition to the extended rational field, it can contain irrational elements from the irrational set as well as polynomials with rational powers. In this case the denominator of the transcendental part can be factored into roots whose variables are raised to rationals. For example,

\[ (x_1^3 - x_2) = (x_1 - x_2^{1/3})(x_1^2 + x_1 x_2^{1/3} + x_2^{2/3}) \]

While procedure INTRPT can perform integration of multivariate transcendental part, given that the integration will be possible over \( R[x_1, x_2, \ldots, x_n] \) with the solution being in the form of logarithmic and/or inverse arctangent functions, it is not capable of factoring the type of example shown above.
Figure 4.1 Example on Transcendental Function Integration
4.5 Conclusions

It has been shown that the ALTRAN system is capable of performing complex algebraic operations. Not only can large problems be executed, but it is capable of performing modular arithmetic operations for the purposes of multivariate factorization.

In performing the integration of rational functions to obtain the rational part, Horowitz's method has the advantage of saving execution time and storage space over Hermite's method. With regard to the rational part the method is well defined and solved. However in the case with integration of the transcendental part defined over an extended rational field, it is not completely solved. What is necessary is a symbolic algebraic system capable of performing operations over an extended rational field in addition to dealing with polynomials having rational exponents. In this regard we have found ALTRAN incapable of performing such operations.

It is possible that partial solutions for the integration of the transcendental part is defined for denominators that can be factored over the integers, their solutions being in terms of logarithmic or inverse arctangent functions.
REFERENCES


APPENDIX A

A Listing of the Program HERM
PROCEDURE HERM (AB, R, S)  

INPUT:  AB  RATIONAL FUNCTION  
OUTPUT: R  RATIONAL PART OF INTEGRAL  
S  TRANSCENDENTAL PART  

LONG ALGEBRAIC (X) AB, A, B, R, S, F, Z, Y, G  
INTEGER K, M, H, C  
ALGEBRAIC ALTRAN HERM2  
R = 0  
S = 0  
IF (AB == 0) GO TO LF  
A = ANUM (AB)  
B = ADEN (AB)  
N = DEG (A, X)  
F = DEG (B, X)  

IF THE DENOMINATOR IS A CONSTANT USE THE ALTRAN  
PROCEDURE PINT .  

IF (F == 0) GO TO L1  
R = PINT (AB, X)  
GO TO LF  

IF THE DEGREE OF THE NUMERATOR (A) IS LESS THAN THE  
DEGREE OF THE DENOMINATOR (B) CALL THE RATIONAL  
INTEGRATION PROCEDURE .  

L1^ IF (M <= N) GO TO L2  
HERM2 (AB, N, R, S)  
GO TO LF  

IF THE DEGREE OF THE NUMERATOR (A) IS GREATER THAN THE  
DEGREE OF THE DENOMINATOR (B) THEN DO DIVISION IN THE  
INTEGER DOMAIN .  
INTEGRATE THE POLYNOMIAL BY THE ALTRAN PROCEDURE PINT  
AND THE REGULAR RATIONAL PART BY THE RATIONAL INTEGRATION  
PROCEDURE .  

L2^ C = G + FPO (EXPAND (B) ; X, N) **(M-N+1)  
A = C + A  
F = C + B  
Z = AQUO (A, B, X, Y)  
Z = Z / C  
G = PINT (Z, X)  
Y = Y / F  
N = DEG (ADEN (Y), X)  
HERM2 (Y, N, R, S)  
K = R + G
PROCEDURE POCEF (C,NI,N)

INPUT A C UNIVARIATE POLYNOMIAL
NI INTEGER SUCH THAT THE POLYNOMIAL B CAN BE
EXPRESSED AS B=C*X**NI.
N THE MAXIMUM DEGREE OF B.
OUTPUT A VECTOR SUCH THAT ITS ELEMENTS ARE EQUAL TO THE
COEFFICIENTS OF B.

LONG ALGEBRAIC (X) B,C
INTEGER J,M,NI,N ,A
ARRAY (1^N) A
VALUE C,NI,N
B=C
B=B*X**NI
M=N-1
DO J=H,0,-1
A(J+1)=COEPFO(EXPAND(B),X,J)
ENDO
RETURN (A)
END
PROCEDURE HERM2(AB,N,R,S)

INPUT  AB  REGULAR RATIONAL FUNCTION
N  DEGREE OF THE DENOMINATOR
OUTPUT  R  RATIONAL PART OF INTEGRAL
S  TRANSCENDENTAL PART

LONG INTEGER WF
LONG ALGEBRAIC (X) AB,R,S,ACOM,BF,BI,RP,SP,C
INTEGER J,II,N,L
ARRAY(1:N) C
ARRAY(1:N) BF
ARRAY(1:N) WF
ARRAY(1:N,1:N) ACOM
LONG ALGEBRAIC ARRAY ALTRAN RDEC
LONG ALGEBRAIC ALTRAN HERM1
R=u
S=u
C=u

COMPUTE THE COMPLETE PARTIAL FRACTION DECOMPOSITION BY CALLING
PROCEDURE RDEC SUCH THAT
AB= ACOM(1,1)/(WF(1)*BF(1)) + ACOM(2,1)/(WF(2)*BF(2)) +
ACOM(2,2)/(WF(2)*BF(2)+2)+
+ ACOM(L,L)/(WF(L)*BF(L)+L)

RDEC(AB,N,ACOM,BF,WF,L)

PERFORM THE REDUCTION PROCEDURE FROM J=2,3,...,L
CALL PROCEDURE HERM1 TO COMPUTE RP, S SUCH THAT
RP+INTEGRAL(S)=INTEGRAL(SUM(ACOM(J,J)/BF(J)**I*WF(J)))
WHERE I=2,3,...,J

HERM1(C,BI,J,N,RP,SP)

ADD TO THE RATIONAL AND TRANSCENDENTAL PART BOTH RP,SP
RESPECTIVELY.
SET J=J+1 AND TEST IF J LESS OR EQ A L; REPEAT THE REDUCTION
PROCEDURE (F, X)

INPUT F UNIVARIATE POLYNOMIAL

OUTPUT X VARIABLE OF DIFFERENTIATION

PROCEDURE, ELSE RETURN.

R = R + R/P * W(F, J)
S = S + S * P * W(F, J)
DO END
RETURN (R, S)
END

PROCEDURE DIFFX(F, X)

INT K, L
LONG ALGEBRAIC F, X, POL, FD, G

VALUE F, X

IF (F == 0) RETURN(F)
K = DEG(F, X)
IF (K == 0) RETURN (0)
FD = ADEN((F))
G = EXPAND(ANUM(F))
POL = 0
DO L = 1, K
POL = POL + L * X ** (L-1) * COEFPC(G, X, L)
DO END
RETURN(POL/FD)
END
PROCEDURE HERMI(A, B, I, KM, R, S)

INPUT
A ARRAY CONTAINS POLYNOMIALS
B UNIVARIATE POLYNOMIAL
I THE EXPONENT OF THE POLYNOMIAL B
KM ARRAY SIZE

OUTPUT
R RATIONAL FUNCTION
S RATIONAL FUNCTION SUCH THAT
R+INTEGRAL (S)=INTEGRAL (A(1)/B(I)+A(2)/B(I)**2+...

------------------------------------------------------------------------

LONG ALGEBRAIC (X) A, B, BD, S, R, C, D, CC, DD, M1, H, RX
INTEGER J, N, I, KM
ARRAY(1:AKM) A
LONG ALGEBRAIC ALTRAN EGCD
LONG ALGEBRAIC ALTRAN PEGCD
LONG ALGEBRAIC ALTRAN DIFFX
VALUE A, B, I, KM
R=0

------------------------------------------------------------------------

SET S=A(I), N=DEG(B)+1, M=N-1, CALL PROCEDURE PEGCD TO COMPUTE C,D,RX
SUCH THAT B+C+D=RX
WHERE DEG(C)<DEG(DIFF(B)), DEG(D)<DEG(B).

------------------------------------------------------------------------

S=A(I)
N=DEG(B)+1
M=N-1
BD=DIFFX(B, X)
IF(DEG(BD, X).NE.0) GO TO LNO
C=0
D=1
RX=BD
GO TO LC

LNO PEGCD (B, BD, N, M, C, D, RX)

DO J=I, 2, -1
IF(S.EQ.0) GO TO L1
EGCD(S, B, BD, C, D, RX, CC, DD, M1)
R=R-0D/(M1+(J-1)+B**(J-1))
H=DIFFX(BD, X)+(J-1)*CC
H=H/(M1+(J-1))
S=H+A*(J-1)
CONTINUE
END

END
PROCEDURE EGCD (A,Z,Y,C,D,R,RR,SS,W)

INPUT A UNIVERIATE POLYNOMIAL
   Z UNIVERIATE POLYNOMIAL
   Y UNIVERIATE POLYNOMIAL
   C UNIVERIATE POLYNOMIAL
   D UNIVERIATE POLYNOMIAL
   R INTEGER SUCH THAT R=Z*C+Y*D
       AND DEG(C)<DEG(Y), DEG(D)<DEG(Z)

OUTPUT RR UNIVERIATE POLYNOMIAL
   SS UNIVERIATE POLYNOMIAL
   W INTEGER SUCH THAT A*W=RR*Z+SS*Y

LONG INTEGER V
INTEGER K,N
VALUE A,Z,Y,C,O,R

IF C=0 SET RR=8, SS=A, W=R AND RETURN, ELSE SET RR=A+C, SS=A*D,
   N=DEG(Y) AND M=DEG(RR)
IF N LESS THAN M SET S=R AND RETURN, ELSE SET
   RR=REMAINDER ((RR*LDU(Y)**) (M-N+1)) Y,
   SS=Z+QUOTIENT ((RR*LDU(Y)**) (M-N+1)) /Y
   W=R*LDU(Y)**) (M-N+1) AND RETURN.

IF C.NE.0 GO TO L1
RR=0
SS=A
W=R
RETURN (RR,SS,W)

L1
RR=A+C
SS=A*D
M=DEG(RR,X)
N=DEG(Y,X)
IF M.GE.N GO TO L2
W=R
RETURN (RR,SS,W)

L2
V=COEFF(IEXPAND(Y),X,N)**(M-N+1)
H=V*RR
Q=QUOTIENT(H,Y,X,RR)
SS=V*SS+Q*Z
W=V*R
RETURN (RR,SS,W)

END
PROCEDURE PEGCD(AA, BB, N, R, S, F)

INPUT AA UNIVARIATE POLYNOMIAL

OUTPUT R UNIVARIATE POLYNOMIAL

OUTPUT S UNIVARIATE POLYNOMIAL

OUTPUT F INTEGER SUCH THAT

R*AA+S*BB=F AND DEG(R)<DEG(BB), DEG(S)<DEG(AA)

LONG ALGEBRAIC (X) AA, BB, R, S, Z, ZI, C, Y, F

LONG INTEGER A, B


ALGEBRAIC ARRAY ALTRAN ATRANS

ALGEBRAIC ALTRAN ADET

LONG ALGEBRAIC ARRAY ALTRAN SOLEQ

LONG INTEGER ARRAY ALTRAN POCEF

ARRAY(1AN) AA

ARRAY(1AM) BB

ARRAY(1AMN,1AMN) Z

ARRAY(1AMN,1AMN) ZI

ARRAY(1AMN) C

ARRAY(1AMN) Y

VALUE AA, BB, N, M

IF DEG(BB)=0 SET R=J, S=1, F=BB AND RETURN ELSE

SET COEFFICIENT OF AA, BB IN VECTOR A, B RESPECTIVELY AND CONSTRUCT

MATRIX Z SUCH THAT

A(0) A(1) A(2) A(3) A(4) A(5) A(6) A(7) A(8) A(9)

B(0) B(1) B(2) B(3) B(4) B(5) B(6) B(7) B(8) B(9)

C(0) C(1) C(2) C(3) C(4) C(5) C(6) C(7) C(8) C(9)


F(0) F(1) F(2) F(3) F(4) F(5) F(6) F(7) F(8) F(9)

IF (DEG(BB,X),NE,0) GO TO L1

R=0

S=1

F=BB

RETURN (R, S, F)

L1^
Z(I+J-1,J) = A(I)
DO END
DO END
DO J=M,MN
JC=MN-J
DO I=1,M
Z(I+JC,J) = B(I)
DO END
DO END

---


SOLVE THE SYSTEM OF LINEAR EQUATIONS Z*Y = C
WHERE C IS THE CONSTANT VECTOR

---

Z1 = ATRANS(Z)
C(1) = ADET(Z1)
F = C(1)
Y = SOLLQ(Z,C,MN)
MM = MM - 1
NN = N - 2

---

CONSTRUCT THE POLYNOMIAL R AND S SUCH THAT THE FIRST N ELEMENTS IS THE COEFFICIENT OF R AND THE REMAINDERS ARE THE COEFFICIENT OF S.

---

DO J=MM,0,-1
K = R + Y(MM-J+1)*X**J
DO END
DO J=NN,0,-1
S = S + Y(MN-J)*X**J
DO END
RETURN (R,S,F)
END
PROCEDURE RDEC(AA,N,XZ,Y,W,K)

INPUT AA REGULAR RATIONAL FUNCTION (A/B)
N DEGREE OF THE DENOMINATOR
OUTPUT XZ MATRIX WITH POLYNOMIAL ELEMENT,
Y ARRAY CONTAINS THE SQUARE FREE POLYNOMIALS
W INTEGER ARRAY SUCH THAT
AA=XZ(1,1)/(W(1)*Y(1))+XZ(2,1)/(W(2)*Y(2))+
XZ(2,2)/(W(2)*Y(2)**2)+.................+
XZ(K,K)/(W(K)*Y(K)**K)
K THE EXPONENT OF THE MAXIMUM FACTOR OF B.

LONG ALGEBRAIC (X) AA,AB,XZ,ZI,XX,Y,AL,BI
LONG INTEGER W,Z,V,J
INTEGER K,N,J,I,L
ARRAY(1N)Y
ARRAY(1N)XX
ARRAY(1N)Z
ARRAY(1N)W
ARRAY(1N,1N)XZ
LONG ALGEBRAIC ARRAY ALTRAN RSQD
LONG ALGEBRAIC ARRAY ALTRAN PCDE
VALUE AA,N
AB=AA
ZI=0
W=U
XZ=O

PERFORM PARTIAL FRACTION DECOMPOSITION BY CALLING PROCEDURE
RSQDDEC SUCH THAT
AA=XX(1)/(Y(1)*Z(1))+XX(2)/(Z(2)*Y(2)**2)+XX(K)/(Z(K)*Y(K)**K)
RSQDDEC(AB,N,XX,Y,Z,K)

IF XX(I)=0 OR DEG XX(I) LESS THAN DEG Y(I) THERE IS NO FURTHER
DECOMPOSITION, ADD XX(I) TO XZ(I,1) AND SET W(I)=1, ELSE CALL
PROCEDURE PCDE TO COMPUTE THE COMPLETE PARTIAL FRACTION TERMS

DO I=1,K
AI=XX(I)
BI=Y(I)
VI=Z(I)
IF((AI.EQ.0).OR.(DEG(AI,X)<DEG(BI,X)))GO TO L2
J=IQUO(DEG(AI,X),DEG(BI,X))+1

PERFORM THE COMPLETE PARTIAL FRACTION DECOMPOSITION
FOR XX(I)/Y(I)**I SUCH THAT
XX(I)/Y(I)**I=ZI(1)/Y(I)+ZI(2)/Y(I)**2+.................+ZI(I)/Y(I)**I
v SET w(i) = zh(i) * lok(y(i))**(deg(xx(i)) - deg(y(i)) + 1)

v z(i) = pdeg(ai, bi, j, n)
DO l = 1, j
xz(i, l) = z(l)
DOEND
l = deg(ai, x) - deg(bi, x) + 1
w(i) = v(i) * cdefpo(expand(bi), x, deg(bi, x))**l
GO TO L1
l2^ xz(i, 1) = a
w(i) = 1
L1^ CONTINUE
DOEND
RETURN(xz, y, w, k)
END

PROCEDURE REP (A, B)

v INPUT ^ A UNIVARIATE POLYNOMIAL
v B UNIVARIATE POLYNOMIAL
v OUTPUT ^ J INTEGER SUCH THAT B = Z*A**J WHERE Z IS POLYNOMIAL

v LONG ALGEBRAIC(X, A, B)
INTEGER J
VALUE A, B
J = 0
L1^ IF(AGCD(A, B), EQ, 1) RETURN (J)
B = B / A
J = J + 1
GO TO L1
END
PROCEDURE RSQDEC( AA, N, XX, Y, Z, K)

INPUT AA REGULAR RATIONAL FUNCTION (A/B)
N DEGREE OF THE DENOMINATOR (B)
OUTPUT XX ARRAY CONTAINS POLYNOMIALS
Y LINEAR LIST OF SQUARE FREE FACTORS
Z ARRAY CONTAINS INTEGERS SUCH THAT
AA=XX(1)/(Z(1)*Y(1)) + XX(2)/(Z(2)*Y(2)**2) + ... + XX(K)/(Z(K)*Y(K)**K)
K THE EXPONENT OF THE MAXIMUM FACTOR OF B.

LONG ALGEBRAIC (X) AB, A, B, Y, XX, AA, E, F, G
LONG INTEGER B1, Z
INTEGER K, N, KM1, I, KD, KM, J, RN
ARRAY(1AN) XX
ARRAY(1AN) Y
ARRAY(1AN) Z
ARRAY(1AN) F
ARRAY(1AN) G
ARRAY(1AN, 1AN) E
LONG ALGEBRAIC ARRAY ALTRAN SOLEQ
LONG INTEGER ARRAY ALTRAN MATSFQ
LONG ALGEBRAIC ARRAY ALTRAN PSQFRE
LONG ALGEBRAIC ARRAY ALTRAN TNLNGTH
VALUE AA, N
AB=AA
XX=0

SET A=NUPHATOR OF AA , B1=CONTENT OF THE DENOMINATOR OF AA AND
B=PRIMATIV PART OF THE DENOMINATOR OF AA.
CALL PROCEDURE PSQFRE TO OBTAIN THE SQUARE FREE POLYNOMIALS. THEN
ORDER THESE POLYNOMIALS USING PROCEDURE TNLNGTH SUCH THAT
B=Y(1)*Y(2)**2*...*Y(K)**K
IF THERE IS NO MORE FACTORIZATION THEN SET XX(1)=A, Y(1)=B ,
Z(1)=B1 AND RETURN.

A=ANUM(AB)
B=ADEN(AB)
B1=CONT(B)
B=B/B1
Z=PSQFRE(B, N)
TLNGTH(N, B, Z, K, Y)
Z=0
IF(K<>1)GO TO L1
XX(K)=A
Y(K)=B
Z(K)=B1
RETURN(XX, Y, Z, K)
K1=K-1

L1^
IF THERE IS ONLY ONE FACTOR AND RAISED TO POWER K THEN SET

\[ XX(K) = A, Z(K) = B_1, XX(I) = 0, Z(K) = 0 \] FOR I = 1, ..., K-1 AND RETURN

DO I = 1, KM1
  IF (deg(Y(I), X) <> 0) GO TO L2
  DOEND

  XX(K) = A
  Z(K) = B_1
  XX(I) = 0
  Z(I) = 1

  RETURN(XX, Y, Z, K)

CONSTRUCT THE COEFFICIENT MATRIX USING PROCEDURE MATSFD.

CONSTRUCT THE CONSTANT VECTOR F BY PLACING THE COEFFICIENT
OF THE NUMERATOR (A) IN IT.

SOLVE SYSTEM OF LINEAR EQUATION USING PROCEDURE SOLEQ SUCH THAT
\[ E_0 = F \]

L2A
  E = MATSFD(B, Y, K, N)
  XX = POCF(A, G, N)
  DO I = 1, N
    F(I) = XX(N-I+1)
  DOEND
  G = SOLEQ(E, F, N)

CONSTRUCT XX(I), Z(I)

SET NO = 0, J = 1

XX(J) = SUM OF [G(I) * X**I] WHERE I = NO, ..., NO+N(J-1)

IF I = N-1 THEN END; ELSE NO = NO+N(J), J = J+1 AND REPEAT THIS STEP.

KM1 = 0
K0 = 1
DO I = 1, K
  M = deg(Y(I), X)
  IF (M < 0) GO TO L3

XX(I) = 0
Z(I) = 1
GOTO L4

L3A
  RN = I * M
  KM1 = KM1 + RN
  KM = KU + KN - 1
  XX(I) = 0
  DO J = KU, KM
    XX(I) = XX(I) + G(J) * X**KM-J
  DOEND
  K0 = K0 + 1
  Z(I) = B_1

L4A
  CONTINUE
  DOEND
  RETURN(XX, Y, Z, K)

END
PROCEDURE PCDEC(A, B, J, KM)

I MUT A UNIVARITATE POLYNOMIAL
B UNIVARITATE POLYNOMIAL
KM ARRAY SIZE
J INTEGER SUCH THAT J = DEG(A)/DEG(B) + 1
OUTPUT XX ARRAY CONTAINS POLYNOMIAL SUCH THAT
A/B(J)**J = SUM OF (XX(I)/B(J)**I) WHERE I =

1, 2, ..., J

LONG ALGEBRAIC (X) A, B, Q, QD, AD, XX
INTEGER M, N, KM, I, J
ARRAY(I^KM)*XX 1
VALUE A, B, KM, J
M = DEG(A, X)
N = DEG(B, X)
XX = 0
I = J

TO PERFORM COMPLETE PARTIAL FRACTION DECOMPOSITION SET
Q = A - D(C(B) ** (DEG(A) - DEG(B) + 1))
Q = Q * (EXPAND (B), X, N) ** (M - N + 1)
Q = QUOTIENT(Q/B)
XX = 0

XX(J) = REMAINDER (Q/B) AND Q = QUOTIENT(Q/B)
IF DEG(Q) LESS THAN DEG(B) SET XX(I) = Q AND RETURN,
ELSE SET J = J - 1 AND GO TO STEP L1.

L1
QD = A/QO(Q, B, X, AD)
XX(I) = AD
IF (DEG(QC, X) < N) GO TO L2
Q = QD
I = I + 1
GO TO L1

L2
XX(I) = QD
RETURN (XX)
END
PROCEDURE TLNGTH(IK,Y,KM,Z)

INPUT Y UNIVARIATE POLYNOMIAL
ARRAY CONTAINS SQUARE FREE POLYNOMIALS SUCH THAT
B=Y(1)**N(1)*Y(2)**N(2)...*Y(L)**N(L)

OUTPUT KM DEGREE OF B

THE EXPONENT OF THE MAXIMUM FACTOR OF B.

Z A LINEAR LIST OF THE SQUARE FREE POLYNOMIALS SUCH THAT
B=Z(1)**1+Z(2)**2...+Z(KM)**KM

ORDER THE POLYNOMIALS SUCH THAT POLYNOMIAL Z(I) WILL BE RAISED
TO POWER I AND PLACED IN LOCATION I IN VECTOR Z.

DO I=1,IK IF(Y(I),EQ,0)GO TO L1
KM=REP(Y(I),B)
B=B/Y(I)**KM
Z(KM)=Y(I)
DO END
L1 RETURN(KM,Z)
END
PROCEDURE CONT (BB)

INPUT ^ BB  UNIVARIATE POLYNOMIAL OVER INTEGER
OUTPUT ^ Z  CONTENT OF BB

LONG ALGEBRAIC(X) B, BB, Z
LONG INTEGER A, C, U
INTEGER M, I
VALUE BB
B=ANUM(BB)
D=ADEN(BB)
M=DEG(B, X)
IF(M<>U) GO TO L2
Z=1
Z=Z/D
RETURN (Z)

CONSTRUCT VECTOR Z SUCH THAT POLYNOMIAL B WILL BE
EQUAL TO 
B = Z(M)*X**(M-1)+Z(M-1)*X**(M-2)+.....+Z(0)
AND CONTENT OF B WILL BE EQUAL TO 
GCD(Z(M),Z(M-1),........,Z(0)).

L2^ Z=COEFPOL EXPAND(B), X, M)
M=M-1
DO I=N, U, -1
A=COEFPOL EXPAND(B), X, I)
Z=IGCD(Z, A)
C=Z
IF(C.EQ.1) GO TO LF
RETURN (Z)
END
PROCEDURE MATSFD(B,F,L,N)

INPUT
B UNIVARIATE POLYNOMIAL
F ARRAY CONTAINS THE SQUARE FREE POLYNOMIALS
L THE EXPONENT OF THE MAXIMUM FACTOR OF B
N ARRAY SIZE

OUTPUT
Z MATRIX USED TO COMPUTE THE PARTIAL FRACTION TERMS

LONG ALGEBRAIC (X)B,B,F,CI,FI,C
LONG INTEGER Z
INTEGER NOI,I,KI,IZ,J,L,N,II,J
LONG INTEGER ALTRAN REP
ARRAY(1AN)F
ARRAY(1AN,1AN)Z
ARRAY(1AN)C
VALUE B,F,L,N
NO=0

INITIALIZATION
SET I=1

DO J1=1,L
BI=F(J1)
IF(DEC(BI,X)==0) GO TO L4

SET FI=B/F(I)*I.PLACE THE COEFFICIENT OF FI IN VECTOR C THEN
SET J=2 AND C IN THE FIRST COLUMN OF THE N(I) GROUP.
CONSTRUCT THE REMAINDER OF THE N(I) COLUMNS BY SHIFTING DOWNWARD
ALL THE ELEMENTS IN VECTOR C BY ONE PLACE WHILE PLACING AN ELEMENT
OF VALUE ZERO IN THE FIRST LOCATION, ADD C TO THE MATRIX IN THE
J TH COLUMN OF THE N(I) GROUP, IF J IS NOT EQUAL TO N(I) SET J=J+1
AND REPEAT THIS STEP.

CI=BI*+J1
KI=DEC(CI,X)
NI=NO+KI
FI=BI/CI
NOI=NO+1
DO J=NOI,NI
II=NI-J
C=POCEF(FI,II,N)
DO I=1,N
Z(I,J)=C(N-I+1)
DOEND
DOEND
NO=NI

CONTINUE LOOPING BY SETTING I=I+1, IF I GREATER THAN K THEN END,
ELSE RETURN TO COMPUTE VECTOR C FROM THE BEGINNING OF OUTER DO LOOP.
PROCEDURE PSQFRE (BR, IK)

INPUT A BR PRIMITIVE UNIVARIATE POLYNOMIAL SUCH THAT
BR=B(1)*B(2)*...*B(K)**K
IK DEGREE OF BR
OUTPUT A Q LINEAR LIST OF THE SQUARE FREE FACTORS

LONG ALGEBRAIC (X) Q, D, B, I, E, F, B, BR
INTEGER I, IQ, IK
ARRAY (1 IK) Q
LONG ALGEBRAIC ALTRAN DIFF X
VALUE BR, IK

FIND THE GCD BETWEEN BR AND ITS DERIVATIVE WHICH IS
EQUAL TO B(2)*B(3)*...*B(K)**(K-1)
CONSTRUCT E1=BR/GCD WHICH IS EQUAL TO E1 WHERE
E1=B(1)+B(2)*...*B(K)
REPEAT THIS STEP FOR GCD OBTAINED BEFORE AND SET IT
EQUAL TO E2 WHERE
E2=B(2)+B(3)*...*B(K)
FROM WHICH B(1)=E1/E2.
CONTINUE THIS UNTIL ALL THE SQUARE FREE FACTORS ARE
COMPUTED.

L1^ IF (I=0) GO TO L3
L4^ IF (DEG(D, X) = DEG(F, X)) GO TO L3
L3^ IF (DEG(E, X) = 0) GO TO L8

END
PROCEDURE SOLEQ (A, B, N)

INPUT A MATRIX N+N

OUTPUT R THE UNKNOWN VECTOR SUCH THAT A*R=B

LONG ALGEBRAIC (X)R
LONG INTEGER C, A, B, BIG, F, D
INTEGER I, II, III, J1, J, K, JK, N
ARRAY (1:N) R
ARRAY (1:N, 1:N) A
ARRAY (1:N, 1:N) B
ARRAY (1:N, 1:N) F
ARRAY (1:N, 1:N) C
LONG INTEGER ALTRAN ABS
VALUE A, B, N
B=-B
F=A
C=0
D=1
DO II=1, N
  J1=1
  BIG=-1.00000
  DO I=1, N
    DO K=1, II
      IF (C(K), EQ, I) GO TO L1
      END DO
    IF (BIG, GE, ABS(A(I, I))) GO TO L1
    BIG=ABS(A(I, I))
    J1=I
    L1 CONTINUE
    DOEND
    C(II)=J1
    III=II+1
    DO J=1, N
      IF (II, EQ, N) GO TO L2
      DO JK=III, N
        IF (J, EQ, J1) GO TO L3
        F(J, JK)= (A(J1, II)+A(J, JK)-A(J1, JK))/D
        GO TO L4
        L3 CONTINUE
        L4 CONTINUE
        DOEND
    IF (J, EQ, J1) GO TO L5
    R(J)= (A(J1, II)+B(J)-B(J1)*A(J, II))/D
    GO TO L6
    L5 CONTINUE
    L6 CONTINUE
    DOEND
    D=A(J1, II)
    IF (D, EQ, 0) GO TO LF
    B=R
    A=F
    DOEND
    DO I=1, N
K = C(I)
R(I) = B(K)
DO END
R = R/D
RETURN(R)
END

PROCEDURE ABS (A2)

INPUT A2 INTEGER
OUTPUT A2 ABSOLUTE VALUE

LONG INTEGER A2
VALUE A2
IF (A2 LT 0) A2 = -A2
RETURN(A2)
END
APPENDIX  B

A Listing of the Program RINTGS

The following procedures are listed:

1. RINTGS
2. RINTG
3. MATX
PROCEDURE RINTGS(AB, R, S)

--- INPUT AB RATIONAL FUNCTION ---
--- OUTPUT R RATIONAL PART INTEGRAL ---
--- S TRANSCENDENTAL PART ---

LONG ALGEBRAIC(X) AB, A, B, R, S, F, Z, Y, G
LONG INTEGER C
INTEGER K, N
LONG ALGEBRAIC ALTRAN RINTG
VALUE AB
R = 0
S = 0
IF(AB == 0) GO TO LF
A = ANUM(AB)
B = ADEN(AB)
M = DEG(A, X)
N = DEG(B, X)
IF(N <= 0) GO TO L1
IF(N <= 0) GO TO L2
R = PINT(AB, X)
GO TO LF
L1
IF(N <= 0) GO TO L2
RINTG(AB, N, R, S)
RINTG(AB, N, R, S)
GO TO LF
G = COEFP0(EXPAND(B), X, N) ** (M-N+1)
A = C*A
F = C*B
Z = AQUO(A, B, X, Y)
Z = Z/C
G = PINT(Z, X)
Y = Y/F
N = DEG(ADEN(Y), X)
PROCEDURE RINTG(AB,N,R,S)

INPUT AB REGULAR RATIONAL FUNCTION
N DEGREE OF THE DENOMINATOR
OUTPUT R RATIONAL PART INTEGRAL
S TRANSCENDENTAL PART

INTEGER I,K,JK,N,IK
ARRAY(1AN)F
ARRAY(1AN)G
ARRAY(1AN,1AN)E
ARRAY(1AN)Z
LONG ALGEBRAIC ARRAY ALTRAN PSQFRE
LONG ALGEBRAIC ALTRAN CONT
LONG ALGEBRAIC ALTRAN SOLEQ
LONG INTEGER ARRAY ALTRAN POCEF
LONG ALGEBRAIC ARRAY ALTRAN MATX
VALUE AB,N
R=0
S=0
U=1
A=ANUM(AB)
BP=ADEN(AB)
BI=CONT(BP)

CALCULATE THE CONTENT OF AB AND CALL PSQFRE TO FIND
THE SQUARE FREE FACTORS OF THE PRIMITIVE PART OF
DENOMINATOR AB.

BP=BP/BI
Z=PSQFRE(BP,N)
DO IK=1,N
IF(Z(IK).EQ.0) GO TO L0
END

L0\^ 1 IK=IK-1

IF THE NUMBER OF THE SQUARE FREE FACTORS IS EQUAL TO ONE,
LET THE FUNCTION AB BECOME THE TRANSCENDENTAL PART
AND RETURN.

IF(IK<>1) GO TO L1
R=0
S=AB
RETURN (R,S)

CONSTRUCT •• •
U =B(1)*B(2)*........*B(K)
V =B(1)*B(2)*........*B(K)**(K-1)
E IS THE UNKNOWN N*N COEFFICIENT MATRIX FOUND BY
**CALLING PROCEDURE MATX.**

F CONSTANT VECTOR.

L1^ DO I=1, IK
U=U+Z(I)
DO END
V=BP/U
I=DEG(V,X)-1
K=DEG(U,X)-1
E=MATX(Z,U,V,N,IK)
F=POCEF(A,U,N)

**SOLVE THE SYSTEM OF LINEAR EQUATIONS TO FIND THE**
**COEFFICIENTS OF THE NUMERATOR OF BOTH RATIONAL AND**
**TRANSCENDENTAL PARTS.**
**DEMONINATOR OF RATIONAL PART IS EQUAL TO CONTENT OF**
**DENOMINATOR AB MULTIPLIED BY V.**
**DENOMINATOR OF THE TRANSCENDENTAL PART IS EQUAL TO**
**CONTENT OF THE DENOMINATOR AB MULTIPLIED BY U.**
**RETURN R, S.**

G=SOLVE(E,F,N)
W=BI
JK=1

L8^ IF (G(JK)>0) R=R+G(JK)**X**I
     I=I-1
     JK=JK+1
     IF (I,GE,0) GO TO L8

L10^ IF (G(JK)>0) S=S+G(JK)**X**K
     K=K-1
     JK=JK+1
     IF (K,GE,0) GO TO L10
     R=R/(W*V)
     S=S/(W*U)
     RETURN (R,S)
     END
PROCEDURE MAIX (F, U, V, N, K)

INPUT F LINEAR LIST OF THE SQUARE FREE FACTORS
   U POLYNOMIAL EQUAL TO B(1)*B(2)*...*B(K)
   V POLYNOMIAL EQUAL TO B(2)*(B(3)**2)*...*(B(K)**(K-1))
   N DEGREE OF DENOMINATOR AB.
   K NUMBER OF SQUARE FREE FACTORS.

OUTPUT M UNKNOWN COEFFICIENT MATRIX.

HEIGHT 20.45 BOX 1

HOW  F, U, V, N, K

W=0

DO J=1, K
   X1=U/F(J)
   JX=REP(F(J), V)
   W=W+JX*DIFF(F(J), X)*X1
END

CONSTRUCT MATRIX M SUCH THAT IF C IS THE NUMERATOR OF
   RATIONAL PART AND D IS THE NUMERATOR OF TRANSCENDENTAL
   PART, THEN
   C = C(M-1)*X**(M-1)+C(M-2)*X**(M-2)+...+C(0)
   AND
   D = D(N-M-1)*X**(N-M-1)+...+D(0).

WHERE DIFF(C, X) TAKES THE PARTIAL DERIVATIVE OF C WITH
   RESPECT TO X.
RETURN WITH THE MATRIX M.

DO J=1, N
   M(J+1, JK)=R(J+1)
END

IF(J, LT, (N-M1)) GO TO L5
R = \text{POCEF}(W, 0, N)
NJ = N - J
DO J1 = 1, N
M(J1, NJ) = R(J1)
END
J = 0
W = W + X
L7^a IF (J \geq M - 1) GO TO L9
X1 = W + (J + 1) + U
R = \text{POCEF}(X1, J, N)
NJ = NJ - 1
JK = J + 1
DO J1 = 1, N
M(J1, NJ) = R(J1)
END
J = J + 1
GO TO L7
L9^a RETURN (M)
END
PROCEDURE RINTGS(AB,NV,R,S)

INPUT A.AB RATIONAL FUNCTION
OUTPUT R RATIONAL PART INTEGRAL
S TRANSCENDENTAL PART

EXTERNAL INTEGER NX =NV
LONG ALGEBRAIC (X(NX)) AB,A,B,R,S,F,Z,Y,G,C
INTEGER M,N,NV
LONG ALGEBRAIC ALTRAN FINTG
VALUE AB
R=0
S=0
IF(AB==0) GO TO LF
A=ANUM(AB)
B=ADEN(AB)
N=DEG(A,X(1))
N=DEG(B,X(1))
IF(N>0) GO TO L1

IF THE DEGREE OF THE DENOMINATOR WITH RESPECT TO
X(1) EQUAL ZERO USE THE ALTRAN PROCEDURE PIINT.

R=PIINT(AB,X(1))
GO TO LF
L1 A IF(M>0) GO TO L2

IF THE DEGREE OF THE NUMERATOR (A) WITH RESPECT TO
X(1) IS LESS THAN THE DEGREE OF THE DENOMINATOR
(X) CALL THE RATIONAL INTEGRATION PROCEDURE.

RINTG(AB,NV,R,S)
GO TO LF

IF THE DEGREE OF THE NUMERATOR (A) WITH RESPECT TO
X(1) IS GREATER THAN THE DEGREE OF THE DENOMINATOR
(X) THEN DO DIVISION.
INTEGRATE THE POLYNOMIAL BY THE ALTRAN PROCEDURE
PIINT AND THE REGULAR RATIONAL PART BY THE RATIONAL
INTEGRATION PROCEDURE.

L2 A C=COEFPO(EXPAND(B),X(1),N)**(M-N+1)
A=C*A
F=C*B
Z=AQUO(A,B,X(1),Y)
Z=Z/C
G=PIINT(Z,X(1))
Y=Y/F
N=DEG(ADEN(Y),X(1))
RINTG(Y,N,R,S)
RETURN (R,S)

PROCEDURE CONT (BB)

INPUT BB MULTIPLIATE POLYNOMIAL OVER INTEGER

OUTPUT Z CONTENT OF BB

EXTERNAL INTEGER NX
LONG ALGEBRAIC (X(NX)). B, BB, Z, A, C, D
INTEGER I, N
VALUE BB
B=ANUM (BB)
D=ADEN (BB)
NEDEG (B, X(1))
IF (M<>0) GO TO L2
Z=1
Z=Z/D
RETURN (Z)

CONSTRUCT VECTOR Z SUCH THAT POLYNOMIAL B WILL BE
EQUAL TO W
B =Z(M) *X(1) ** (M-1) + Z(M-1) *X(1) ** (M-2) + ... + Z(0)
GCD (Z(M), Z(M-1),..., Z(1))

L2^ Z=COEFP0 (EXPAND (B), X(1), M)
M=M-1
DO I=N, 0, -1
A=COEFP0 (EXPAND (B), X(1), I)
Z=AGCD (Z, A)
C=Z
IF (D,LG (C, X(1)) .EQ. 0) GO TO LF
END
Z=Z/D
RETURN (Z)
END
PROCEDURE RINTG(AB,N,R,S)

INPUT AB, N  DEGREE OF THE DENOMINATOR WITH RESPECT TO X(1)
OUTPUT R, S  RATIONAL PART INTEGRAL

EXTERNAL INTEGER NX
LONG ALGEBRAIC(X(NX)) AB, R, S, A, BP, Z, U, V, E, F, G, N, BI
INTEGER I, J, JK, N, IK
ARRAY(IAN) F
ARRAY(IAN) G
ARRAY(IAN) E
ARRAY(IAN) Z
LONG ALGEBRAIC ARRAY ALTRAN PSQFRE
LONG ALGEBRAIC ARRAY ALTRAN CONT
LONG ALGEBRAIC ARRAY ALTRAN ASOLVE
LONG ALGEBRAIC ARRAY ALTRAN POCEF
LONG ALGEBRAIC ARRAY ALTRAN MATX
VALUE AB, N
E=0
S=0
U=1
A=ANUM(AB)
BP=ADEN(AB)
BI=CONT(BP)

CALCULATE THE CONTENT OF AB AND CALL PSQFRE TO FIND
THE SQUARE FREE FACTORS OF THE PRIMITIVE PART OF
DENOMINATOR AB.

BP=3P/BP
Z=PSQFRE(3P,N)
DO IK=1,N
IF(Z(IK).LE.Q,0) GO TO L0
DO END
IK=IK-1

IF THE NUMBER OF THE SQUARE FREE FACTORS IS EQUAL TO ONE,
LET THE FUNCTION AB BECOME THE TRANSCENDENTAL PART
AND RETURN.

IF(IK<->1) GO TO L1
E=U
S=AB
RETURN (R, S)

CONSTRUCT
U =B(1)*B(2)*...*B(K)
V =B(2)*B(3)*...*B(K)*B(K-1)
E is the unknown \( N \times N \) coefficient matrix found by calling procedure MATX.

\[
\begin{align*}
&L1^A \quad \text{DO I=1,IK} \nonumber \\
&\quad \text{U=U*Z(1)} \nonumber \\
&\quad \text{DOEND} \nonumber \\
&\quad \text{V=BP/U} \nonumber \\
&\quad \text{I=DEG(V,X(1))-1} \nonumber \\
&\quad \text{K=DEG(U,X(1))-1} \nonumber \\
&\quad \text{E=MATX(Z,U,V,N,IK)} \nonumber \\
&\quad \text{F=PODIF(A,J,N)} \nonumber
\end{align*}
\]

Solve the system of linear equations to find the coefficients of the numerator of both rational and transcendental parts.

\[
\begin{align*}
&\text{DENOMINATOR OF RATIONAL PART IS EQUAL TO CONTENT OF} \nonumber \\
&\text{DENOMINATOR AS MULTIPLIED BY V.} \nonumber \\
&\text{DENOMINATOR OF THE TRANSCENDENTAL PART IS EQUAL TO} \nonumber \\
&\text{CONTENT OF THE DENOMINATOR AS MULTIPLIED BY U.} \nonumber \\
&\text{RETURN R,S,} \nonumber
\end{align*}
\]

\[
\begin{align*}
&G=\text{ASOLVE(E,F)} \nonumber \\
&\text{W=DI} \nonumber \\
&JK=1 \nonumber \\
&L6^A \quad \text{IF(G(JK),NE,U)R=R+G(JK)*X(1)**1} \nonumber \\
&\quad \text{L10} \nonumber \\
&\quad \text{IF(K,NE,U)GO TO L10} \nonumber \\
&\quad \text{R=R/(H+V)} \nonumber \\
&\quad \text{S=S/(H+U)} \nonumber \\
&\quad \text{RETURN (R,S)} \nonumber \\
&\text{END} \nonumber
\end{align*}
\]
PROCEDURE MATX (F,U,V,N,K)

INPUT F LINEAR LIST OF THE SQUARE FREE FACTORS
U POLYNOMIAL EQUAL TO B(1)*B(2)*...*B(K)
V POLYNOMIAL EQUAL TO B(2)*(B(3)+...+B(K)+...+B(K)+...+B(K))
N DEGREE OF THE DENOMINATOR AS WITH RESPECT TO X(1)
K NUMBER OF SQUARE FREE FACTORS
OUTPUT F UNKNOWN COEFFICIENT MATRIX.

EXTERNAL INTEGER NX
LONG ALGEBRAIC (X(NX))F,U,V,X1,W,R,M
INTEGER K,J,J1,JK,J1,N,J,K,JX
ARRAY(IN1N)
ARRAY(IN1N)M
ARRAY(IN1N)F
LONG ALGEBRAIC ALTRAN DIFFX
LONG ALGEBRAIC ARRAY ALTRAN POCEF
INTEGER ALTRAN REP
VALUE F,U,V,N,K
W=0

CALCULATE W=W(2)+W(3)+...+W(K)
WHERE W(I)=X(1)*U/I+X(1)*DIFFX(B(I)*X(1))

DO J=1,K
X1=U/F(J)
JX=REP(F(J),V)
N=W+JX*DIFFX(F(J),X(1))*X1
END
W=3
J=0
M=DEG(V,X(1))
L=1
REPPOCEF(V,J,M)
JK=N-J

CONSTRUCT MATRIX M SUCH THAT IF C IS THE NUMERATOR OF
RATIONAL PART AND D IS THE NUMERATOR OF TRANSCENDENTAL
PART THEN:
C =C(M-1)*X(1)**(M-1)+C(M-2)*X(1)**(M-2)+...+C(0)
AND
D =D(N-M-1)*X(1)**(N-M-1)+...+D(0).
FROM THE UNKNOWN COEFFICIENT MATRIX CONSTRUCT THE
EQUATION U*V+C*M+U*DIFFC*X(1)
WHERE DIFFC*X(1) TAKES THE PARTIAL DERIVATIVE OF C
WITH RESPECT TO X(1).
RETURN WITH THE MATRIX M.

DO J1=1,N
M(J1,JK)=R(J1)
END
J=J+1
58 IF(J,LT,(N-M1))GO TO L8
59 R=POCEP(W,J,N)
60 NJ=N-J
61 DO J1=1,N
62 M(J1,NJ)=R(J1)
63 END DO
64 J=0
65 L7: W=W+X(1)
66 IF(J4G.E.(M1-1))GO TO L9
67 X1=W+(J+1)*U
68 R=POCEP(X1,J,N)
69 NJ=N-J-1
70 JK=J+1
71 DO J1=1,N
72 M(J1,NJ)=R(J1)
73 END DO
74 J=J+1
75 L9: GO TO L7
76 RETURN (M)
77 END
APPENDIX C

A Listing of the program MVFOI
PROCEDURE MVFOI(N,M,U)

INPUT N NUMBER OF VARIABLES
INPUT M DEGREE OF U WITH RESPECT TO X(1)
INPUT U MULTIVARIATE POLYNOMIAL
OUTPUT F ARRAY CONTAINING THE FACTORS OF POLYNOMIAL U

INTEGER N,M,A,Q,P,U,PQ,IR,J,IT
ALGEBRAIC(X(N))U,UI,UD,Z,H,F,Y
ARRAY(1:IM)Y
ARRAY(1:IM)F
ARRAY(1:IM)A
ARRAY(1:IM)H
ARRAY(1:IM)Z
ALGEBRAIC ALTRAN VSUBT
ALGEBRAIC ALTRAN DIFFX
ALGEBRAIC ALTRAN MRDPO
INTEGER ALTRAN HPRIME
INTEGER ALTRAN GBOUND
ALGEBRAIC ARRAY ALTRAN PFOINT
ALGEBRAIC ARRAY ALTRAN PFCI
ALGEBRAIC ARRAY ALTRAN EXZH
ALGEBRAIC ARRAY ALTRAN TRUPAC

CALL PROCEDURE VSUBT TO PERFORM VARIABLE SUBSTITUTION AND
TO OBTAIN THE INTEGER ARRAY A SUCH THAT
UI=U(X(1),A(2),......,A(N))
AND UI IS SQUARE FREE POLYNOMIAL WITH DEGREE EQUAL TO THE DEGREE
OF U WITH RESPECT TO X(1).

VSUBT(U,N,A,UI)

CALL PROCEDURE PFOINT TO PERFORM VARIABLE SUBSTITUTION AND
TO OBTAIN THE INTEGER ARRAY A SUCH THAT
UI=U(X(1),A(2),......,A(N))
AND UI IS SQUARE FREE POLYNOMIAL WITH DEGREE EQUAL TO THE DEGREE
OF U WITH RESPECT TO X(1).

PFOINT(U,N,IR)

CALL PROCEDURE ALTRAN VSUBT TO PERFORM VARIABLE SUBSTITUTION AND
TO OBTAIN THE INTEGER ARRAY A SUCH THAT
UI=U(X(1),A(2),......,A(N))
AND UI IS SQUARE FREE POLYNOMIAL WITH DEGREE EQUAL TO THE DEGREE
OF U WITH RESPECT TO X(1).

SUBT(U,N,A,UI)

CALL PROCEDURE ALTRAN PFOINT TO PERFORM VARIABLE SUBSTITUTION AND
TO OBTAIN THE INTEGER ARRAY A SUCH THAT
UI=U(X(1),A(2),......,A(N))
AND UI IS SQUARE FREE POLYNOMIAL WITH DEGREE EQUAL TO THE DEGREE
OF U WITH RESPECT TO X(1).

PFOINT(U,N,IR)
IF(Z, EQ, 0) GO TO L1

COMPUTE THE COEFFICIENT BOUND USING PROCEDURE CBOUND FROM WHICH WE OBTAIN PQ SUCH THAT PQ=P***(2**J)

D=CBOUND(UI, M, N, P)
PQ=P***(2**D)

PERFORM ZASSENHAUS ALGORITHM TO OBTAIN THE VECTOR F SUCH THAT

UI=F(1)*F(2)*...*F(IR) (MOD PQ)

F=PFCI(P, PQ, UI, Z, IR, M, N)

OBTAIN THE UNIVARIATE FACTORS BY CALLING PROCEDURE TRUFAC SUCH THAT

UI=H(1)*H(2)*...*H(L)

H=TRUFAC(UI, 1, F, IR, N, PQ, M)
F=0
DO IT=1, IR
IF(Z(IT), EQ, 0) GO TO L4
DO END

L4^ IT=IT-1

APPLY THE EXTENDED ZASSENHAUS ALGORITHM TO OBTAIN THE MULTIVARIATE FACTORS BY USING PROCEDURE EXZH SUCH THAT

U=Y(1)*Y(2)*...*Y(R) (MOD (PQ, S**J))

EXZH(PQ, U, Z, A, IT, N, M, Y, J)

APPLY THE PROCEDURE TRUFAC TO OBTAIN THE ACTUAL FACTORS SUCH THAT

U=F(1)*F(2)*...*F(K)

F=TRUFAC(U, J, Y, IT, N, PQ, M)
RETURN(F)

END
PROCEDURE VSUBT(U,N,A,UX)

INPUT  U  MULTIVARIATE POLYNOMIALS
      N   NUMBER OF VARIABLES
OUTPUT A  ARRAY CONTAINING INTEGERS USED FOR SUBSTITUTION
      UX  UNIVARIATE POLYNOMIAL SUCH THAT

UX=U(X(1),A(2),...........,A(N))

INTEGER N,M,A,L,I,J,K1,C,Z
ALGEBRAIC(X(N))U,UX,UX1,UX2,LDC,TRC
ALGEBRAIC ALTRAN DDIFFX
ARRAY(1AN)A
ARRAY(1AN)C
VALUE U,N

SET N=K1=J=1,C=A=0, LDC EQUAL TO THE LEADING COEFFICIENT TERM AND TRC TO THE TRAILING COEFFICIENT TERM.

M=1
C=0
A=U
K1=1
J=1
L=0EG(U,X(1))
LDC=COEFFPO(EXPAND(U),X(1),L)
TRC=COEFFPO(EXPAND(U),X(1),L)

IF A VARIABLE OR VARIABLES OF THE SET (X(2),...,X(N)) CAN BE FACTORED FROM THE LEADING COEFFICIENT TERM OF U, ASSIGN A VALUE OF K1 TO THE VARIABLE OR VARIABLES, DO THE SAME FOR THE TRAILING COEFFICIENT TERM EXCEPT THAT THE ASSIGNED VALUE WILL BE (K1+1)*J MOD 5 INSTEAD OF K1 SET THE REMAINDER OF THE VARIABLES EQUAL ZERO, PLACE THESE VALUES INTO VECTOR A.

DO I=2,N
   IF((DEG(LDC,X(I)),NE.0).OR.(DEG(TRC,X(I)),NE.0))GO TO L1
   UEND
   GO TO L3
L1
   DO I=2,N
   IF(AGCD(LDC,X(I)),NE.X(I))GO TO LY
   A(I)=K1
   C(I)=1
   GO TO L2
LY
   IF(AGCD(TRC,X(I)),NE.X(I))GO TO L2
   IF(C(I),EQ,1)GO TO L2
   A(I)=MOD(((K1+1)*J,5)
   J=J+1
   C(I)=1
   CONTINUE
L2
   CONTINUE
   DO=ND


SUBSTITUTE (A(2), ..., A(N)) FOR (X(2), ..., X(N)) IN U
AND LET THE NEW POLYNOMIAL EQUAL TO UX.
IF DEGREE UX IS EQUAL TO THE DEGREE OF U WITH RESPECT TO X(1) AND
THE GREATEST COMMON DIVISOR OF UX, DIFF(UX, X(1)) EQUAL TO ONE
THEN END.
REINITIALIZE THE SET (X(2), ..., X(N)) IF X(I) IS NOT ONE
OF THE LEADING OR TRAILING COEFFICIENT TERM OF U SET A(I) = 0
SET J = I + 1

I = 1
UX = U
DO Z = 2, N
UX = UX(X(Z) = A(Z))
DO = END
DOUX = DIFFX(UX, X(1))
IF(DEG(UX, X(1)), EQ, L) GO TO L6
IF(C(I), NE, 1) A(I) = 0
J = I + 1

FROM I = J TO N DO
IF(A(I), NE, 0) GO TO L5
A(I) = K1
GO TO L3
J = I + 1
CONTINUE
DO = END
DO = END
DOUX = DIFFX(UX, X(1))
IF(DEG(UX, X(1)), EQ, L) GO TO L6
IF(C(I), NE, 1) A(I) = 0
J = I + 1

DO I = J, N
IF(A(I), NE, 0) GO TO L5
A(I) = K1
GO TO L3
CONTINUE
DO = END
DOUX = DIFFX(UX, X(1))
IF(DEG(UX, X(1)), EQ, L) GO TO L6
IF(C(I), NE, 1) A(I) = 0
J = I + 1

DO Z = 2, N
IF(A(Z), EQ, 0) GO TO L51
DO = END
GO TO L52
IF(K1 GT 0) GO TO LX
K1 = -K1 + 1
IF(K1 LT S) GO TO LX1
M = M + 1
K1 = M
GO TO L0
LX
K1 = -K1
LX1
I = 1
IF(C(N), NE, 1) A(N) = 0
GO TO L34
L6
IF(DEG(A, UX) X(1)), EQ, 0) RETURN (A, UX)
IF(I, GE, N) GO TO L51
GO TO L34
PROCEDURE CBOUND(U,M,N,Q)

INPUT: U UNIVARIATE POLYNOMIAL
M DEGREE OF U
N NUMBER OF VARIABLES
Q PRIME NUMBER
OUTPUT: J INTEGER FROM WHICH THE MODULUS CAN BE COMPUTED SUCH
THAT B=Q**(2**J)

INTEGER M, Q, C, MAXC, LDC, B, J, N, M1
ALGEBRAIC(X(N))U
ARRAY(1^(M+1)) C
INTEGER ALTRAN MAX
INTEGER ARRAY ALTRAN POCEF
VALUE U, M, N, Q
M1=M+1
C=POCEF( EXPAND(U), 0, M+1, N)
DO J=1, M1
IF(C(J) LT U) C(J) = -C(J)
DOEND

CALL PROCEDURE MAX TO SEARCH FOR THE MAXIMUM COEFFICIENT
(MAXC) OF THE POLYNOMIAL U, SET J=1

MAXC=MAX(C, M+1)
LDC=COEFPO(EXPAND(U), X(1), M)
IF(LDC, LT, U) LDC=-LDC

IF 3*ABS(LDC(U))*MAXC IS LESS THAN Q**(2**J) THEN END
ELSE SET J=J+1 AND RETURN FOR TEST AGAIN.

B=3*LDC*MAXC
DO J=1, 20
IF(B LT Q**(2**J)) RETURN(J)
DOEND
END
PROCEDURE PFOINT(AX, NI, N, P, Z, M)

INPUT  AX  UNIVARIATE POLYNOMIAL
       NI  DEGREE OF AX WITH RESPECT TO X(1)
       N  NUMBER OF VARIABLES
       P  PRIME NUMBER
OUTPUT Z  ARRAY CONTAINING THE IRREDUCIBLE POLYNOMIALS
M  NUMBER OF IRREDUCIBLE POLYNOMIALS OVER GF(P) SUCH THAT
AX=Z(1)*Z(2)*...*Z(M) (MOD P)

INTEGER N, M, NI, ZZ, P
ALGEBRAIC (X(N)) AX, Z, W
ARRAY(1*NI) Z
ARRAY(1*NI) 1
ARRAY(1*NI) W
INTEGER ARRAY ALTRAN CPTOM
ALGEBRAIC ARRAY ALTRAN CPBQ
ALGEBRAIC ARRAY ALTRAN NULLSP
ALGEBRAIC ARRAY ALTRAN BRLKPF

TO COMPUTE X**((P*I)) MODULO AX CALL PROCEDURE CPBQ WHICH COMPUTES
VECTOR Z AS OUTPUT SUCH THAT
Z(I)=X**((P*I)) (MOD AX), WHERE I=0, ..., NI-1
Z=CPBQ(AX, NI, P, N)

CONSTRUCT THE ZZ MATRIX BY PLACING THE COEFFICIENT OF
POLYNOMIAL Z(I) IN THE I TH ROW OF THE MATRIX ZZ FOR
I=0, ..., NI-1; CALLING PROCEDURE CPTOM TO PERFORM THIS FUNCTION.
ZZ=CPTOM(Z, NI, N)

TO COMPUTE THE INDEPENDENT VECTORS CALL PROCEDURE NULLSP WHERE
THE CORRESPONDING FACTORS W ARE COMPUTED,
NULLSP(ZZ, NI, P, M, W, N)

CALL PROCEDURE BRLKPF TO OBTAIN THE IRREDUCIBLE POLYNOMIALS Z OVER
GF(P) SUCH THAT
AX=Z(1)*Z(2)*...*Z(M) (MOD P)
Z=BRLKPF(P, AX, M, W, NI, N)
RETURN(Z, M)
END
PROCEDURE CPBQ (A, J, P, N)

INPUT A UNIVARIATE POLYNOMIAL
J DEGREE OF A
P PRIME NUMBER
N THE NUMBER OF VARIABLES

OUTPUT Q ARRAY OF POLYNOMIALS SUCH THAT
Q(I) = X(1)^I (MOD A)

ALGEBRAIC (X(N)) Q, B, C, A, D
INTEGER P, L, K, J, I, NI
ALGEBRAIC ALTRAN MNULPO
ALGEBRAIC ALTRAN MREDPO
ARRAY(I, A) Q
VALUE A, J, P

SET K = LOG(P), L = 2**K, M = P - L AND B = X
WHERE K IS THE GREATER INTEGER LESS THAN OR EQUAL TO LOG(P), WHERE THE BASE OF LOG FUNCTION IS TWO.

L1
L = L + L
IF (L > P) GO TO L1
L = L/2
B = X(1)
M = P - L
L = IQUO(L, 2)

SET B EQUAL TO THE REMAINDER OF B^M/Q (MOD P)
IF M IS LESS THAN L GO TO STEP 3, ELSE SET M = M - L AND B EQUAL TO THE REMAINDER X*B/A (MOD A)

L2
C = MNULPO(B, B, P)
AQUO(C, A, X(1), B)
NI = DEG(B, X(1))
C = MREDPO(EXPAND(B), P)
IF (M, LT, L) GO TO L3
M = K - L
B = C + X(1)
AQUO(B, A, X(1), C)
NI = DEG(C, X(1))
B = MREDPO(EXPAND(C), P)

SET L = L/2, IF L IS NOT EQUAL TO ZERO GO TO L2, ELSE
SET C = 1, Q(1) = 1 AND FOR I = 2, ..., J DO,
SET C EQUAL TO THE REMAINDER OF B*C/A (MOD P)
SET Q(I) = C AND CONTINUE LOOPING.

L3
L = IQUO(L, 2)
IF (L, NE, L) GO TO L2
D=3
C=1
Q(I)=C
NI=J-1
DO I=2, J
B=MULPO(EXPAND(D),EXPAND(C),P)
B=REDPO(EXPAND(B),P)
A=QUO(B,A,X(I),C)
NI=DEGIC,X(I)
Q(I)=REDPO(EXPAND(C),P)
C=Q(I)
END
RETURN (Q)
END
PROCEDURE CPTOM(Q,L,N)

INPUT  Q ARRAY CONTAINING UNIVARIATE POLYNOMIALS

L ARRAY SIZE

N NUMBER OF VARIABLES

OUTPUT QQ MATRIX CONTAINING THE COEFFICIENTS OF THE POLYNOMIALS

IN ARRAY Q.

ALGEBRAIC(X(N))Q

INTEGER Q1,QQ,L,N,I,J

ALGEBRAIC ARRAY ALTRAN POCEF

ARRAY(1^L)Q

ARRAY (1^L) Q1

ARRAY (1^L,1^L) QQ

VALUE Q,N

---

FOR J=1,L DO

SET VECTOR Q1 EQUAL TO THE COEFFICIENT OF POLYNOMIAL Q(I),

THEN PLACE VECTOR Q1 IN J TH ARRAY OF MATRIX QQ.

---

DO J=1,L

Q1=POCEF(Q(J),L,N)

DO I=1,L

QQ(J,I)=Q1(I)

DOEND

RETURN(QQ)

END
PROCEDURE NULLSP(A, M, P, R, V, N)

INPUT A MATRIX CONTAINING THE COEFFICIENTS OF THE EQUATION X**P MODULO U(X).
M DEGREE OF U(X).
OUTPUT V ARRAY CONTAINING THE INDEPENDENT VECTORS.
R NUMBER OF INDEPENDENT VECTORS.

ALGEBRAIC (X(N)) V
ARRAY(1^M, 1^M) A1
ARRAY(1^M) C
ARRAY(1^M) V
ARRAY(1^M, 1^M) A
ARRAY(1^M, 1^M) VR
INTEGER ALTRAN IRECS
VALUE A, N, P

SET VECTOR C=-1 FOR I=1, M
SET A(I, I)=A(I, I)-1

DO I =1, M
DO J =1, M
IF(I = EQ. J) A(I, J) = A(I, J) -1
END
END
A1=A
V=0
R=0
C=-1

SCAN THE ROW K OF MATRIX Q FOR DEPENDENCE. IF THERE IS SOME J IN
THE RANGE BETWEEN 0 AND M SUCH THAT Q(K, J) IS NOT EQUAL TO ZERO
AND C(J) IS LESS THAN ZERO, THEN MULTIPLY THE J TH COLUMN BY
-1/Q(K, J).
ADD Q(K, J) TIMES THE J TH COLUMN TO THE I TH COLUMN FOR ALL
IF K IS GREATER THAN M GO TO NEXT STEP, ELSE
REPEAT THE SCANNING PROCESS.

DO K =1, M
I=K
DO S =1, M
IF ((A(I, S), NE. 0), AND, (C(S), LT. 0)) GO TO L51
GO TO L52
L51
DO L =1, M
IF (S, EQ. C(L)) GO TO L52
END
GO TO L5
L52 CONTINUE
END
COMPUTE THE INDEPENDENT VECTORS AND THE CORRESPONDING POLYNOMIALS SUCH THAT $R = F; + 1 \cdot$ FOR $J = 1, \ldots, M$ 

$\text{IF } J = K \text{ OR } VR(J) = 1 \text{ OR } CC(S) = J \text{ OR } CC(S) = K \text{ OR } S$ 

$\text{IF } J \text{ IS GREATER THAN } K \text{ THEN } VR(J) = 0$, ELSE $R = VR(J)$ 

$\text{IF } K \text{ IS GREATER THAN } M \text{ THEN } $ 

$\text{REPEAT THE SCANNING PROCESS FOR } K = 2, \ldots, M$ 

$R = K - 1 \cdot$ DO $J = 1$ 

$\text{IF } C(J) = K \text{ GO TO } L_3$ 

$00 \ L = 1, M \text{ NOT EQUAL TO } J , \text{ SET } C(J) = K , K = K + 1$ 

$\text{IF } C(J) = J \text{ GO TO } L_2$ 

DO $J = 1, M$ 

$VR(J) = 1 \cdot$ GO TO $L_7$ 

$\text{IF } VR(J) = 0$ 

$\text{END DO END}$ 

$A = A_1$ 

$\text{DO }$ $J = M - 1$ 

$\text{DO } I = 1, R$ 

$V(L) = V(L) + VR(I) A(I, L) + A(L, J) + A(I, J)$ 

$\text{END DO}$ 

$\text{END DO END}$ 

$\text{CONTINUE DO END}$ 

$\text{IF } K \text{ IS EQUAL TO } 0 \text{ OR } VR(K) = 1 \text{ OR } VR(K) = 2 \text{ OR } VR(K) = 3 \text{ OR } VR(K) = 4 \text{ ELSE IF ALL THE ABOVE}$ 

$\text{END DO END}$
PROCEDURE BRLKPF (P,A,H,R,L,N)

--- INPUT ---

P THE PRIME NUMBER
A UNIVARIATE POLYNOMIAL
H ARRAY CONTAINING POLYNOMIALS COMPUTED FROM PROCEDURE
R NUMBER OF FACTORS IN VECTOR H.
L ARRAY SIZE EQUAL TO THE DEGREE OF A
N NUMBER OF VARIABLES

--- OUTPUT ---

T VECTOR CONTAINING ALL THE IRREDUCIBLE POLYNOMIAL
T(1),T(2),...,T(R) SUCH THAT
A=T(1)+T(2)+...+T(R) (MODULO P).

INTEGER \( k,P,P1,i,K,R,J,N,L \)
ALGEBRAIC \( x \) \( A,H,S,T,B1,G \)
ARRAY \( \{H\} \)
ARRAY \( \{S\} \)
ARRAY \( \{T\} \)
ALGEBRAIC ALTRAN HREDPO
ALGEBRAIC ALTRAN CPGCD1
ALGEBRAIC ARRAY ALTRAN ORDPOL
VALUE \( P,A,H,R,N \)

--- SET ---

SET \( S=0 \), \( T=0 \), \( M=0 \), \( S(1)=A \)

I=I+1 AND EMPLOY ANOTHER FACTOR \( H(I) \), IF \( H(I) \) HAS A VALUE
EQUAL TO ZERO THEN END, ELSE SET \( T=0 \), \( K=1 \).

S=0
T=0
M=0
P1=P-1
I=1
S(1)=A
I=I+1
IF \( (H(I)) \) .EQ. 0 RETURN \( T \)
T=0
K=1
BI=H(I)

EMPLY ANOTHER POLYNOMIAL \( S(K) \), ASSIGN TO \( G \), \( \text{GCD}(H(I)-J,S(K)) \) MOD \( P \),
IF DEGREE OF \( G \) IS NOT EQUAL TO ZERO OR IF IT IS EQUAL TO
THE DEGREE OF \( S(K) \) THEN SET \( G \) IN VECTOR \( T \) USING
PROCEDURE ORDPOL.
SET \( M=M+1 \), \( S(K)=\text{REMS}(K),G \), IF \( S(K)=0 \) GO TO LX, ELSE IF \( M=R \)
GO TO LX, ELSE CONTINUE LOOPING.

L3A
C=S(K)
S(K)=0
K=K+1
DO \( J=0 \), \( P1,1 \)
IF \( \text{DEG}(C),(X(1)) \) .EQ. 0 GO TO L6
G=CPGCD1 \( P,(B1-J),C,N \)
G = MRREDPO(EXPAND(G), P)
IF (DEG(G, X(1)) .EQ. 0) GO TO LF
IF (DEG(G, X(1)) .EQ. DEG(C, X(1))) GO TO L6
T = ORDPOL(G, T, L, P, N)
M = M + 1
ARE(I(C, G, X(1), C)
C = MRREDPO(EXPAND(C), P)
IF (C .EQ. C) GO TO LX
IF (M .EQ. R) GO TO L7
CONTINUE
DO END

\begin{verbatim}
\textbf{INSERT S(K) INTO VECTOR T,}
\textbf{IF S(J) IS NOT EQUAL TO ZERO RETURN TO EMPLOY ANOTHER POLYNOMIAL}
\textbf{FROM VECTOR S, ELSE SET S=0 AND RETURN TO EMPLOY ANOTHER FACTOR}
\end{verbatim}

\begin{verbatim}
L6A T = ORDPOL(C, T, L, P, N)
LXA DO J = 1, R
   IF (S(J) .NE. L) GO TO L4
   DO END
   S = 0
   GO TO L3
\end{verbatim}

\begin{verbatim}
\textbf{INSERT S(K) INTO VECTOR T, THEN INSERT ALL OTHER NONZERO POLYNOMIALS}
\textbf{OF VECTOR S INTO VECTOR T, THEN END.}
\end{verbatim}

\begin{verbatim}
L7A T = ORDPOL(C, T, L, P, N)
   DO I = 1, R
      IF (S(I) .NE. U) ORDPOL(S(I), T, L, P, N)
   DO END
   RETURN(T)
END
\end{verbatim}
PROCEDURE JRODPOL(A,B,L,P,N)

INPUT  A  UNIVARIATE POLYNOMIAL
       B  ARRAY CONTAINING POLYNOMIALS
       L  SIZE OF ARRAY B

OUTPUT  B  ARRAY B AFTER INSERTING POLYNOMIAL A INTO LOCATION I.
         I BEFORE INSERTION, WE SHIFT DOWNWARD BY ONE LOCATION THE ELEMENTS
         OF ARRAY B STARTING FROM LOCATION I. AFTER INSERTION THE
         RESULTS OF ELEMENTS OF B ARE SUCH THAT THE DEGREE (B(I-1)) IS
         LESS THAN THE DEGREE OF(A) WHICH IS LESS THAN THE DEGREE (B(I+1))

INTEGER I,M,N,J,P,L
ALGEBRAIC (X(N))A,B,F
ARRAY(1^L)B
VALUE A,B,N

SET M=DEGREE OF A, J=1.
IF B IS NULL ARRAY, THEN INSERT A INTO B(1) AND RETURN.
IF M IS GREATER THAN THE DEGREE OF B(J) SET J=J+1
AND REPEAT THIS TEST, ELSE SHIFT DOWNWARD BY ONE PLACE ALL THE
ELEMENTS OF ARRAY B STARTING AT THE I TH POSITION, AND INSERT
A IN LOCATION I.

L1^ M=DEG(A,X(1))
   J=1
L2^ IF(B(J),NE.,0)GO TO L3
   B(J)=A
   RETURN(B)
L3^ IF(M,LT,DEG(B(J),X(1)))GO TO L4
   J=J+1
   GO TO L2
L4^ F=B(J)
   B(J)=A
   IF(F,NE.,0)GO TO L5
   A=F
   J=J+1
   GO TO L4
L5^ RETURN(B)
END
PROCEDURE CHONIC(P,A,N)

INPUT P PRIME NUMBER
A UNIVARIATE POLYNOMIAL
N NUMBER OF VARIABLES

OUTPUT A UNIVARIATE POLYNOMIAL AFTER PUTTING A IN THE MONIC FORM.

INTEGER P,L,K,N,M,J
ALGEBRAIC(X(N))A,B
ALGEBRAIC ALTRAN XREDPO
ALGEBRAIC ALTRAN IRECS

VALUE A,P

SET J EQUAL TO THE LEADING COEFFICIENT OF A, THEN K IS THE RECIPROCAL OF J MODULO P.
SET A EQUAL TO K*A MODULO P.

M=DEG(A)*X(1))
J=COEFPO(EXPAND(A),X(1),M)
K=IRECS(J,P)
B=K*A
A=XREDPO(EXPAND(B),P)
RETURN (A)
END

PROCEDURE MAX(C,L)

INPUT C ARRAY CONTAINING INTEGERS
L ARRAY SIZE

OUTPUT MAXI MAXIMUM INTEGER IN THE ARRAY C

INTEGER L,C,MAXI,I
ARRAY(L)C
VALUE C,L
MAXI=C(1)
DO I=2,L
IF(MAXI.LT.C(I))MAXI=C(I)
END
RETURN (MAXI)
END
SUBROUTINE CPGCD1(P,A,B,N)

INPUT P PRIME NUMBER
    A UNIVARIATE POLYNOMIAL
    B UNIVARIATE POLYNOMIAL
    N NUMBER OF VARIABLES

OUTPUT C MONIC GREATEST COMMON DIVISOR OF A AND B OVER GF(P)

INTEGER P,R1,N
ALGEBRAIC (X(N))A,B,C,R
ALGEBRAIC ALTRAN MRDEPO
ALGEBRAIC ALTRAN Mнич
INTEGER ALTRAN IRECS
VALUE A,B,P

SET R EQUAL TO THE REMAINDER OF A/B (MOD P)
SET R1 TO THE RECIPROCAL OF THE DENOMINATOR OF POLYNOMIAL
R AND C=R*K/R1
LET A=B,B=C
IF B NOT EQUAL ZERO RETURN TO STEP L2, ELSE SET C TO THE
MONIC OF POLYNOMIAL A THEN END.

L2^ AQUO(A,B,X(1),R)
R1=IRECS(AVEN(R),P)
R=ANUM(R)+R1
C=MRDEPO(EXPAND(R),P)
A=B
B=C
IF(B,NE,0) GO TO L2
C=CHONIC(P,A,N)
RETURN (C)
END
PROCEDURE PFCI(P, M, C, G, T, NI, N)

INPUT P

M THE MODULUS NUMBER WHICH IS EQUAL TO P**(2**J)

C UNIVARIATE POLYNOMIAL

G ARRAY OF POLYNOMIALS OVER GF(P) SUCH THAT

\( C = G(1) \times G(2) \times \cdots \times G(T) \pmod{P} \)

T NUMBER OF IRREDUCIBLE POLYNOMIALS

NI ARRAY SIZE

N NUMBER OF VARIABLES

OUTPUT F ARRAY CONTAINS POLYNOMIALS \( F(1), \ldots, F(T) \) SUCH THAT

\( U = F(1) \times F(2) \times \cdots \times F(T) \pmod{M} \)


ALGEBRAIC \( X(NI) \), C, GP, CP, AP, BP, G, SP, TP, A, B, F

ARRAY(1\&NI) C

ARRAY(1\&NI) GP

ALGEBRAIC ALTRAN MREDPO

ALGEBRAIC ALTRAN PFH1

ALGEBRAIC ALTRAN PEGCDX

INTEGER ALTRAN IRECS

VALUE P, M, C, G, T

SET CP = C \pmod{P}, K = 1 AND GP = G \pmod{P}

FOR = 1 TO T DO

SET AP = GP(I), BP = EQUAL TO THE REMAINDER OF CP/AP \pmod{P}

CP = MREDPO(EXPAND(C), P)

K = 1

DO I = 1, T

GP(I) = MREDPO(EXPAND(G(I)), P)

DO END

I = T - 1

DO I = 1, T

AP = GP(I)

AREM(CP, AP, X(1), BP)

BP = MREDPO(EXPAND(BP), P)

J = DEG(AP, X(1)) + 1

JI = DEG(BP, X(1)) + 1

CALL PROCEDURE PEGCDX TO COMPUTE SP, TP SUCH THAT

\( GP(I) \times SP + BP \times TP = 1 \pmod{P} \)

MULTIPLYING BOTH SP AND TP BY THE RECIPROCAL OF JJ \( \pmod{P} \) AND

ASSIGN THE NEW VALUES TO SP AND TP

NOW GP(I) \times SP + BP \times TP = 1 \pmod{P}

PEGCDX(AP, BP, J, JI, JJ, K, SP, TP, N)

JJ = IRECS(JJ, P)

SP = MREDPO(Expande(JJ*SP), P)

TP = MREDPO(Expande(JJ*TP), P)
CALL PROCEDURE PFHI TO COMPUTE A, B SUCH THAT C = A + B (MOD M)

SET F(I) = A, C = B AND CP = BP AND CONTINUE LOOPING.

PFH1(P, M, C, AP, BP, SP, TP, A, B, N)
F(I) = A
C = B
CP = BP
DO END

SET Z EQUAL TO THE LEADING COEFFICIENT OF C, THEN F(T) EQUAL TO MULTIPLICATION OF RECIPROCAL Z AND C (MOD M)

Z = COEFPO (EXPAND(C), X(1), DEG(C, X(1)))
LC = IRECS(Z, M)
A = LC*C
F(I) = MREVPO (EXPAND(A), M)
RETURN(F)
END

input * P prime number
M modulus number
C univariate polynomial
AA univariate polynomial
BB univariate polynomial such that C = AA * BB (mod P)
S univariate polynomial
T univariate polynomial such that
AA + S + BB * T = 1 (mod P)
N number of variables
output A univariate polynomial
B univariate polynomial such that
C = A * B (mod M)

integer P, Q, M, Q2, QT, N
algebraic (X(N)) C, AA, BB, S, T, A, 3, U, AT, BT, ST, TT, Y, Z, AS, BS, TM
algebraic_altran Mk_euqo
algebraic_altran P_seq1
value P, M, C, AA, BB, S, T

set Q = P, A = AA (mod P)
B = BB (mod P)
S = S (mod P)
T = T (mod P)

if Q is equal to M then end, else set U = (C - A * B) / Q, Q2 = Q**2

A = Mk_euqo (expand (AA) , P)
B = Mk_euqo (expand (BB) , P)
S = Mk_euqo (expand (S) , P)
T = Mk_euqo (expand (T) , P)
Q = P
l1
if (Q <= M) go to L3
return (A, B)
l2
Q2 = (C - A * B) / Q

if Q2 is greater than M then set QT = M / Q,
AT = A (mod QT), BT = B (mod QT)
ST = S (mod QT), TT = T (mod QT) and
call procedure P_seq1 to compute Y, Z such that
AT + Y + BT * Z = U (mod QT) then go to step L4, else

go to L3

if (Q2 <= M) go to L3
QT = M / Q
AT = Mk_euqo (expand (A) , QT)
BT = Mk_euqo (expand (B) , QT)
ST = Mk_euqo (expand (S) , QT)
TT = Mk_euqo (expand (T) , QT)
CALL Procedure PSEQT to compute Y, Z such that A*Y+B*Z=U (MOD Q)

L3^ PSEQT(Q, A, B, S, T, U, Y, Z, N)

SET AS=Q*Z+AS, BS=Q*Y+BS. IF Q+Q, LT, N GO TO L5
A=AS, B=BS, RETURN(A, B)
TM=(AS*S+BS*T-1)/Q
PSEQT(Q, A, B, S, T, TM, AT, BT, N)
S=S-1/AT
T=T-Q*BT
Q=Q2, A=AS, B=BS, GO TO L1
END
PROCEDURE PEGCDX(AA, BB, K, M, JJ, JK, R, S, N)

INPUT
AA UNIVARIATE POLYNOMIAL
BB UNIVARIATE POLYNOMIAL
N NUMBER OF VARIABLES
M DEGREE OF BB
K DEGREE OF AA
JK INTEGER

OUTPUT
R UNIVARIATE POLYNOMIAL
S UNIVARIATE POLYNOMIAL
JJ INTEGER SUCH THAT
R*AA+S*BB=JJ*X**JK
WHERE THE DEGREE OF R IS LESS THAN
THE DEGREE OF BB AND DEGREE OF S IS LESS THAN THE DEGREE OF AA.

ALGEBRAIC ARRAY ALTRAN(x(N)) AA, BB, R, S, Z, C, Y, A, B
ALGEBRAIC ARRAY ALTRAN POCEF
ARRAY(1AM) A
ARRAY(1AM) B
ARRAY(1AMN, 1AMN) Z
ARRAY(1AMN) C
ARRAY(1AMN) Y
VALUE A, BB, K, M, JK

IF DEG(BB) = 0 SET R=0, S=1, F=BB AND RETURN ELSE
SET VECTORS A AND B TO THE COEFFICIENT OF AA AND BB RESPECTIVELY
AND COMPUTE THE MATRIX Z SUCH THAT

A(0) A(1) A(2) ... A(J) A(N) A(N+1) ... A(N+M)
B(0) B(1) B(2) ... B(J) B(M) B(M+1) ... B(M+N)

IF (DEG(BB, X(1)), NE, 0) GO TO L1

K=0
S=1
JJ=1
RETURN(R, S, JJ)

L1
DO I=1,K
Z(I+J-1,J)=A(I)
DOEND
DOEND

SET C(JK)=1
SOLVE SYSTEM OF LINEAR EQUATIONS Z*Y=C
WHERE C IS THE CONSTANT VECTOR

DO J=1,MN
JC=J-M
DO I=1,M
Z(I+JC,J)=B(I)
DOEND
DOEND
C(JK)=1
Y=SOLVE(Q(Z,C,MN,N)

CONSTRUCT THE POLYNOMIALS R AND S SUCH THAT THE FIRST N ELEMENTS ARE THE COEFFICIENTS OF R AND THE REMAINDER ARE THE COEFFICIENTS OF S.

MM=MM-1
NN=K-2
DO J=MM,0,-1
R=R+Y(MM-J)*X(J)**(MM-J)
DOEND
DO J=NN,0,-1
S=S+Y(NN-J)*X(J)**(NN-J)
DOEND
IC=IGCD(ADEN(R),ADEN(S))
JJ=ADEN(R)*ADEN(S)/IC
JK=ADEN(R)
R=NUM(R)*ADEN(S)/IC
S=JK*NUM(S)/IC
RETURN(R,S,JJ)
END
PROCEDURE PSEQT(Q, A, B, S, T, U, Y1, Z1, N)

INPUT Q THE MODULUS NUMBER
A UNIVARIATE POLYNOMIAL
B UNIVARIATE POLYNOMIAL
S UNIVARIATE POLYNOMIAL
T UNIVARIATE POLYNOMIAL
U UNIVARIATE POLYNOMIAL SUCH THAT A*S+B*T=1 (MOD Q)
N NUMBER OF VARIABLES
OUTPUT Y1 UNIVARIATE POLYNOMIAL
Z1 UNIVARIATE POLYNOMIAL SUCH THAT
A*Y1+B*Z1=U (MOD Q)

INTEGER N, Q
ALGEBRAIC(X(N)), A, B, S, T, U, M, Z1, Y1, Q1, V, W
ALGEBRAIC ALTRAN MREDPO
INTEGER ALTRAN IRECS
VALUE Q, A, U, S, T, U

SET W=U MODULO Q, V=T+W AND Q1=QUOTIENT OF V/A, Z1=REMAINDER OF V/A.
REASSIGN Q1 BY MULTIPLYING Q1 BY THE RECIPROCAL OF ITS
DENOMINATOR MODULO Q, AND REPEAT THIS SAME STEP FOR Z1.

W=MREDPO(EXPAND(U),Q)
V=MREDPO(EXPAND(T+W),Q)
Q1=QMOD(V,A,X(N),Z1)
Q1=ANUM(Q1)*IRECS(I prep(ADEN(Q1),Q),Q)
Q1=MREDPO(EXPAND(Q1),Q)
Z1=ANUM(Z1)*IRECS(I prep(ADEN(Z1),Q),Q)
Z1=MREDPO(EXPAND(Z1),Q)

SET Y1=S*W+B*Q1 MODULO Q THEN END

V=S*W+B*Q1
Y1=MREDPO(EXPAND(V),Q)
RETURN(Y1, Z1)
END
PROCEDURE TRUFAC(U,H,P,RR,N,PQ,NI)

INPUT U INPUT MULTIVARIATE POLYNOMIAL
H INTEGER USED AS POWER OF IDEALS
P ARRAY CONTAINING MULTIVARIATE POLYNOMIALS
RR NUMBER OF POLYNOMIALS IN ARRAY P
N NUMBER OF VARIABLES
PQ THE MODULUS NUMBER
NI DEGREE OF U WITH RESPECT TO X(1)

INTEGER H, JK, JJ, RR, N, I, M, J, PQ, IR, U1, NI
EXTERNAL INTEGER LK
ALGEBRAIC (X(N))P, U, Y, R, US, Z, FAC, L, CP, YP, E, EE
EXTERNAL INTEGER IH
ARRAY(IANI) P
ARRAY(IANI) L
ARRAY(IANI) FAC
ALGEBRAIC ALTRAN MDSRKP
ALGEBRAIC ALTRAN MREDPO
INTEGER ALTRAN FACT
ALGEBRAIC ALTRAN XORDER
ALGEBRAIC ALTRAN LIST
ALGEBRAIC ALTRAN PCONT
VALUE U, H, P, RR, N, PQ

OBTAIN THE DIRECT TRUE FACTORS, FOR I=1 TO RR DO,
SET US EQUAL TO U TIMES ITS LEADING COEFFICIENT
Z EQUAL TO P(I) TIMES MULTIPLICATION OF ALL THE LEADING
COEFFICIENTS OF POLYNOMIALS P(1), ...., P(RR) EXCEPT THE LEADING
COEFFICIENT OF P(I).
IF THE REMAINDER OF US/Z IS ZERO PLACE Z ON THE LIST FAC,
SET US=US/P(Z) AND CONTINUE LOOPI findet, WHERE PP IS THE PRIMITIVE
PART OF THE GIVEN POLYNOMIAL ELSE INSERT P(I) INTO VECTOR L AND
CONTINUE LOOPI findet.

FAC=0
L=0
JK=0
JJ=0
DO I=1, RR
US=COEFP(C: EXPAND (U)), X(1), DEG(U, X(1))) * U
Z=1
DO J=1, RR
IF (J, NE, I) Z=Z*COEFP(C: EXPAND (P(J)), X(1), DEG(P(J), X(1)))
DOEND
Y=Z*P(I)
IF (H, EQ, 1) GO TO LM1
Y=MDSRKP(Y, H, N)
Y=MREDPO (EXPAND (Y), PQ)
AQUO (US, Y, X(1), R)
IF (R, EQ, U) GO TO L1
JJ=JJ+1
L(JJ)=P(I)
GO TO L2
L1^ CP=PCONT(Y,N)
JK=JK+1
FAC(JK)=Y/CP
U=U/FAC(JK)
L2^ CONTINUE
UEND
V
V
V
V
V
V
---
IF VECTOR L IS EMPTY , THEN END, ELSE IF L CONTAINS THE NUMBER OF POLYNOMIALS LESS THAN FOUR, PLACE U ON VECTOR FAC AND END,
ELSE SET N=1, IR=NUMBER OF NONZERO ELEMENTS IN L, U1=DEGREE OF U OVER TWO AND US=U*LOC (U)
INCREASE THE NUMBER OF COMBINATIONS M BY ONE FOR THE POLYNOMIALS IN ONE OF THE TRUE FACTORS, IF U IS EQUAL TO ONE THEN END, ELSE IF M IS GREATER THAN OR EQUAL TO (IR-1), OR M IS GREATER THAN U1/2, PLACE U ON VECTOR FAC AND END.
---
IF(JJ, EQ, 0) RETURN (FAC)
IF(JJ, GE, 4) GO TO L4
L3^ FAC(JK+1)=U
RETURN (FAC)
L4^ N=1
IR=JJ
U1=DEG(U,X(1))
US=COEFPQ(EXPAND(U),X(1),DEG(U,X(1)))*U
M=M+1
L5^ IF(U, EQ, 1) RETURN (FAC)
IF((M, GE, (IR-1)), OR, (2*M, GT, U1)) GO TO L3
IH=FACT(IR)/(FACT(M)*FACT(IR-M))
V
V
V
V
V
---
SELECT COMBINATION OF POLYNOMIALS, SET IH EQUAL TO THE COMBINATION OF M OUT OF IR ELEMENTS,
CALL PROCEDURE LLIST TO OBTAIN A NEW COMBINATION OF POLYNOMIALS WHERE THE DEGREE OF E IS LESS THAN U1, ALSO OBTAIN EE FROM PROCEDURE LLIST, WHERE EE IS THE MULTIPLICATION OF THE REMAINING (IR-M) LEADING COEFFICIENTS, IF E IS EQUAL TO ZERO, PLACE U ON FAC THEN END, ELSE SET Y=EE (MOD(PQ,S**H)), CP=CONTENT OF Y, YP=Y/CP AND R EQUAL TO THE QUOTIENT OF US/YP.
IF R IS NOT EQUAL TO ZERO RETURN TO STEP L7, ELSE INSERT YP INTO FAC, U=U/YP, U1=DEGREE OF U AND IR=IR-M
IF IR IS EQUAL TO ZERO RETURN TO STEP L5, ELSE SET U=U TIMES THE LEADING COEFFICIENT OF THE LAST POLYNOMIAL IN VECTOR FAC.
---
L7^ LLIST(U1,L,N,M,IR,RR,NL,E,EE)
IH=IH-1
IF(E, EQ, 0) GO TO L3
Y=EE
IF(H, EQ, 1) GO TO LM2
Y=MORPK(Y,H,N)
LM2^ Y=MORPO(EXPAND(Y),PQ)
CP=PCONT(Y,N)
YP=Y/CP
AQUGUS,YP,X(1),R)
IF(R, NE, 0) GO TO 'L7
FAC(JK)=YP
U=U/YP.
U1=DEG(U,X(1))
IR=IR-M
IF(IR<0)GO TO L5
U=COEFP0(EXPAND(FAC(JK)),X(1),DEG(FAC(JK),X(1)))*U
IF(H.EQ.1)GO TO LM3

SET U=1
MOD (PQ,S**t)+
DELETE ALL THE POLYNOMIALS THAT ARE USED TO CONSTRUCT E FROM
VECTOR L USING PROCEDURE AZERO, ALSO DELETE ANY POLYNOMIAL
WITH DEGREE GREATER THAN U1/2 FROM VECTOR L, AND RETURN
TO STEP L6.

-----------------------------
-----------------------------
-----------------------------
-----------------------------
-----------------------------
-----------------------------

LM3A
U=MODPK(U,H,N)
U=HREDPO(EXPAND(U),PQ)
DO I=1,RR
IF(2*DEG(L(I),X(1))GT,U1)L(I)=0
END
L=AZERO(L,H,N,NI)
L=XORDER(L,RR,N,NI)
GO TO L6
END
PROCEDURE AZERO(L,N,N,RR)

INPUT L ARRAY CONTAINS MULTIVARIATE POLYNOMIALS
      RR ARRAY SIZE
      N NUMBER OF VARIABLES
      M NUMBER OF POLYNOMIALS IN ARRAY L
      C ARRAY CONTAINING INTEGERS USED AS POINTERS TO POLYNOMIALS
      STORED IN ARRAY L

OUTPUT L ARRAY CONTAINING MULTIVARIATE POLYNOMIALS AFTER SETTING TO ZERO THOSE LOCATION POINTED TO BY VECTOR C.

ALGEBRAIC (X(N)) L
EXTERNAL INTEGER ARRAY(1AM) C
ARRAY(1ARR)L
INTEGER M,N,J,I,RR
VALUE L,M,N
DO I=1,M
    J=C(I)
    L(J)=0
    DO END
RETURN(L)
END

PROCEDURE XORDER (L,RR,N,NI)

INPUT L ARRAY CONTAINING MULTIVARIATE POLYNOMIALS
      RR NUMBER OF POLYNOMIALS IN ARRAY L
      N NUMBER OF VARIABLES
      NI ARRAY SIZE

OUTPUT LL ARRAY CONTAINING NONZERO MULTIVARIATE POLYNOMIALS
      VECTOR L

ALGEBRAIC (X(N)) L, LL
INTEGER I,N,J,RR,NI
ARRAY(1AI) L
ARRAY(1AI)LL
J=0
LL=0
DO I=1,RR
   IF(L(I),EQ,0)GO TO L1
   J=J+1
   LL(J)=L(I)
CONTINUE
DOEND
RETURN (LL)
END
PROCEDURE XPOINT (M1,N1,M)

INPUT - M1 NUMBER OF POINTERS USED FOR TEST
       N1 NUMBER OF FACTORS
       M NUMBER OF POINTERS IN ARRAY C
       C EXTERNAL ARRAY USED AS POINTERS
OUTPUT C NEW COMBINATION OF M FACTORS USING VECTOR C TO POINT TO
       THEIR LOCATIONS.

 INTEGER M1,N1,M1,I
 EXTERNAL INTEGER ARRAY (1^M)C
 VALUE M1,N1,M

 FOR I=1 TO M DO
   IF C(I) IS NOT EQUAL TO (N1-M+1) THAT IS NOT EQUAL TO THE MAXIMUM
   POSSIBLE VALUE, GO TO STEP L1, ELSE CONTINUE LOOPING.

   DO I=1, M
   IF(C(I) .NE. (N1-M+1)) GO TO L1
   DO END
   RETURN

 IF C(M1) IS NOT EQUAL TO (N1-M+1) SET C(M1)=C(M1)+1 AND END, ELSE
 CALL RECURSIVELY PROCEDURE XPOINT WITH ARGUMENTS M1-1,N1,M
 TO CHANGE THE VALUE OF POINTER C(M1-1), SET C(M1)=C(M1-1)+1, THEN END

 L1^ IF(C(M1) .EQ. (N1-M+M1) ) GO TO L2
     C(M1)=C(M1)+1
     RETURN
 L2^ XPOINT(M1-1,N1,M)
     C(M1)=C(M1-1)+1
     RETURN
 END
PROCEDURE MULT(M, A, N1, N)

INPUT M NUMBER OF POLYNOMIALS
A ARRAY CONTAINING MULTIVARIATE POLYNOMIALS
N1 ARRAY SIZE
N NUMBER OF VARIABLES
C EXTERNAL ARRAY USED AS POINTERS FOR ARRAY A SUCH THAT
E=B(C(1))*B(C(2))*...*B(C(N))

OUTPUT B MULTIVARIATE POLYNOMIAL EQUAL TO THE MULTIPLICATION
OF ALL B(C(J)) POLYNOMIALS, WHERE J=1,2,...,M

INTEGER I, M, J, N1, N
EXTERNAL INTEGER ARRAY(1M) C
ALGEBRAIC (X(N)) A, B
ARRAY(1N1) A
VALUE M, A, N1, N
B=1
DO I=1, M
J=C(I)
B=B+A(J)
END
RETURN(B)
END

PROCEDURE FACT(N1)

INPUT N1 POSITIVE INTEGER

OUTPUT FACTORIAL N1 SUCH THAT FACT=N1*(N1-1)*...*2*1, WHERE
N1 GREATER THAN ZERO AND EQUAL 1 WHEN N1 IS EQUAL TO ZERO

INTEGER N1, FACT
VALUE N1
IF(N1.EQ.0)RETURN(1)
RETURN(N1*FACT(N1-1))
END
PROCEDURE LLIST(U1,A,N,M,R,NR,NI,ALGEBRAIC(X(N)),J,LOC,H)

INPUT U1 INTEGER
M NUMBER OF VARIABLES
A ARRAY CONTAINING MULTIVARIATE POLYNOMIALS
ALGEBRAIC(X(N)) INTEGER ALTRAN MULT
ALTRAN XPOINT

OUTPUT LB MULTIVARIATE POLYNOMIAL EQUAL TO MULTIPLICATION OF THE
M CHOSEN POLYNOMIALS SUCH THAT ITS DEGREE IS LESS THAN U1/2
LOC MULTIPLICATION OF THE LEADING COEFFICIENT OF THE REMAINING
(NR-M) POLYNOMIALS IN ARRAY A.

INTEGER U1,N,M,R,I,J,NR,NI
ALGEBRAIC(X(N)) INTEGER ALTRAN MULT
ALTRAN XPOINT

SET LB=0 AND LOC=1
FOR J=1 TO IH DO
    IF C=0 SET C(L)=L FOR L=1,2,...,M THEN GO TO L2, ELSE GO TO L1 AND
    CALL PROCEDURE XPOINT TO COMPUTE THE INDICES OF THE M POLYNOMIALS
    SET H EQUAL TO MULTIPLICATION OF THE M POLYNOMIALS. IF DEGREE
    OF H IS GREATER THAN U1/2 CONTINUE LOOPING FOR J.

LB=0
LOC=1
DO J=1,IH
    IF(C,N)=0 GO TO L1
    DO I=1,M
        C(I)=I
    DOEND
    GO TO L2
L1^ XPOINT(M,R,M)
L2^ H=MULT(H,I,N)
    IF(2+DEGH(X(I)),LE,U1) GO TO L3
    DOEND
    GO TO L5
L3^ LB=H

SET LB=H
FOR I=1 TO R DO,
    IF C(IJ) IS NOT EQUAL TO I CONTINUE LOOPING OVER J, THEN SET LOC
**EQUAL TO LOC TIMES A(I), ELSE IF C(J) IS EQUAL TO I, CONTINUE**

**LOOPING OVER I, ON EXIT FROM THE OUTER LOOP RETURN VALUES LB, LOC**

---

```

DO I = 1, R
  DO J = 1, M
    IF (I .EQ. C(J)) GO TO L4
  END DO
  LOC = LOC * COEFPO(EXPAND(A(I)), X(1), DEG(A(I)), X(1))
  CONTINUE
```

```

RETURN(LB, LOC)
```

---

END
PROCEDURE EXZH (PQ, UA, H, A, IR, N, M, Y, J)

INPUT
PQ THE MODULUS NUMBER
UA MULTIVARIATE POLYNOMIAL
H ARRAY CONTAINS POLYNOMIALS SUCH THAT
UA(X(1), A(2), A(3), ..., A(N)) = H(1) * H(2) * ... * H(IR)
A ARRAY CONTAINING INTEGERS USED FOR SUBSTITUTION
IR NUMBER OF POLYNOMIALS IN ARRAY H
N NUMBER OF VARIABLES
M DEGREE OF UA WITH RESPECT TO X(1)
OUTPUT
Y ARRAY CONTAINS MULTIVARIATE POLYNOMIALS
J INTEGER SUCH THAT
UA = Y(1) * Y(2) * ... * Y(IR) (MOD (S^* J, PQ))
WHERE S = (X(2) - A(2), X(3) - A(3), ..., X(N) - A(N))

INTEGER N, IR, PQ, A, J, I, IR1, I1, M, J1, J2, J3, I1, I2, I3, I4, I5, I6, I7, I8, I9, I10
EXTERNAL INTEGER LK
ALGEBRAIC(X(N)) UA, U, H, Y, G, F, F2, G2, V, R1, R2, W1, ALPHA, BETA, C
ARRAY(1AM+1) ALPHA
ARRAY(1AM+1) BETA
ARRAY(1AN) A
ARRAY(1AM) H
ARRAY(1AM) Y
ALGEBRAIC ALTRAN PEGCDX
INTEGER ALTRAN IRCS
ALGEBRAIC ALTRAN MDSRPK
ALGEBRAIC ALTRAN MREQPO
ALGEBRAIC ALTRAN PSEQT
VALUE PQ, UA, H, A, IR, N
WRITE PQ, UA, H, A, IR, N

SET Y=0, LK=0 AND CALL PROCEDURE MDSRPK TO COMPUTE DEGREE
OF UA IN (X(2), X(3), ..., X(N))
SET IZ EQUAL TO ONE PLUS THE DEGREE OF UA IN (X(2), ..., X(N))

Y=0
LK=0
F=MDSRPK(UA, G, N)
IZ=LK+1
LK=1
IR1=IR-1

FOR I=1 TO (IR1-1) DO
SET F=H(I), G=H(I+1) * ... * H(IR) (MOD PQ)
U=UA(X(1), X(2) - A(2), ..., X(N) - A(N)) (MOD PQ),
R1=F*G-U (MOD PQ),
R1=F*G-U (MOD PQ), J3=DEGREE OF F + DEGREE OF G
FOR I=1 TO J3 DO
CALL PROCEDURE PEGCDX TO COMPUTE ALPHA(I), BETA(I) SUCH THAT
F*ALPHA(I)+G*BETA(I)=JJ*X(1) * ... * I, WHERE JJ IS AN INTEGER.
MODIFY ALPHA(I), BETA(I) SUCH THAT
ALPHA(I)=ALPHA(I)+RECIPROCAL(JJ) (MOD PQ)
DO L=1,IF1
  F=H(L)
  I1=L+1
  G=1
  DO J=I1,IR
    G=G+H(J)
  DOEND
  G=MREDPO(EXPAND(G),PQ)
  U=UA
  DO I=2,N
    U=U(X(I)=X(I)+A(I))
  DOEND
  U=MREDPO(EXPAND(U),PQ)
  R1=MREDPO(EXPAND(F*G-U),PQ)
  J1=DEG(F,X(1))+1
  J2=DEG(G,X(1))+1
  J3=J1+J2-2
  ALPHA=U
  BETA=0
  DO I=1,J3
    PEGCDX(F,G,J1,J2,JJ,I,ALPHA(I),BETA(I),N)
    IL=IRECS(JJ,PQ)
    ALPHA(I)=MREDPO(EXPAND(ALPHA(I)*IL),PQ)
    BETA(I)=MREDPO(EXPAND(BETA(I)*IL),PQ)
    WRITE ALPHA(I),BETA(I),G,F
  DOEND

SET J=2
CALL PROCEDURE MDSRPK TO COMPUTE W1 SUCH THAT
W1=R1 (MOD S**J)
SET F2=F2,G2=G AND J1=DEGREE OF W1 WITH RESPECT TO X(1)
IF J1 IS EQUAL TO J3 CALL PROCEDURE PSEQT TO COMPUTE ALPHAI(J3+1),
BETA(J3+1),
FOR K=0 TO J1 DO
  F2=F2- BETA(K)*CW1(K) (MOD PQ)
  G2=G2-ALPHA(K)*CW1(K) (MOD PQ)
  WHERE W1=SUM (CW1(I)*X(1)**I),II=0,1,...,J1
CONTINUE LOOPING OVER K.

J=2
L1^ W1=MDSRPK(R1,J,N)
F2=F2
G2=G
J1=DEG(W1,X(1))
WRITE W1
IF(J1,LT,J3)GO TO L3
PSEQT(PQ,G,F,BETA(1),X(1)**N,BETA(J3+1),ALPHA(J3+1),N)
ALPHA(J3+1)=ANUM(ALPHA(J3+1))*IRECS(ADEN(ALPHA(J3+1)),PQ)
BETA(J3+1)=ANUM(BETA(J3+1))*IRECS(ADEN(BETA(J3+1)),PQ)

L3^ DO I=2,J1
  R1=COEFPC(EXPAND(W1),X(1),I)
  K2=MREDPO(EXPAND(R1*BETA(1+I)),PQ)
  F2=F2-R2
  G2=MREDPO(EXPAND(R1*ALPHA(I+1)),PQ)
119 WRITE F2, G2
120 DO END
121 F2 = MREDPO(REDAND(F2), P1)
122 G2 = MREDPO(REDAND(G2), PQ)

v

v

SET R1 = F2 + G2 - U (MOD PQ)

IF R1 IS EQUAL TO ZERO OR J IS GREATER THAN IZ GO TO L2, ELSE

SET J = J + 1, F = F2, G = G2, AND RETURN TO STEP L1.

SET U = G2; Y(L) = F2 AND CONTINUE LOOPING FOR L

SET Y(IF) = G2 THEN END.

v

v

R1 = F2 * G2 - U
R1 = MREDPC(REDAND(R1), PQ)
IF (R1, EQ, U) GO TO L2
IF (J, GE, IZ) GO TO L2
J = J + 1
F = F2
G = G2
GO TO L1

Y(L) = F2
U = G2
DO END
Y(IR) = G2
DO I = 1, IR
DO J2 = 1, N
Y(I) = Y(I) + (X(J2) = X(J2) - A(J2))
DO END
DO END
RETURN (Y, J)
END
PROCEDURE MDSRPK(F,K,N)

INPUT  F  MULTIVARIATE POLYNOMIAL
       K  INTEGER
       N  NUMBER OF VARIABLES

OUTPUT H1 MULTIVARIATE POLYNOMIAL SUCH THAT
      H1=F (MOD S**K)
       LK INTEGER USED TO COMPUTE DEGREE OF F IN(X(2),X(3),...,X(N))

ALGEBRAIC (X(N))F,F1,FX,H,H1
INTEGER I,J,M,DSUM,D,K,N
EXTERNAL INTEGER LK
ARRAY(2AN)D
ALGEBRAIC ALTRAN EXPWR
VALUE F,K,N

SET F1 EQUAL TO THE COEFFICIENT OF X(1)**J.
COMPUTE THE POWERS OF THE VARIABLES IN THE FIRST TERM
OF F1 AFTER PLACING IT IN A CANONICAL FORM USING PROCEDURE
EXPWR AND SET DSUM EQUAL TO THIS SUMMATION. FROM EXPWR WE OBTAIN
THE INTEGER COEFFICIENT FX. SET FX=FX*X(I)**D(I). WHERE D(I) IS
THE EXPONENT OF X(I).

H1=0
M=DEG(F,X(1))
DO J=N,0,-1
H=H
F1=COEFCYXC(EXPWR(F),X(1),J)
EXPWR(F1,N,0,FX)
DSUM=0
IF(FX.EQ.0)GO TO L2
DO I=2,N
DSUM=DSUM+D(I)
FX=FX*X(I)**D(I)
END

IF LK IS GREATER THAN ZERO TEST IF DSUM IS GREATER THAN LK. IF TRUE
SET LK=DSUM.

IF(LK.LT.D)GO TO L12
IF(LK.LT.DSUM)LK=DSUM

CONTINUE
DO END

SET F1=F1-FX. IF DSUM IS LESS THAN K SET H=H+FX.
IF F1 IS NOT EQUAL TO ZERO RETURN TO STEP L1; ELSE
SET H1=H1+H*X(1)**J. IF J=0 THEN END; ELSE J=J-1 AND RETURN TO
COMPUTE F1.

L2^ F1=F1-FX
PROCEDURE EXPWR (FF,N,01,FZ)

INPUT  FF   MULTI-VARIATE POLYNOMIAL
  N   NUMBER OF VARIABLES
OUTPUT D1   ARRAY CONTAINING POWERS OF X(I)
       FZ   INTEGER COEFFICIENT OF FF

ALGEBRAIC (X(N))FF,FZ
INTEGER K,01,IN
ARRAY (2AN)01
VALUE FF,N
01=0
DO IN=2,N
  D1(IN) = DEG(FF,X(IN))
  FF = COEFP(C( EXPAND(FF),X(IN),D1(IN))
  IF (FF·EQ.0) GO TO L2
END
FZ=FF
RETURN(01,FZ)
END
APPENDIX D

A Listing of the program INTRPT

The following procedure are listed:

1. INTRPT
2. TRPT
3. PFDEC
PROCEDURE INTERP(N,H,TP,F3)

INPUT  TP   THE TRANSCENDENTAL FUNCTION
       F3   ARRAY CONTAINS REDUCIBLE POLYNOMIALS
OUTPUT COEF1 ARRAY CONTAINS COEFFICIENTS OF THE LOGARITHMIC FUNCTION
       COEF2 ARRAY CONTAINS COEFFICIENTS OF THE INVERSE ARC TANGENT FUNCTION
       XARTN ARRAY CONTAINS THE ARGUMENT OF THE INVERSE ARC TANGENT FUNCTION
       XS   ARRAY CONTAINS THE VALUES THAT THE COEFFICIENT AND THE
             ARGUMENT OF THE INVERSE ARC TANGENT MUST BE DIVIDED BY
             ITS SQUARE ROOT.
       L   ARRAY CONTAINS TRANSCENDENTAL FUNCTION WHICH IS NOT
             ABLE TO BE INTEGRATED OVER THE RATIONAL FIELD SUCH THAT
       INTEGRAL(2) = SUM(COEF1(J) * LOG(XLOG(I)))
       + SUM(COEF2(J) * ARCTAN(XARTN(J)) / SQUARE(XS(J))) / SQUARE(XS(I))
       WHERE SQUARE IS THE
            SQUARE ROOT FUNCTION.

INTEGER N,1,J
ALGEBRAIC(DP,TP,F3,S,A,Z,COEF1,XLOG,XARTN,XS,COEF2,L,H)
ARRAY(IAN)XS
ARRAY(IAN)XLOG
ARRAY(IAN)XARTN
ARRAY(IAN)COEF1
ARRAY(IAN)COEF2
ARRAY(IAN)A
ARRAY(IAN)F3
ARRAY(IAN)L
ALGEBRAIC ALTRAN CONT
ALGEBRAIC ALTRAN TRPT
ALGEBRAIC ARRAY ALTRAN PFDEC

SETS S TO THE NUMERATOR OF TP AND H TO THE CONTENT OF THE
TP DENOMINATOR.
CALL PROCEDURE PFDEC TO OBTAIN THE PARTIAL FRACTION
TERM SUCH THAT   TP = SUM(A(I) / X3(I))

J=0
S=NUM(TP)
H=CONT(ADEN(TP),H)
A=PFDEC(ADEN(TP) / H,F3,N,S,H)

FOR I=1 TO N DO
SET Z=A(I) / X3(I)
   CALL PROCEDURE TRPT TO COMPUTE COEF1(I), COEF2(I), XLOG(I)
   XARTN(I),XS(I).
   IF COEF1(I) = COEF2(I) = 0, NO INTEGRATION CAN BE DONE
   WITHOUT EXTENSION FOR THE RATIONAL FIELD, ADD Z/H TO L,
   ELSE DIVIDE COEF1(I), COEF2(I) BY H AND PRINT RESULTS
   AND CONTINUE LOOPING.
WRITE (TP
DO I = 1, N
IF (A(I).EQ. L) GO TO L2
Z = A(I)/FB(I)
TRPT(Z, M, COEF1(I), XLOG(I), COEF2(I), XARTN(I), XS(I))
COEF1(I) = COEF1(I)/H
COEF2(I) = COEF2(I)/H
IF (COEF2(I).EQ.0).AND. (COEF1(I).EQ.0)) GO TO L1
IF (COEF1(I).NE.0) WRITE COEF1(I), XLOG(I)
IF (COEF2(I).NE.0) WRITE COEF2(I), XARTN(I), XS(I)
GO TO L2
L1A J = J + 1
L(J) = Z/H
WRITE L(J)
L2A CONTINUE
DO END
WRITE 
WHERE INTEGRAL (S(X)) = SUM (COEF1(I)*LOG (XLOG(I)))
I=1
WRITE 
WHERE M2
WRITE + SUM (COEF2(I)/SQRT(XS(I)) + ARCTAN (XARTN(I)/SQRT(XS(I))))
I=1
WRITE 
WHERE M2
WRITE + SUM (INTEGRAL L(I))
I=1
RETURN
END
PROCEDURE TKPT(A,M,CO1,XLN,CO2,XART,Z)

INPUT A  
M  
CO1  
CO2  
XLN  
XART  
Z  

PURE TRANSCENDENTAL PART S(I)/T(I)
NUMBER OF VARIABLES
COEFFICIENT OF THE LOGARITHMIC TERM
COEFFICIENT OF THE INVERSE ARCTAN TERM
ARGUMENT OF THE LOGARITHMIC FUNCTION
ARGUMENT OF THE INVERSE ARCTAN FUNCTION
ELEMENT OF I(X(2),X(3),...,X(N)). THIS TERM BOTH
COE2,XART MUST BE DIVIDED BY ITS SQUARE ROOT.

INTEGER N,M
ALGEBRAIC (X(M))A,XLN,XART,CO1,CO2,Z,F,P,Q,C,N1,XM
ALGEBRAIC ALTRAN DIFFX

SET Z=CO1=CO2=XLN=XART=0
SET F=S(I)/DIFFX(T), WHERE DIFFX IS THE DERIVATIVE WITH
RESPECT TO X(I).
IF DEGREE OF F IS EQUAL TO ZERO, SET CO1=F,XLN=T(I) AND GO
TO L2, ELSE SET XM=0.

Z=0
CO1=0
CO2=0
XLN=0
XART=0
F=ANUM(A)/DIFFX(ADEN(A),X(I))
IF ((DEG(ANUM(F),X(I)),NE,0), OR (DEG(ADEN(F),X(I)),NE,0)) GO TO L1
CO1=F
XLN=ADEN(A)
GO TO L2
L1A

IF DEGREE OF T(I) NOT EQUAL TO 2 GO TO STEP L2
IF DEGREE OF S(I)=2, SET XM TO THE COEFFICIENT OF X(1) IN S(I),
N1 TO THE CONSTANT TERM IN S(I), C TO X(1)**2 COEFFICIENT IN T(I)
P TO X(1) COEFFICIENT IN T(I) AND Q TO THE CONSTANT TERM OF T(I).
SET CO1=XM/(2*C), XLN=S(T(I),Z4+Q*C-P+P, CO2=(Z4+N1*C-P+XM)/C
AND XAR=Z4*P+C+P THEN END.

L1B

IF (DEG(ADEN(A),X(I)),NE,2) GO TO L2
IF (DEG(ANUM(A),X(I)),EQ,1) XM=COEFPD( EXPAND(ANUM(A)),X(1),1)
N1=COEFPD( EXPAND(ANUM(A)),X(1),0)
C=COEFPD( EXPAND(ADEN(A)),X(1),2)
P=COEFPD( EXPAND(ADEN(A)),X(1),1)
Q=COEFPD( EXPAND(ADEN(A)),X(1),0)
CO1=XM/(2+C)
XLN=ADEN(A)
Z=4*Q+C-P+P
CO2=(Z4+N1*C-P+XM)/C
PROCEDURE PFDEC(B,X,N,S,M)

INPUT B  MULTIVARIATE POLYNOMIAL
       X  ARRAY CONTAINS POLYNOMIALS
       N  DEGREE OF B
       M  NUMBER OF VARIABLES
OUTPUT A  ARRAY CONTAINS POLYNOMIAL SUCH THAT
       S/B= SUM (A(I)/XB(I))

INTEGER N,K,I,IR, NB,NH,J,JK,L,M
ALGEBRAIC(X(N)) B,X,N,F,XM,Z,A,S
ARRAY(1AN) H
ARRAY(1AN) F
ARRAY(1AN) XB
ARRAY(1AN) A
ARRAY(1AN,1AN) XM
ALGEBRAIC ARRAY ALTRAN POCEF
ALGEBRAIC ARRAY ALTRAN ASOLVE

SET IR EQUAL TO THE NUMBER OF POLYNOMIAL IN VECTOR B, NK=1
FOR I=1 TO IR DO,
SET NB EQUAL TO THE DEGREE OF XB(I), Z=B/XB(I) AND NH=NK+NB-1
FOR J=NK TO NH DO
SET JK=J-NK, VECTOR H TO THE COEFFICIENT OF THE TERM (Z*X(1)+JK)
PLACE H IN THE (NH-JK) TH COLUMN IN MATRIX XM AND CONTINUE LOOPING
FOR J,
SET NK=I+1 AND CONTINUE LOOPING FOR I.

A=0
DO IR=1,N
   IF(X(1)=0,0,0) GO TO L1
   DO L=1,IR
      IF(X(1)=0,0,0) GO TO L1
      N=DEG(B,X(1))
      NK=1
      DO L=1,IR
         NB=DEG(XB(I),X(1))
         Z=B/XB(I)
         NH=NK+NB-1
         DO J=NK,NH
            JK=J-NK
            H=POCEF(Z,JK,N,H)
            DO L=1,N
               XM(L,NH-JK)=H(L)
            DO
            NK=NH+1
   DO
   H=X(I)
SET H TO THE COEFFICIENT OF POLYNOMIAL S.
Solve the system of linear equations \( XM \cdot F = H \)

\[
\begin{align*}
M &= \text{POGEF}(S, \pi, N, H) \\
F &= \text{ASOLVE}(XM, H)
\end{align*}
\]

\[
\begin{align*}
\text{SET} &\text{ NK}=1 \text{ FOR } I=1 \text{ TO IR DO} \\
\text{SET} &\text{ NB=DEGB(I), NH=NK+NB-1} \text{ AND } A(2) = \text{SUM}( F(J) \cdot X(1) \cdot (N-B-JK-1) ) \\
\text{WHERE} &\text{ J} = \text{NK}, \ldots, \text{NH AND JK} = \text{J-NK} \\
\text{SET} &\text{ NK} = \text{NH+1} \text{ AND CONTINUE LOOPING FOR I.}
\end{align*}
\]

\[
\begin{align*}
\text{NK} &= 1 \\
\text{DO} I &= 1, \text{IR} \\
\text{NB} &= \text{DEGB}(\pi B(I), X(1)) \\
\text{NH} &= \text{NK} + \text{NB}-1 \\
\text{DO} J &= \text{NK}, \text{NH} \\
\text{JK} &= J - \text{NK} \\
A(I) &= A(I) + F(J) \cdot X(1) \cdot (N-B-JK-1) \\
\text{DO END} \\
\text{NK} &= \text{NH+1} \\
\text{DO END} \\
\text{RETURN}(A) \\
\text{END}
\end{align*}
\]