SYMDIP:

A COMPUTER PROGRAM FOR SYMBOLIC

DIFFERENTIATION AND INTEGRATION

by

KEK-WAN WANG
B.Sc. (Hons.), Nanyang University, Singapore, 1970

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in the Department

of

Mathematics

ACCEPTED
FACULTY OF GRADUATE STUDIES

[redacted]
DEAN

DATE 17 Nov/76

We accept this thesis as conforming to the required standard

[redacted]

UNIVERSITY OF VICTORIA

September 1976

ABSTRACT

Supervisor : Dr. B. Ehle

The goal of this thesis is the design and implementation of a computer program capable of computing symbolic derivatives and indefinite integrals. The program must be suitable for implementation on the University of Victoria IBM 370/145 computer with the VS-1 operating system. In particular, the program is required to satisfy both of the following conditions :

(a) It should be able to solve a differentiation problem or an integration problem using less than 1 minute of CPU time and less than 256K bytes of main storage when run on the university computer.

(b) Its design should make it suitable for use as a teaching tool in a typical first year calculus class.

The program, SYMDIP (symbolic differentiation and integration package), meets all of these basic design goals. Tests using a set of 30 problems taken from the 1975-76 fall and spring examination papers for Mathematics 130 show that all but one of these 30 problems can be sucessfully solved by the program. To illustrate that the package is capable of solving many problems which are beyond the capabilities of a first year student, a set of 18 more complicated differentiation and integration problems are also solved by SYMDIP.

The user states his problem in a simple and natural way as follows :

DIFFERENTIATE "function" WITH RESPECT TO "variable"

or

INTEGRATE "function" WITH RESPECT TO "variable" .

"Function" is the function to be differentiated or integrated and it is written using the FORTRAN notation. "Variable" is any single alphabetic character of the user's choice.

The complete program is composed of 3 separate job steps : the analyzer, the processor, and the writer. These job steps are named in accordance with their respective functions within the package. The analyzer and the writer are written in the SNOBOL4 language while the main body of the program, the processor, is written using the algebraic programming language ALTRAN. Because SYMDIP requires 354K bytes of main storage space for execution, an overlay structure is constructed to be used with the program. Storage requirements of SYMDIP are reduced to 256K bytes of main memory in this way. Design of this structure is complicated by the relatively large number (approximately 300) of subprocedures involved. The average user is completely unaware of both the overlay structure and the separate job steps because he activates the package using a simple catalogued procedure.

Examiners [redacted]

TABLE OF CONTENTS

Page

CHAPTER ONE

The Problem 1

1.1. Introduction 2
1.2. Algebraic Packages and Languages 4
1.3. ALTRAN 11
1.4. The Problem 20

CHAPTER TWO

Package Capabilities and Limitations 22

2.1. Introduction 23
2.2. Differentiation 28
2.3. Integration 28
2.4. Error Handling Within The Package 40
2.5. Package Usage Suggestions 42

CHAPTER THREE

Package Design 45

3.1. Introduction 46
3.2. The Analyzer 51
3.3. The Processor - Differentiation 73
3.4. The Processor - Integration 82
3.5. The Writer 98
3.6. Interfaces Between Job Steps 112

CHAPTER FOUR

Overlay Structure 114

4.1. Introduction 115
4.2. Multi-Region Overlays and The VS-1 Operating System 122
4.3. Constructing An Overlay Structure 129
4.4. Tests Results Contrasting Various Modes of Operation 132

CHAPTER FIVE

Conclusions 135

5.1. Accomplishments 136
5.2. Extensions 137

REFERENCES 140

APPENDICES 142
APPENDIX A. The Analyzer(The Preprocessor in SNOBOL4) 143
APPENDIX B. ALTRAN Procedures and Their Descriptions 156
APPENDIX C. The Writer(The Postprocessor in SNOBOL4) 200
APPENDIX D. Procedures DIFF, PINT, SPLIT(ALTRAN Routines) . . . 213
APPENDIX E. Function Names and Their Representations 214
APPENDIX F. Special Functions 215
APPENDIX G. The Solutions of The Set of 30 Test Problems . . . 217
APPENDIX H. The Solutions of The 18 Harder Problems 223
APPENDIX I. JCL Requested of The SYMDIP User 228
APPENDIX J. The Complete JCL For SYMDIP 229
APPENDIX K. The Program For Creating Overlay Levels 230

LIST OF FIGURES

Figure		Page
1.1.	The polynomial POLY = $2x^3 - 4yz + x^2yz^3$ . .	14
3.1.	Output produced by the analyzer for Example 3.1.	70
3.2.	Output produced by the analyzer for Example 3.2.	71
3.3.	Output produced by the processor for Example 3.3. and Example 3.4.	72
3.4.	Output produced by the processor for Example 3.1.	80
3.5.	Output produced by the processor for Example 3.3.	81
3.6.	Output produced by the processor for Example 3.2.	97
3.7.	Output produced by the processor for Example 3.4.	97
3.8.	Temporary data sets used by SYMDIP.	112
4.1.	Turn-around statistics for class-C jobs of less than 5 min. CPU time.	116
4.2.	Turn-around statistics for class-B jobs of less than 5 min. CPU time.	118
4.3.	An acceptable overlay structure for Example 4.1.	124
4.4.	An acceptable overlay structure for Example 4.1.	126
4.5.	An acceptable overlay structure for Example 4.2.	127
4.6.	An acceptable multi-region overlay structure for Example 4.2.	128
4.7.	Comparison of overlay structures	133

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to my supervisor, Dr. B. Ehle, for his guidance and encouragement and for his unlimited patience. I also wish to thank the members of the Department of Mathematics and the members of the University of Victoria Computer Center for their valuable assistance and suggestions. Finally, I wish to thank both Mrs. M. Hicking and Mrs. B. Hames for their careful typing of the final manuscript.

CHAPTER ONE

THE PROBLEM

1.1 Introduction

1.2 Algebraic Packages and Languages

1.3 ALTRAN

1.4 The Problem

1.1. Introduction.

The goal of this thesis is the design and implementation of a computer program to perform symbolic differentiation and integration.

Thus, given the problems

$$\frac{d}{dx}\ (x^3 + 7)$$

and

$$\int \sin(x)\ dx$$

stated in a suitable form, the program would be expected to generate results like $3x^2$ and $-\cos(x)$ + constant, respectively.

The program is designed to be used primarily as a teaching tool while introducing the concepts of differentiation and integration to students. In order to make it an effective teaching tool, the program is designed to satisfy the following criteria:

(a) Students should be able to use the system with no more than one hour of instruction.

(b) The program should be able to solve a problem in less than 1 minute of CPU time using less than 256K bytes of memory space.

The first criterion is related to the needs of both the student and the instructor. If the time required to explain how to use the package is too great, the instructor may not be able to fit the explanation into his lecture schedule. If using the package is too complicated the student may not be willing to even try it.

The second criterion is one of economy. Both memory space and CPU time are among the most critical and expensive resources of a computer system. Thus, if many students want to solve problems using the program, the impact on the system should be minimal. Satisfying this constraint is essential if the program is to be usable.

The second criterion was the hardest to satisfy. This was partly due to the complexity and size of the algebraic language package employed and partly due to the complexities of computing indefinite integrals. It was in fact necessary to put more severe restrictions on the types of functions which could be integrated than it was on those which were to be differentiated. The resulting package, which we call SYMDIP (for symbolic differentiation and integration package), is able to solve most of the problems found in a typical first year calculus text. It is also able to solve many more difficult problems and might be of use in other areas.

In Section 2, we discuss some algebraic packages and languages that have been developed in the last few years. Emphasis will be placed on the general capabilities and limitations these packages and languages possess.

In Section 3, we discuss the programming language ALTRAN. The major portion of our program has been written using the ALTRAN language. As in Section 2, the emphasis will be on ALTRAN's capabilities and limitations.

In the last section of this chapter, we formally list the set of difficulties and problems that we have faced and solved.

1.2 . Algebraic Packages and Languages .

The computer's ability to perform large quantities of elementary arithmetic has made such machines indispensable in almost every area of science. However, it is only during the last decade that their ability to perform large amounts of elementary algebra has been exploited [2]. It is reasonable to expect that as their capability in this area is increased, their use for solving practical problems in science and engineering will also grow. There is no doubt that the facilities available today for the handling of routine algebra on computers are extremely elementary and for the most part rather crude [3]. Nonetheless, they have been used in a number of different branches of science and have made possible significant advances which, in their absence, would have been much harder [2].

Our contact with a present day algebraic system is generally via a programming language (ALTRAN, LISP or SNOBOL4, for example) whose structure is similar to languages intended for numerical calculation. However, algebraic programming languages differ from the conventional languages in that the "values" of the variables declared in the program are algebraic expressions represented formally within the computer memory .

Thus, two interpretations of the statement

$$Y = X + X$$

are possible. In the algebraic context, Y will have the string '2*X' as its formal value while in numerical calculation (as, for example, in PL/I or FORTRAN), the numeric value of Y is twice that of X.

The value of the variable, when printed, will also have two obviously distinct formats. Conventionally, it is the numeric value of the variable that is printed. In an algebraic system, however, the output will be an algebraic expression. This expression may be a simple numeric value or a more complex expression involving variables and numbers in some algebraic combination.

For a number of reasons, development of computer programs for manipulating algebraic operations has progressed slowly. Perhaps the most obvious difficulty facing someone wishing to construct an algebraic system is that the hardware of a computer is not designed to manipulate algebraic expressions. As a result, the very simple algebraic operations like addition and multiplication of two algebraic expressions must be handled by software programs. No algebraic facilities can exist on the computer until appropriate programs have been written. The writing of these programs, while conceptually straightforward, has in practice, required the use of advanced and, in many cases, recently developed programming techniques [2]. Moreover, it has frequently

been the case that these programs could only be constructed by small groups of dedicated programmers with access to large computing facilities.

Thus, it is naturally hoped that the computers of the future will be able to provide the facilities for algebraic manipulations as well as numerical applications through hardware support rather than software. At present, it appears that such a change will be slow. An obvious difficulty is related to representing algebraic expressions in the computer memory. It is clear that when algebraic expressions are written on paper they are not all of the same length, so when they are represented in the computer, they do not all occupy the same amount of memory space.

Most of the existing algebraic systems were developed during the 1960's. In 1960, LISP [9,13], SNOBOL4 [5], and ALPAK (algebra package) [3] were implemented. This was the first group of algebraic systems. A new version of ALPAK was developed in 1964, called ALPAK-B [3]. (The first version was referred to as ALPAK-A). In 1965, several other algebraic systems were available, including Korsvold's system and the GRAD ASSISTANT. The most productive period for the development of algebraic systems and packages was 1969-71. During this period, some significant developments in algebraic manipulation were achieved. Algebraic systems developed in this 3 year period included ALAM[2], REDUCE[2], CAMAL[2], SCRATCHPAD[2], ESP[2], FORMAC[2], SAC[2], MATHLAB[2], and ALTRAN[2,3,6]. The ALTRAN system and language were

designed and implemented by a group of people led by Hall, Brown and McIlroy. It was, in fact, an extension of the work done with ALPAK-A and ALPAK-B.

It is clear from the literature that applications have originated either around a particular system or within a particular problem area. In the first case, the existence of a general-purpose system has led to a proliferation of separate applications, while in the second case a particular problem area has led to a proliferation of algebra systems designed to deal with problems in that area. This type of growth is not surprising when one considers the wide variety of algebraic forms and operations which are possible. In order to develop a package with powerful features in one area, it may be necessary to limit its capabilities in others. For example, SNOBOL4 was designed to be used efficiently and extensively in examining strings for the occurrence of specified strings (i.e. pattern matching) and text preparation[5]. However, the SNOBOL4 package does not provide any algebraic facility for performing even the most simple algebraic operations. REDUCE, which is based on LISP, is well known to physicists who work in the field of quantum-electrodynamics, and is outstanding in the number of applications published in that field. In celestial mechanics and general relativity, REDUCE is known to be relatively slow[2]. CAMAL is the most powerful system in solving the problems of celestial mechanics, such as the problems of calculating the motion of the moon[2], but some of its computations are performed using a little known system ESP. In a

teaching situation, LISP is one of several languages which might be used if one is concerned with the CPU time and memory space consumed. Unfortunately there are no algebraic facilities built into the LISP package. Thus, the user must provide all the required algebraic facilities by writing the necessary routines in LISP. Over a broad range of applications, ALTRAN and FORMAC have been most successful. Surveys on the application of existing algebraic systems in a broad area of science by Barton and Fitch [2] reveal that ALTRAN and FORMAC are the most widely used general-purpose systems.

Both ALTRAN and FORMAC were designed to provide a comprehensive range of facilities, thus enabling them to be applied over a wide research area. However, there are several major differences between the two systems. These include their capabilities of handling differentiation and integration. In brief, any function expressed in terms of simple variables can be differentiated in ALTRAN by one of its package routines. For example, when the function

```
X/(X+1)
```

is differentiated with respect to X the result will be given as

```
1/(X+1)**2
```

In FORMAC, any function composed of elementary functions can be differentiated. Thus, if the function

```
COS(X**2)
```

is differentiated with respect to X by a FORMAC package routine, it will return

```
-2*X*SIN(X**2)
```

as the result. This same function cannot be differentiated by the

differentiation routine in the ALTRAN package.

For integration, the facility provided by the ALTRAN system is only used on polynomials with rational coefficients. For example, the function

```
(X-1)**2
```

when integrated, yields

```
(X-1)**3/3
```

However, an attempt to integrate the function

```
1/X
```

results in an error and termination of the ALTRAN program. No integration facility is provided in the FORMAC system.

In the preliminary design of SYMDIP, it was necessary to make several decisions which would affect later package capabilities. The first decision was what type of language would be used in writing the package. One approach would have been to use a language like LISP or SNOBOL4. Neither of these languages is specifically designed for algebraic manipulation, but both have the character manipulation capability necessary to build such facilities. Because of the limited time available, it was decided that a more powerful package could be produced if a language was used which already incorporated some basic algebraic facilities.

When considering the algebraic language to be chosen, the following points were considered to be important.

(a) The language should already be available on the university computer or its installation procedure should be relatively simple and not require a large amount of manpower.

(b) The package should provide as many algebraic facilities as possible.

(c) The package should not be prohibitively expensive in terms of CPU time and memory space.

Satisfying all these points is usually difficult in practice. Clearly (a) is more important than the other two. FORMAC does not satisfy either (a) or (b), since the implementation of that system on our installation has not yet been successful. Other algebraic packages such as FORMULA ALGOL, MATHLAB, ESP, SAC, and SCRATCHPAD were not chosen because they were not available on our installation or because they were incompatible with our system. LISP and SNOBOL4 do not provide even basic routines for doing algebraic calculations and thus were not considered. Consequently, ALTRAN became the most obvious language in which to write our package.

A copy of the ALTRAN algebraic package was acquired by the University of Victoria Computer Centre from Bell Laboratories. We implemented it in the summer of 1975 using the installation procedures that are provided in the associated manuals[7]. The ALTRAN system is currently available to any user of the University of Victoria Computer Centre.

1.3. ALTRAN.

ALTRAN is a complete system for symbolic computation with rational functions in several variables with integer coefficients. It has been designed to handle large problems with ease. As in most scientific programming languages, it provides the elementary operations of addition, subtraction, multiplication, division and exponentiation of numeric quantities. The same operations may also be performed on algebraic expresions. ALTRAN also allows the programmer to replace a variable by another variable, by an expression or by a numeric value. More complicated operations such as differentiation and integration are available through procedure calls to its library routines.

The ALTRAN system is composed of a translator, an interpreter and a run-time library. Most of these are written in FORTRAN IV. Some of the library routines handling specialized functions are written in ALTRAN. A small set of routines that are mainly used by the translator are written in ASSEMBLY language [3,6].

To use the ALTRAN system, a person writes an ALTRAN procedure specifying the algebraic manipulations to be performed.

The ALTRAN translator translates this source code into FORTRAN source code, which can then be compiled by a FORTRAN compiler. The structure of the FORTRAN source code which is produced by the ALTRAN translator is very simple. It consists of 6 arrays which are initialized by data statements. These are followed by calls to several subroutines. One of these calls is to the ALTRAN interpreter. The original source program is encoded by the translator into an

intermediate form in which variables and operations are replaced by code numbers. This text is placed in one of the six arrays for later "execution" when the interpreter is finally called with this array as one of its arguments. Variable names, their attributes (real, integer, etc.) and other miscellaneous information are stored in the other arrays for use by the interpreter during program execution.

If we call the rules which determine whether or not a statement is well formed syntax and call the rules which specify the meaning of a well formed statement semantics, both the syntax and semantics of the ALTRAN system have essentially been based on those of FORTRAN with ideas drawn from PL/I [6]. The number of data types has been increased so that it not only includes integer, logical and real, but also algebraic, rational and label.

A variable will acquire one of these data types only when it has been declared to be that type through a declaration statement. Thus, when a variable has been declared to have the algebraic type, the variable can be assigned a formal value of any algebraic expression in terms of those variables specified in the declaration statement. For example, when S is declared in the statement:

```
ALGEBRAIC (X:5, Y:6) S
```

S can then be any algebraic expression in terms of X and Y. The numbers 5 and 6 are the maximum exponents for X and Y, respectively, in such an expression. The variables X and Y are called indeterminates.

During the execution of any ALTRAN program, space for all data (including algebraic expressions) is allocated dynamically in a large array called the workspace. The size of this workspace is fixed at the time the ALTRAN system is implemented at an installation. The basic unit of allocation in the workspace is called a block and always contains data of one type. Thus, a block may contain either pointers to other blocks, integers or floating-point numbers. ALTRAN includes a procedure, called the "garbage collector", which is used when necessary to compact the blocks in the workspace that are not currently being used.

The value of each declared variable is represented by some structure in the workspace. A short integer (single precision) is stored as an integer. A long integer (double precision) is stored as an array of integers each representing a digit of the long integer in base 10 notation. Rational numbers (always in irreducible form) are stored as a pair of short or long integers. Short and long real numbers are represented, respectively, by single or double precision floating-point numbers.

A polynomial is represented by a block which contains three pointers. The first points to an array of its indeterminates, the second to an array of coefficients and the third points to an array of the powers of the indeterminates in the polynomial. For example, the polynomial

$$\text{poly} = 2x^3 - 4yz + x^2yz^3$$

will be represented in the workspace as follows:

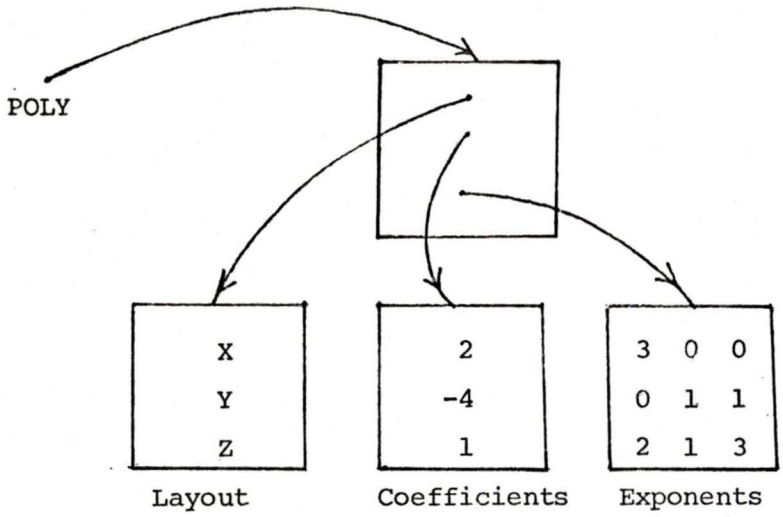

Figure 1.1. : The polynomial

$$POLY = 2x^3-4yz+x^2yz^3 \quad .$$

The representation of a general algebraic variable in the workspace is given as a pointer to a block, called a formal product, which contains pointers to each of the factors and one pointer to another block containing the degrees of the corresponding factors.

The ALTRAN system is able to manipulate vectors and arrays of rational functions as well as individual functions. For example, if A and B have been declared to be 2 x 2 algebraic arrays, then the statement

```
READ A,B
```

with input

```
X; (X**2,3,Y**2,2,7,X*Y)
```

yields

$$A = \begin{bmatrix} X & X \\ X & X \end{bmatrix}$$

and

$$B = \begin{bmatrix} X^{**}2 & 3 \\ 2 & 7 \end{bmatrix} \quad \begin{matrix} Y^{**}2 \\ X*Y \end{matrix}$$

In addition to occurring in input data, a parenthesized list can also appear explicitly in an ALTRAN procedure. Thus the assignment statements

```
A = X
```

and

```
B = (X**2,3,Y**2,2,7,X*Y)
```

have the same effect as the preceding read statement.

In ALTRAN, the elementary arithmetic operations may also be applied on vectors or arrays of the same dimension. For example, suppose C and D are the following 2 x 2 rational arrays:

$$C = \begin{bmatrix} 1/2 & 2 \\ 4 & -1/2 \end{bmatrix}$$

$$D = \begin{bmatrix} 1/2 & -2 \\ -3 & 3/2 \end{bmatrix} .$$

Let E be a 2 x 2 integer array. Then the statement

```
E = C + D
```

yields

$$E = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} .$$

One very important feature of ALTRAN data representation is that it permits sharing of data. For example, a block of coefficients may become a part of several polynomials, or a polynomial may become a factor in several formal products.

Besides the differentiation and integration facilities previously mentioned, other important algebraic facilities that are available in the ALTRAN package include the following :

(a) Routines for solving linear equations. For example, assume we are given a matrix A and a vector B, where

$$A = \begin{bmatrix} A11 & A12 & A13 \\ A21 & A22 & A23 \\ A31 & A32 & A33 \end{bmatrix},$$

$$B = \begin{bmatrix} B1 \\ B2 \\ B3 \end{bmatrix}.$$

Then a call to the ALTRAN procedure ASOLVE with A and B as arguments will use Cramer's Rule to solve for X in the equation $A \cdot X = B$. Specifically, if $X^T = [X1,X2,X3]$ a call to the linear equations solver yields

$$X1 = DET\begin{bmatrix} A12 & A13 & B1 \\ A22 & A23 & B2 \\ A32 & A33 & B3 \end{bmatrix} / \Delta,$$

$$X2 = -DET\begin{bmatrix} A11 & A13 & B1 \\ A21 & A23 & B2 \\ A31 & A33 & B3 \end{bmatrix} / \Delta,$$

and

$$X3 = DET\begin{bmatrix} A11 & A12 & B1 \\ A21 & A22 & B2 \\ A31 & A32 & B3 \end{bmatrix} / \Delta,$$

$$\text{where } \Delta = DET\begin{bmatrix} A11 & A12 & A13 \\ A21 & A22 & A23 \\ A31 & A32 & A33 \end{bmatrix}$$

and DET denotes the determinant.

(b) A routine for expanding algebraic expressions.

For example, given the expression

```
2*(X+3)**2
```

a call to the ALTRAN procedure EXPAND will produce

```
2*X**2+12*X+18 .
```

(c) Routines for computing the greatest common divisor of two integers or two algebraic expressions. For example, if we call the ALTRAN procedure AGCD with arguments

```
A = (X+3)**2*(X+1)
```

and

```
B = (X+1)**2*(X+3)
```

the value returned is

```
(X+1)*(X+3) .
```

The ALTRAN system has an efficient run-time error handling facility. Whenever a fatal run-time error occurs in the user's program the run is terminated immediately with a termination message, a symbolic snap and the run statistics. The termination message indicates the error number of the type of abnormal termination. The symbolic snap is a picture of the program at the time of failure, and indicates the procedure in which the fatal error occurred and the statement then being executed. The symbolic snap also provides the names of the procedures and the names and values of all variables used in each active procedure.

The output buffer management facility of the ALTRAN system prints the solution of an ALTRAN program in a readable form. The variable's symbolic name is always printed prior to its value. Each appears on a separate line. For example, if

$$y = (x^3+3x-2)/(2+x),$$

the statement

```
WRITE Y
```

would result in two lines of output:

```
# Y
(X**3+3*X-2)/(2+X).
```

Long algebraic expressions are printed on several lines. Each line, except for the last, terminates with an arithmetic operator.

Because of this generality, ALTRAN is a relatively large system. It includes over 300 routines. About 250 of these are used to handle such things as pointers, stacks and blocks when an ALTRAN program is being executed. The remaining routines are used to perform specialized functions such as differentiation and integration. If the workspace of the ALTRAN system has been fixed at 40K bytes, then execution of a simple ALTRAN program occupying 2K bytes would require approximately 290K bytes of memory space to hold the interpreter, workspace and other necessary routines.

The current version of ALTRAN supplied by Bell Laboratories can differentiate only functions of a simple variable with rational coefficients. There are two routines in the ALTRAN package which are written to handle differentiation, namely DIFF and DIFFN. Procedure DIFF returns the partial derivative

$$\frac{\partial f}{\partial x},$$

while procedure DIFFN returns the multiple partial derivative

$$\frac{\partial^n f}{\partial x_1 \partial x_2 \ldots, \partial x_1}$$

The integration operation of ALTRAN is more restrictive than differentiation in that only polynomials of simple variables can be integrated. There are also two routines which handle integration, namely PINT and PINTN. The procedure PINT returns the integral

$$\int_0^x p(t)dt = P(t)\Big|_0^x = P(x)$$

while the procedure PINTN returns the multiple integral

$$\int_0^{x_n} \ldots \int_0^{x_2} \int_0^{x_1} p(t_1,t_2,\ldots,t_n)\ dt_1,dt_2 \ldots dt_n$$

$$= P(t_1,t_2,\ldots,t_n) \Big|_{(0,0,\ldots,0)}^{(x_1,x_2,\ldots,x_n)} = P(x_1,x_2,\ldots,x_n).$$

Functions other than those described above cannot be differentiated or integrated without the writing of an ALTRAN program by the user. This is a functional limitation of the ALTRAN system.

There are other limitations, the most important being related to the form of algebraic expressions in an ALTRAN program. These can be summarized as follows:

(a) An algebraic expression cannot contain a real-valued quantity. For example, none of its coefficients can be real numbers.

(b) All the variables in an algebraic expression must be simple variables. For example, the following algebraic expression

```
4*SIN(X) - 3*(X)+5
```

will result in an error because SIN(X) is not a simple variable.

(c) All of the variables and indeterminates used in an ALTRAN program must be declared.

Conditions (a) - (c) have some obvious influences on our package design. The effect of (a) on the input function is clear. Conditions (b) and (c) imply that we need to replace all the elementary function names, the variable, and the alphabetic constants in the input function by some indeterminates which have been declared in our ALTRAN program. Consequently, a preprocessor and a postprocessor used as "translators" are necessary. The package therefore requires 3 major job steps.

1.4. The Problem.

We now list the set of problems we have solved in order to achieve our goals. The problems can be classified as follows:

(a) The package must be able to differentiate any function involving any elementary function introduced in a typical first year calculus course. Functions of functions up to a depth of 5 can be differentiated.

(b) The package must be able to integrate any function involving any elementary function introduced in a first year calculus class. Simple functions of

functions up to a depth of 3 can be integrated.

(c) The package should be able to run in less than 256K bytes of memory space without any significant decrease in functional capability.

(d) The CPU time required by the package to solve a given problem should be kept as low as possible.

(e) The JCL required of the user must be simple. In particular, the user should not be aware that there are 3 major job steps involved in the package.

(f) The error-handling facility of the package must be implemented with care and it must provide sufficient and meaningful error messages to the user.

(g) Since the package is designed for use as a teaching tool, students must be able to provide their problems in a simple and natural form. The result must also be presented in a form which is easy to understand.

CHAPTER TWO

PACKAGE CAPABILITIES AND LIMITATIONS

2.1 Introduction

2.2 Differentiation

2.3 Integration

2.4 Error Handling Within The Package

2.5 Package Usage Suggestions

2.1. Introduction.

This chapter describes the functional capabilities and limitations of our package. In this section we specify the general types of input formats which are acceptable to the package. Examples showing the output of the package are also given. Finally, several definitions which are needed in later sections are presented. In Sections 2 and 3, we specify in detail the set of functions our package is able to differentiate and integrate. Section 4 discusses the error-handling facilities which are provided by our package, with the emphasis on the various types of errors which the package is able to recognize, as well as actions taken by the package when any one of these errors occurs. Finally, in the last section of this chapter, we give a variety of suggestions on how to formulate functions in order to reduce the computer time required by the package. In this same section we indicate several techniques which may transform a problem which is not solvable by the package into a problem which is solvable.

In order that a problem can be solved by SYMDIP, the problem must first be stated in a correct form. The user of SYMDIP is allowed to state his problem in either of the following forms:

(a) DIFFERENTIATE "function" (WITH RESPECT TO "variable")

(b) INTEGRATE "function" (WITH RESPECT TO "variable")

The words in quotes have the following meaning. "Function" denotes a function of one variable which is represented using the usual syntax rules of FORTRAN. For example, the function

$$2 \sin^3(x+3)$$

would be represented as

```
2 * SIN(X+3)**3
```

"Variable" denotes any single letter of our usual 26 letter alphabet. As the form of the problem implies, the symbolic differentiation or integration is done with respect to this variable.

In statements (a) and (b), the expressions enclosed in brackets are optional. If they are not included as part of the problem, the variable will be determined by SYMDIP according to these rules:

(1) Set "variable" equal to the first single alphabetic letter encountered in "function".

(2) If (1) fails, set "variable" equal to 'X'.

Thus, given the problem

```
DIFFERENTIATE  2 * SIN(Y+3) - A*X*Y ,
```

the "variable" will be determined by SYMDIP. By applying rule (1), the "variable" will be set to 'Y'.

As a second example, consider the problem

```
INTEGRATE  A * LN(Y+3) WITH RESPECT TO Y .
```

Because the variable of integration is already specified as 'Y', it will not be changed although 'A' is the first single letter in "function".

If the 'WITH RESPECT TO "variable" ' part of a problem is not correctly stated, the package disregards this part of the problem statement and the previous rules are applied. For example, in the following problem

```
DIFFERENTIATE  2 * Z * SQRT(Z+A) WITH RESPECT A
```

the "variable" will be set equal to 'Z' by rule (1) although 'A' was apparently desired by the user.

For convenience, SYMDIP allows the user to use 'DIFF' and 'INT' in place of 'DIFFERENTIATE' and 'INTEGRATE', respectively. Likewise, 'W.R.T.' may be used in place of 'WITH RESPECT TO'.

Thus, all of the following are acceptable and each would produce the same result (except for the obvious changes in variables):

```
DIFFERENTIATE 2 * T*SQRT(T + A) WITH RESPECT TO  T
DIFF  2 * X * SQRT(X+A)
DIFF 2 * Y*SQRT(Y+A) W.R.T. Y
```

Problems to be solved by SYMDIP are punched on cards. A problem can be stated using all 80 columns of a card. If a problem does not fit on a single card, additional cards may be used. Blanks within the "function" are allowed. "Function" must be separated from the first part (i.e. 'DIFF' or 'INT') and the last part (i.e. 'WITH RESPECT TO "variable"') of a problem by at least one blank. At least one blank must also separate individual words and the "variable" in 'WITH RESPECT TO "variable"'.

Having discussed the general form of a problem, we are now ready to discuss the "function" portion in more detail. The following definitions will be needed.

Definition 2.1: A function name is any one of the following:

SIN, COS, CSC, SEC, TAN, COT, SINH, COSH, CSCH, SECH, TANH, COTH, ASIN (ASIN denotes the inverse of SIN), ACOS, ACSC, ASEC, ATAN, ACOT, ASINH, ACOSH, ACSCH, ASECH, ATANH, ACOTH, LN (natural logarithm), LOG (common logarithm), EXP (exponentiation), SQRT (square root).

Definition 2.2: A function name is nested provided it appears as part of the argument of another function name. A function name is properly nested in another provided they are related as follows:

function name (function name(argument))

As we have previously indicated, the "function" in a problem must be written in a form which is consistent with the FORTRAN syntax. In addition to this, there are several conditions which the "function" must obey, including the following.

(a) The maximum exponent associated with a function name, a variable, or an alphabetic constant is 8.

(b) The maximum number of function names is 12.

(c) The maximum number of distinct alphabetic constants is 5.

(d) Every alphabetic constant is a single letter.

(e) Only function names defined in Definition 2.1 are used in the "function".

In the rest of this chapter, these conditions will be referred to as "Conditions (a) - (e)". We note that all of these limits were imposed for programming purposes and none represent limitations imposed by the ALTRAN system. Several other rules, which must also be satisfied by "function" for only differentiation or only integration problems, are listed in those sections where differentiation and integration are specifically discussed.

We conclude this section by describing the form of solution produced by SYMDIP. The output provided to the user consists of the problem statement followed immediately by the computed results. Although a problem provided by the user may not be in a complete form, for example, the 'WITH RESPECT TO "variable"' part of a problem may not be stated, SYMDIP will always print the problem in a complete form. The following examples show typical solutions produced by SYMDIP.

Input:

```
DIFF X-SIN(X)
```

Output:

```
PROBLEM 1

DIFFERENTIATE X-SIN(X)

WITH RESPECT TO X

THE RESULT IS :

1 - COS(X)
```

Input:

```
INT X * LN(X)
```

Output:

```
PROBLEM 2

INTEGRATE X * LN(X)

WITH RESPECT TO X
```

THE RESULT IS :

X ** 2 * (LN(X)+ 2) / 4 + CONSTANT

2.2. Differentiation .

The rules which determine if a function can be differentiated by our package are easy to state. A function is differentiable by the package provided it satisfies Conditions (a) - (e), stated in the last section, and the following condition:

(f) The maximum depth of nesting is 4.

Thus, the function

3 * TON(X)

is not differentiable by the package because the function contains an undefined function name 'TON', and hence violates Condition (e). Similarly, the function

SIN(X+A+B) * C * D * E * F

violates Condition (c) of Section 2.1, because there are more than 5 alphabetic constants. Thus, it could not be differentiated by the package. However, the function

SIN(COS(X-A)) + B - X ** 6

satisfies Conditions (a) - (e) and Condition (f), and therefore it can be differentiated with respect to A or B or X.

2.3. Integration.

The set of functions which are integrable by SYMDIP are classified into 15 forms. Although we shall introduce these functions by presenting their general forms, we shall at the same time give several simple examples, which we feel will help the reader to understand the meaning of the form.

To simplify our task, we indicate the notation which will be used in describing the functions which SYMDIP can integrate. We assume the reader is familiar with the meaning of the terms rational, coefficient, polynomial and monomial.

Symbol	Meaning
F1	A polynomial in a simple variable with rational coefficients.
F2	A monomial in a simple variable with rational coefficients.
F3	A polynomial in a simple variable with rational coefficients and the degree of the variable is less than 3.
F4	F1/F2 or F1/F3.
U,U1,U2,U3,U4	Functions in a simple variable with rational coefficients.
L1	exp, inverse trigonometric functions or inverse hyperbolic functions.
L2	Trigonometric functions or hyperbolic functions.
L3	sin, cos, sinh or cosh.
G	L2, exp, ln, sqrt or the identity function.
A,B	Rationals or alphabetic constants.
A1,B1	Any expression which is independent of the variable of integration.
Z	A x + B.
I	A positive integer.
I1, I2	Integers.
M,N	Arithmetic expressions which evaluate to a positive integer value.
P,Q,J,K,L	Arithmetic expressions which evaluate to an integer value.
W(x,y)	A function in variables x and y with rational coefficients.

H(x) A function of x.

Finally, if the functions G or U occur more than once in an expression describing a particular integration form, each occurrence denotes the same function.

Using the above notation, we are ready to describe the first kind of function which can be integrated by SYMDIP. A function is integrable if it consists of a single term satisfying Conditions (a) - (e) and has a form which matches at least one of the 15 forms given below.

Because it may not be immediately obvious, we note that a function must satisfy the following conditions if it is to match one of these 15 forms.

(I) Only proper nesting (Definition 2.2) of elementary functions is permitted. In addition, inverse trigonometric, inverse hyperbolic and the log functions cannot be nested.

(II) The maximum depth of nesting is 1.

The 15 acceptable integration forms are:

Form 1 $\int F4\ dx$

Example:

(I) $\int (ax^5 - x + 3)/(x^2 - 7x + 5)\ dx$

Form 2A $\int G'(U)\ G^M(U)\ L1(G(U))\ dx$

Examples:

(I) $\int x^7 \exp(x)\ dx$

(II) $\int 2xy \cos(x^2) \sin(x^2) \operatorname{asinh}(\sin(x^2))\ dx$

Form 2B $\int G'(U)\ G^M(U)\ L3^N(G(U))\ dx$

$(N \leq 3)$

Examples:

(I) $\int x^3 \sinh^3(x)\, dx$

(II) $\int \sin(x) \cos^2(x) \cosh(\cos(x))\, dx$

Form 2C $\int G'(U)\, G^J(U)\, \ln^K(G(U))\, dx$

(If $J \neq -1$, $K>0$)

Examples:

(I) $\int (1+x)^5 \ln(1+x)\, dx$

(II) $\int -(1/x^2) \ln^3(1/x)\, dx$

Form 3 $\int G'(U)\, L2^J(G(U))\, L2^K(G(U))\, dx$

(here L2's are functions of the same type, i.e. they are both trigonometric or hyperbolic functions)

Examples:

(I) $\int \sin(x) \cos^3(x)\, dx$

(II) $\int \cos(x) \tan^2(\sin(x)) \cos(\sin(x))\, dx$

Form 4 $\int U1(x)\, L4(U2(x))\, L4(U3(x))\, dx$

(L4's are functions of the same type and $U2 \neq U3$. $U1(x)$ must satisfy the following 2 conditions:

(I) $U1(x) = (U2'(x) + U3'(x))\, (U2(x) + U3(x))^M\, A1$

(II) $U1(x) = (U2'(x) - U3'(x))\, (U2(x) - U3(x))^N\, B1$

for suitable choice at M, N, A1, and B1).

Examples:

(I) $\int x \sin(x) \cos(2x)\, dx$

(II) $\int x^3 \sinh(x+x^2) \cosh(x-x^2)\, dx$

Form 5 $\int G'(U)\ \text{sqrt}^J(G(U))\ dx$

Examples:

(I) $\int 1/\text{sqrt}(x)\ dx$

(II) $\int 1/(1-x)^2\ \text{sqrt}^3(1/(1-x))\ dx$

(III) $\int \sin(x)\ \text{sqrt}(\cos(x))\ dx$

Form 6A $\int U1(x)\ L4(U2(x))\ L4(U3(x))\ \exp(U4(x))\ dx$

(L4's are functions of the same type and $U2 \neq U3$

U1(x) must satisfy the following conditions:

(I) $U1(x) = (U2'(x) + U3'(x))\ A1$

(II) $U1(x) = (U2'(x) - U3(x))\ B1$

(III) U1(x)/U4'(x) is independent of x for suitable choice of A1 and B1.

Examples:

(I) $\int \sin(x)\ \cos(x)\ \exp(x)\ dx$

(II) $\int \sinh(ax)\ \cosh(x)\ \exp(bx)\ dx$

Form 6B $\int U'(x)\ L3^M(U(x))\exp(A1\ U(x))\ dx$

Example:

$\int (4x-5)\ \sinh^3(2x^2-5x)\ \exp(4x^2-10x)\ dx$

Form 6C $\int G'(U)\ L3^M(G(U))\exp(G(U))\ dx$

Example:

$\int (2x+6)\ \cos(x^2+6x)\ \cos^7(\sin(x^2+6x))$
$\exp(\sin(x^2+6x))\ dx$

Form 7A $\int G'(U)\ G^M(U)\ \text{sqrt}^J(G(U)^2 \pm I^2)\ dx$

Form 7B $\int G'(U)\ G^M(U)\ \text{sqrt}^J(I^2 \pm G(U)^2)\ dx$

(where the expression G(U) should not contain '**2')

Examples:

(I) $\int (x-1)^3 \, \text{sqrt}((x-1)^2 - 4^2) \, dx$

(II) $\int x/\text{sqrt}^2(3^2 + x^2) \, dx$

(III) $\int \frac{(2x+1)}{(x^2+x-2)} \ln^3(x^2+x-2) \, \text{sqrt}^3(\ln(x^2+x-2)^2 + 5^2) \, dx$

Form 8A $\int G'(U) \, G^M(U) \, \ln(G(U) + \text{sqrt}(G(U)^2 \pm I^2)) \, dx$

Form 8B $\int G'(U) \, G^M(U) \, \ln(G(U) - \text{sqrt}(G(U)^2 \pm I^2)) \, dx$

(where the expression G(U) should not contain '**2').

Examples:

(I) $\int x \ln(x+\text{sqrt}(5^2 + x^2)) \, dx$

(II) $\int 2x \, (x^2-3)^3 \ln(x-\text{sqrt}((x^2-3)^2 - 3^2)) \, dx$

(III) $\int \sin(x+5) \ln(\cos(x+5) - \text{sqrt}(\cos(x+5)^2 + 3^2)) \, dx$

Form 9A $\int x^P \text{sqrt}^Q(J x^2 + K x) \, dx$

($J, K \neq 0, Q = 1$ or -1)

Form 9B $\int x^P \text{sqrt}^Q(Jx^2 + Ky + L) \, dx$

($1 \neq 0$. $P = 1$ or 0; when $P = 1$, $Q = -1$; when $P = 0$, $Q = 1$ or -1)

Form 9C $\int 1/(x^2 \, \text{sqrt}(Jx + K)) \, dx$

Examples:

(I) $\int x^5/\text{sqrt}(2x^2 + 5x) \, dx$

(II) $\int \text{sqrt}(x^2-x-7) \, dx$

(III) $\int 1/(x^2 \text{sqrt}(4x + 3)) \, dx$

For functions of Form 10-14 below, a change of variable will occur which transforms the form of integral given into Form 1. A function of Form 10-14 is integrable only if the new form is integrable.

Form 10 $\int W(\text{sqrt}(G(Z)),G(Z))\ G'(Z)\ dx$

Examples:

(I) $\int (1 - \text{sqrt}(x)) / (1 + 2\ \text{sqrt}(x))\ dx$

(II) $\int \text{sqrt}(x-1) / (2 + x + \text{sqrt}(x-1))\ dx$

Form 11 $\int H(\text{sqrt}(G(U)))\ G'(U)\ dx$

Examples:

(I) $\int (7 + \text{sqrt}(x^3))\ x^2 / (\text{sqrt}(x^3))\ dx$

(II) $\int (1-\text{sqrt}(\sin(x)))\ \cos(x)/(a+3)\ dx$

Form 12 $\int W(\exp(U),\exp(-U))\ U'(x)\ dx$

Examples:

(I) $\int (3 + \exp(x-1)) / (1 - \exp(x-1)\ dx$

(II) $\int (\exp(x) + a) / (b - \exp(-x))\ dx$

Form 13 $\int H(\exp(G'(U)))\ G'(U)\ dx$

Examples:

(I) $\int (4 - \exp(x^2)) / (1 + \exp(x^2))\ x\ dx$

(II) $\int \sin(x)\ \exp(\cos(x)) / (c - \exp(\cos(x)))\ dx$

Form 14 $\int W(L3(G(U)),L3(G(U)))\ G'(U)\ dx$

(L3's are functions of the same type)

Examples:

(I) $\int (1 - \cos(x)) / (1 + \cos(x))\ dx$

(II) $\int (9 - \cosh(x)) / (2 - \sinh^2(x))\ dx$

(III) $\int (a + \cos(\sin(x))) / (1 + \cos(\sin(x)))\ \cos(x)\ dx$

Before describing functions of Form 15, it is necessary to give a brief explanation of how functions of Form 1-14 are integrated. A detailed description is found in Section 3.4.

If a function matches one of Forms1-14, integration is performed by using a general integration formula. This usually requires proceeding through several integration steps in order to produce the solution. For example, the first step in evaluating a Form 2C function is to perform the following transformation:

$$\int g'(u)\, g^j(u)\, \ln^k(g(u))\, dx$$
$$= \int g^j(u)\, \ln^k(g(u))\, d(g(u))$$
$$= \frac{g^{j+1}(u)}{j+1} \ln^k(g(u)) - \frac{k}{(j+1)} \int g^j(u)\, \ln^{k-1}(g(u))\, d(g(u))$$

The integral on the right may be integrated again by using the same method. Other function forms will also be transformed but not necessarily in exactly this way. With this brief description of the way that functions of Forms1-14 are handled, we can give a definition of Form 15 functions.

Form 15. Special functions

These functions can all be integrated in a single step. Thus the form of solution can be placed in a table and it is only necessary to substitute a function and obvious constants to obtain the result. We give 3 examples of special functions and provide the complete list in Appendix F.

Examples:

(I) $\int g'(u)\, \sin(g(u)) \,/\, (1 + \cos(g(u)))\, dx$

(II) $\int g'(u)\, \cos(g(u))\ \ \text{sqrt}\,(1 - I^2 \sin^2(g(u)))\, dx$

(III) $\int \frac{g'(u)}{(m \sin(g(u)) + n \cos(g(u)))}\, dx$

Two comments are needed to complete our presentation of all the 15 forms of functions.

(i) If the argument of a function is a positive integer, the function is treated as a constant. It will not be evaluated to a real-valued quantity.

(ii) The expression 'number ** f(x)' is always transformed to the equivalent form 'EXP(f(x) * LN(number))'.

A function which does not exactly match any of the 15 forms but is a constant multiple of a form which can be integrated by SYMDIP, is also integrable. For example, the function x^7 exp(x) (a+5) does not match any of the forms given above, but it is a constant multiple of x^7 exp(x), which is of Form 2A. Therefore x^7 exp(x) (a+5) is integrable.

In addition to integrating functions having only a single term, SYMDIP is able to integrate functions with more than one term provided they are properly formulated. Specifically, if a function consists of the sum or difference of two or more terms of the form given above, then SYMDIP will integrate the function term by term. Because each integral is done independently of the others, no cancellations between terms will be performed. If a particular term does not belong to any of the 15 forms mentioned previously, the term will not be integrated. The final integral of the function is simply given as the integrals of those integrable terms plus the unintegrable parts.

Although SYMDIP is able to recognize each factor of a function, it will never expand the function. Thus, a function must be written

in expanded form if the user wishes the function to be integrated by considering individual terms of the expansion. To illustrate what we mean, consider the following problem :

INT -SIN(X) + COS(C+X) + 2*A W.R.T. X

Since the function contains 3 terms, SYMDIP would integrate these terms (with respect to X) separately as if there were 3 separate problems. The output produced would be as follows :

```
PROBLEM 1
INTEGRATE -SIN(X) + COS(C+X) + 2*A
WITH RESPECT TO X
THE FUNCTION YOU HAVE PROVIDED HAS BEEN SEPARATED INTO
3 TERMS WHICH HAVE BEEN INTEGRATED SEPARATELY
TERM 1
-SIN(X)
THE INTEGRAL IS :
COS(X) + CONSTANT
TERM 2
COS(C+X)
THE INTEGRAL IS :
SIN(C+X) + CONSTANT
TERM 3
2*A
THE INTEGRAL IS :
2 * A * X + CONSTANT
```

As a second example, consider the problem

INT (X + SIN(X)) * X .

Because SYMDIP does not expand the function, it would be treated as a single term. This function is not integrable by SYMDIP, because it does not match any of the 15 forms of integrable functions given above.

It should be noted that different forms of a function may sometimes lead to different results. For example, if the function (x+sin(x))x were written in the form X ** 2 + SIN(X) * X it would become integrable by SYMDIP because each of the two terms is integrable.

Clearly the functional capabilities of an integration package are expanded by allowing a function to contain several terms. Care must be taken, however, not to integrate term by term if this is not necessary. Consider the polynomial:

$$2x - 3 + x^2 .$$

If it were to be integrated term by term, considerable computer time would be wasted in solving 3 separate problems when one call to the ALTRAN integration routine would do it all. To avoid this situation, terms which do not involve any function names are collected into a single term by SYMDIP. This term is always the last term to be integrated. For example, if the function is given as $x + \sin(x) - x^2$, it will be considered by SYMDIP as having only 2 terms, namely $\sin(x)$ and $x - x^2$, while conventionally the function in fact contains 3 terms. Likewise, the polynomial we have just mentioned is treated as only one term. We leave this section by giving the solutions for the problems

```
INT X**2 + X*SIN(X) + 3
```

and

```
INT X*(X+SIN(X)) + 3
```

Output :

```
PROBLEM 1
INTEGRATE X**2 + X*SIN(X) + 3
WITH RESPECT TO X
THE FUNCTION YOU HAVE PROVIDED HAS BEEN SEPARATED INTO
2 TERMS WHICH HAVE BEEN INTEGRATED SEPARATELY
TERM 1
X*SIN(X)
THE INTEGRAL IS :
SIN(X) - COS(X) * X + CONSTANT
TERM 2
X**2 + 3
THE INTEGRAL IS :
X * (X ** 2 + 9) / 3 + CONSTANT

PROBLEM 2
INTEGRATE X*(X+SIN(X)) + 3
WITH RESPECT TO X
THE FUNCTION YOU HAVE PROVIDED HAS BEEN SEPARATED INTO
2 TERMS WHICH HAVE BEEN INTEGRATED SEPARATELY
TERM 1
X*(X+SIN(X))
*** RUN-TIME ERROR *** INTEGRATION ON THE FUNCTION TERMINATED ***
*** THE INTEGRAND IS BEYOND THE PACKAGE CAPABILITIES ***
```

```
TERM 2

3

THE INTEGRAL IS :

3 * X + CONSTANT
```

2.4. Error Handling Within The Package .

Because the package we have developed is to be used by individuals who know little or nothing about computers and computing, more than the usual effort has been put into providing meaningful error messages and error recovery procedures. We distinguish between three types of errors, namely, input errors, run time errors, and system errors.

An input error will occur when a particular problem is not well formulated. This would occur, for example, when the problem contains an undefined function name. Thus, an input error could only occur in the preprocessor step, that is, the first job step of the package. Whenever an input error is discovered, the problem being processed is terminated and an appropriate error flag is set. After that, the preprocessor proceeds to the next problem if there is one.

In many instances, a well formulated problem may not be solvable by the program. This would occur, for example, when the function is written in a correct form but does not belong to any of the 15 forms of functions stated in the last section. When a situation such as this occurs, a run time error is said to have occurred. An appropriate error flag is set and the problem is immediately terminated. Computation on the next problem to be solved is then begun. If no problems remain, then the postprocessor is entered to print out all problem results and error messages.

The third kind of error is the system error. This type of error

may occasionally occur when computations on a problem are being performed, in other words, in the solution step of a problem. An obvious example is related to the size of the workspace in the ALTRAN program. ALTRAN performs all algebraic operations in a workspace of fixed size. If the derivative or integral being computed contains too many terms, they may not all fit into the workspace provided. In such a situation the computations clearly should be terminated and a system error is said to have occurred. Consequently, the ALTRAN job step is terminated immediately after printing the "snaps" on a temporary file. Execution of the subsequent step, that is, the postprocessor, is then started. The solutions obtained before the system error occurred are printed in the usual way, after that the file containing the "snaps" is read and the system error number is determined. This number is converted into a meaningful error message which is then printed on the user's output. There are several other situations that would also cause a system error. For example, a system error would occur if the program exhausted its allocated CPU time while attempting a problem or if the coefficient of an expression became prohibitively large.

Having discussed the various types of errors, we now illustrate how these error messages will appear in the output.

If a run is successful, the following statement will be printed immediately after the set of solutions:

***** ALL PROBLEMS ATTEMPTED *****

However, if a system error has occurred during the run, the following statement will be printed:

*** THE RUN WAS ABNORMALLY TERMINATED *** ,

and it will then be followed by the system error message which is obtained as described above. Input and run-time error messages are both determined in the postprocessor step by using the error flags set in the previous steps. Each of these messages is printed immediately following the listing of the corresponding problem. The message is preceded and followed by a string of asterisks. A typical example of an input error message is as follows:

```
PROBLEM 5
INTEGRATE TON(X)
WITH RESPECT TO X
*** INPUT ERROR *** ATTEMPT AT PROBLEM SOLUTION CANCELLED ***
*** A FUNCTION NAME IN THE FUNCTION IS NOT RECOGNIZABLE ***
*** THE FUNCTION NAME IS TON ***
```

The following is a run-time error message:

```
PROBLEM 3
INTEGRATE X * ATAN (X) ** 2
WITH RESPECT TO X
*** RUN-TIME ERROR *** INTEGRATION ON THE FUNCTION TERMINATED ***
*** THE INTEGRAND IS BEYOND THE PACKAGE CAPABILITIES ***
*** THE POWER OF THE INVERSE FUNCTION IS OUTSIDE THE PACKAGE
    RANGE ***
```

2.5 Package Usage Suggestions

Because of various package design features, certain function forms are handled more efficiently than others. Formulating your differentiation

and integration problems using the suggestions given below will usually result in reduced computer time being required for their solution. Suggestions (e) and (f) may transform a problem which is not solvable by the package into one which is solvable.

(a) If possible, you should not use expanded forms. For example, (X-1)**3 is much better than X**3 - 3*X**2 + 3*X-1 although they are equivalent expressions.

(b) When it is obvious, you should simplify your function by cancellation of terms. For example, 2*X-4 is better than X+(3+X)-7 and (X-4)+(X+3)-3.

(c) When a function name has a power other than 1 or -1, that function name should be enclosed in brackets. For example, (SIN(X)) ** 3 is better than SIN(X) ** 3.

(d) For integration, attempt to use the four basic trigonometric and hyperbolic functions SIN, COS, SINH, AND COSH rather than CSC, CSCH, SEC, CSCH, TAN, TANH, etc. You can easily do this by expressing the latter functions in terms of basic functions. For example, SIN(X)/COS(X) is better than TAN(X) or SEC(X)/CSC(X).

(e) In an integration problem, if the function provided belongs to any of the Forms 7A, 7B, 8A or 8B, then in the argument of 'SQRT', neither I nor G(U) can be stated to contain '**2'. For example, the following function

```
SQRT(5**2+(X**2+3)**2)
```

is not allowed but the equivalent expression

SQRT(5**2 + (X*X+3)**2) is acceptable.

(f) If you have a function which cannot be integrated by the package, expanding the function may help. For example, (X+3) * ATAN(X) cannot be integrated by SYMDIP, but X * ATAN(X) + 3 * ATAN(X) can.

CHAPTER THREE

PACKAGE DESIGN

3.1 Introduction

3.2 The Analyzer

3.3 The Processor - Differentiation

3.4 The Processor - Integration

3.5 The Writer

3.6 Interfaces Between Job Steps

3.1. Introduction.

This chapter describes the internal organization of our package. In this section we present an overview of the structure of the package and the strategy used in attempting to solve a particular problem. In Sections 2,3,4 and 5 we describe the internal structure of the various components of the package, namely the analyzer, the two parts of the processor and the writer. Finally, in the last section, we illustrate how these separate portions of the package interface with one another. Several completed examples of the data passed between different components of the package are given.

One of the design criteria given in Chapter 1 was that the user be able to pose his problem in a simple and reasonably natural form. As Chapter 2 illustrates, we have been sucessful in doing this. Because of the input/output limitations of ALTRAN (see Section 1.4) and our intention of significantly enlarging the domain of functions (as compared with that of the basic ALTRAN program) which can be differentiated or integrated, it was decided that the package design should include the following components:

(I) The analyzer. A preprocessor which translates the user's problem into an ALTRAN comprehensible form which is acceptable to the differentiation/integration processor. If an integration problem is being processed, the analyzer will determine if the function is a special function (Form 15) as described in Chapter 2.

The type of special function will also be decided by the analyzer.

(II) The differentiation/integration processor. It manipulates the set of ALTRAN comprehensible information using an appropriate strategy in an attempt to solve the problem, and produces intermediate ALTRAN comprehensible output of the solution.

(III) The writer. A postprocessor which translates the results computed by the differentiation/integration processor and prints them. The writer also recognizes error flags and prints the appropriate error messages in output.

The analyzer and the writer were written in SNOBOL4. The processor, which is the main body of the package, was written in ALTRAN. Since most of the computation is performed by the processor, the major portion of execution time of the package is spent in the processor step.

Having described the components of our package, we now discuss the overall strategy of the processor in attempting problems. The derivative of a function is relatively easy to obtain. Given any function composed of elementary differentiable functions, its derivative can always be computed using the rules of differentiation of sums, differences, products, and quotients of 2 functions and the chain rule for differentiation of functions of functions. In particular, we make extensive use of DIFF which is a partial differentiation procedure of ALTRAN which computes the derivative of any rational function of a simple variable. In the case of

integration, the situation is much more complicated. Many integrable functions require particular techniques to solve them. The most common example is the method of "integration by parts". Because of the complicated situation in integration, our package was designed to meet, among others, the following goals:

(a) Find the solution quickly.

(b) Find a comprehensible solution.

The first goal implies that the program must apply an efficient strategy in attempting problems. As a problem solving program, the program must be able to solve a given problem with a method which is as easy and simple as possible.

The second goal is less familiar to numerical programmers. Given an integrable function, it can usually be integrated in a variety of ways. The solutions produced by the various approaches are frequently different in form, although they are mathematically equivalent to one another. For example,

$$\begin{aligned} & \int \sin(x)\cos(x)\,dx \\ &= \frac{1}{2}\sin^2(x) + C_1 \\ &= \frac{1}{2}\cos^2(x) + C_2 \\ &= -\cos(2x)/4 + 6 + 1/\csc^2(x) + \cos^2(x) + C_3 \end{aligned}$$

where C_1,C_2, and C_3 are arbitrary constants.

Clearly, some of these solutions are more comprehensible than others. More precisely, if a student poses an integration problem in terms of sines and cosines, then he would probably prefer to see the answer in those terms as well. In order to retain similar function forms, a radical

transformation of the integrand should not be attempted unless there is no simple way to produce the integral. The use of an integration table, for example, is an approach which can be used to eliminate transformations of the integrand. The overall strategy of our integration program is as follows:

Stage 1. Simplify the problem by an obvious change of variables.

Stage 2. If the function is a special function, the coefficients of the eventual integral are computed in this stage. The actual form of the integral is given by an integration table (in the writer).

Stage 3. If the function is not a special function, attempt to integrate it by calling appropriate processor routines.

Stage 1 is passed very quickly; the only work done in this stage is to replace some complicated arguments by simple variables.

Stage 2 is fast as well. If the function is a special function, the type number of the special function is determined by the analyzer. Since the form of the integral of a special function is partly provided in our integration table, only coefficients of the eventual integral are computed in this stage. To illustrate the last point, consider the function

```
A/(1 - SIN(X))**2
```

which is one of the special functions listed in Appendix F. The exact integral of the function is B*COT(π/4 - X/2) + C*COT(π/4 - X/2)**3, where B = A/2, C = A/6. The integration table contains COT(π/4 - X/2) and COT(π/4 - X/2)**3 in an appropriate entry. Thus, if we wish to integrate

10/(1 - SIN(X))**2, stage 2 must compute the coefficients B and C. Consequently, B = 5 and C = 5/3.

Stage 3 deals with all kinds of functions other than the special functions (Form 15). A function is integrated in this stage using appropriate routines determined by the form of the function.

If a function cannot be integrated in either stage 2 or stage 3, then it is beyond the package capabilities. In this case an appropriate error flag is set.

As an example, consider the following problem :

INTEGRATE 2*X*SIN(X**2)*COS(X**2)*LN(COS(X**2)) W.R.T. X

The argument of LN, namely COS(X**2) is replaced by Y in stage 1. Since

COS(X**2)dX = -2*X*SIN(X**2),

we have

2*X*SIN(X**2)dX = -dCOS(X**2) = -dY.

At the end of stage 1, the problem becomes:

INTEGRATE -Y*LN(Y) W.R.T. Y.

Stage 2 is not entered because the analyzer should have indicated that the original function is not a special function. Stage 3 is entered and the integral is obtained because its form matches integration Form 2C of Section 2.3. In this case the result is:

- Y ** 2 * (2*LN(Y) - 1) / 4 + CONSTANT

which becomes

```
- COS(X**2) ** 2 * (2*LN(COS(X**2) - 1) / 4 + CONSTANT
```

when Y is replaced by COS(X**2).

Note that the integrals computed in stage 3 are given in ALTRAN comprehensible forms. The final user comprehensible form of these integrals is produced by the writer.

The remaining sections of this chapter will give a more detailed description of the various components of the package. Before proceeding to that material we define several terms which are used in the rest of this chapter.

Definition 3.1

Array	A one-dimension variable with more than one entry.
Stack	A list of items to which items are always added and deleted from the same end. Additions or deletions are done one item at a time.
Character string	Any finite sequence of characters. A character string may contain letters, digits, left and right parentheses, blanks, operators and so forth.

3.2. The Analyzer.

The first program to execute when SYMDIP is called is the analyzer. It reads all problems which are supplied by the user, performs the syntactic

analysis on these problems, and finally translates the well formulated problems into an internal representation which can be manipulated by the ALTRAN processor. A problem which is not well formulated is replaced by an appropriate error flag.

The syntactic analysis part of the analyzer determines if a problem is well formulated. The syntax of the function itself is checked. If the function contains certain special constructions (see Section 2.3) this will be discovered by the analyzer. Part of the analyzer's activity is related to "complexity analysis" which determines the basic form of a function based on its elementary function composition. This is a very important step of the analyzer. A special function, for example, is determined in the complexity analysis step of the analyzer. If the form of a function cannot be determined by the analyzer, the problem will be rejected.

In order to describe the analyzer, 3 logical parts can be distinguished. These are (in order of execution):

Part 1. Which performs syntactic analysis.

Part 2. Which transforms a single complex function into a set of functions of a simple variable. In general it replaces the top level function names in "function" with simple variables, Z(I)'s, where I's are consecutive positive integers starting from 1. If the function is to be integrated, the basic form of the function is also determined in this part. Because differentiation is handled in rather a different way from integration in part 2 of the analyzer, the detailed description will

be given in 2 parts.

Part 3. Which replaces the variable and all the alphabetic constants with simple variables. In brief, the variable is replaced by Z0 and alphabetic constants are replaced by Z1(I)'s throughout the function, where I's are also consecutive positive integers starting from 1.

We now discuss the detail of each of these parts.

Part 1. Perform syntactic analysis. Included in this part of the analysis are:

(a) Are the left and right parentheses matched in the function?

(b) Are the operators placed correctly throughout the function?

(c) Is the "function" nonvoid?

(d) Are all the characters used in the function allowable?

(e) Is the WITH RESPECT TO "variable" part of the problem correctly stated by the user?

(f) Are all complete substrings of the character string "function", which consists of two or more letters, valid function names? Are they followed by '('?

(g) Are digits separated from letters by at least one operator?

(h) Are all single letters and all integers followed by something other than '('?

Except for (e), all conditions must be satisfied by the problem. If (e) is not satisfied, the variable will be determined as described in

Section 2.1 of the previous chapter.

To summarize what we have discussed, let us consider the following example:

DIFF 2*X - 3 + SIN(X + 1))? - 4X

This problem violates (a) because the left and right parentheses are not matched. The function also contains an unacceptable character, '?', and '4' is followed by 'X' ('4X' must be written as '4*X') so that (d) and (g) are also violated.

Although a problem may contain more than one error, such as in the example given above, the analyzer will terminate the problem when the first input error is discovered. Consequently, only the message for the first input error to be discovered will be printed (by the writer).

Before we proceed to the description of part 2, we first explain the meaning of some of the variables to be used.

FLAG An array with 3 entries. The first entry is either 33 or 99. If it is 33, the problem is accepted, if it is 99, the problem is rejected. The second entry is used to contain an error number. The third entry is a non-negative integer, and is needed for integration only. The third entry is not used for differentiation and is set to 1. For integration it is set to 1 if the "function" to be integrated has only one term. If the "function" has more than one term recall that it is integrated term by term. For the first term the third

entry equals the number of terms. It is set to 0, to denote continuation, for all remaining terms.

ALPHACONST A stack which contains the integers representing the variable and all alphabetic constants encountered in the "function".

STORE0 A stack with 2 entries. If it is a differentiation problem, the first entry contains the total number of function names and expressions of the form 'number** f(x)' (not necessary distinct) encountered in the "function". If it is an integration problem, the first entry contains the number of highest level distinct function names encountered in the "function". The second entry is used to indicate whether the problem is a differentiation problem or an integration problem. If it is a differentiation problem, the entry is 0. Otherwise if it is an integration problem, the entry is 1.

STORE1 A stack which contains integers representing function names which have been replaced by Z(I)'s. In brief, if a function name is replaced by Z(I), the I^{th} entry of STORE1 will contain the integer corresponding to that function name. For example, COS is represented by 2 so that if COS was replaced by Z(5), then STORE1(5) = 2. Appendix E lists all elementary functions and their corresponding integers.

STORE2 A stack. If it is a differentiation problem, the

"function" is checked for the form 'number**f(x)'. For each such form, say '20**SIN(X)', the number '20' will be stored in STORE2. If it is an integration problem, each entry of STORE2 indicates the number of appearances of the function name stored in the corresponding entry of STORE1.

STORE3 A stack which contains the arguments of all the function names encountered in the "function".

STORE6 A stack. For a differentiation problem, each entry of STORE6 indicates the number of top level function names that appear in the corresponding entry of STORE3. For an integration problem, if the "function" to be integrated involves nesting (see Definition 2.2), the entry will contain the integer that represents the nested function name. Otherwise, if there is no nesting of function, the entry is 0.

STORE7 A stack. For differentiation, each entry of STORE7 indicates the accumulated total of function names that have been replaced by Z(I)'s, before the function names in the corresponding entry of STORE3 are replaced. For integration, if the "function" is a special function, STORE7 will be used to store the coefficients of the special function (see Appendix F).

FUNCTION The "function" (may be a term of the initial function) to be differentiated or integrated.

Variables and alphabetic constants: All single characters in the "function" are represented internally according to their alphabetical order with integers starting from 0. Thus A is represented internally by 0, B is by 1, and so on.

NUMBER2: An integer variable which is equal to 0 if it is a differentiation problem and 1 if it is an integration problem.

We are now ready to discuss the differentiation portion of part 2 of the analyzer. For our convenience, we shall illustrate various points by working on the following example:

Example 3.1

```
DIFF   X*EXP(4 + A*X) + SIN(1 - COS(X))*10**(X - 2)**2
```

Part 2 (Differentiation). Generate internal forms.

(a) Set NUMBER2 = 0 (differentiation).

(b) Is the form 'number**f(x)' present in FUNCTION? If so, replace it by a dummy function name, which is formed by attaching the "number" to the right hand side of 'CC', the argument of the dummy function name is f(x). Repeat for all such forms.

Thus, for Example 3.1,

```
FUNCTION=X*EXP(4+A*X)+SIN(1-COS(X))*CC10(X-2)**2
```

(c) Scan FUNCTION from left to right. For each function name encountered, replace the function name by 'Z(I)' and add 1 to I. Place the argument of the replaced function name in STORE3 and place the integer

corresponding to it in STORE1 (dummy name is internally represented by '100').Thus, for each I, 'Z(I)' replaces one and only one function name. Identical function names, even those having the same argument, are replaced by different 'Z(I)''s; if the function name replaced is a dummy name, then the digit part of the dummy name is also stored in STORE2.

Finally, place the current accumulated total of replaced function names (including the replaced dummy names) in STORE7.

Thus, for Example 3.1,

STORE1 = (26,1,100,

STORE2 = (10,

STORE3 = (4 + A*X,1 - COS(X),X - 2,

STORE7 = (3,

FUNCTION = Z0*Z(1) + Z(2)*Z(3)**2

(d) Scan STORE3 from left to right, for each entry:

(1) Repeat step (c).

(2) Place the number of top level function names that are replaced by 'Z(I)''s in STORE6.

Thus, for Example 3.1:

STORE1 = (26,1,100,2,

STORE3 = (4 + A*X, 1 - Z(4), X - 2,X,

STORE6 = (0,1,0,0,

STORE7 = (3,3,4,4,4,

(e) Place the total number of function names (including the dummy names) replaced in step (c) and step (d) in

STORE0.

Thus, at the end of part 2, we have, for Example 3.1,

FLAG = (0,0,0)

STORE0 = (4,

STORE1 = (26,1,100,2,

STORE2 = (10,

STORE3 = (4 + A*X, 1 - Z(4), X - 2, X,

STORE6 = (0,1,0,0,

STORE7 = (3,3,4,4,4,

FUNCTION = Z0*Z(1) + Z(2)*Z(3)**2

Having completed the description of the differentiation portion of part 2, we now proceed to the integration portion of the same part. Before beginning our discussion, we first explain the meaning of several new variables which are used in addition to those introduced in the beginning of the differentiation portion.

STORE4 A stack which is used whenever the form 'number**f(x)' or the form 'function name (number)' is discovered in FUNCTION. For the first form, say '20**f(x)', the number '27' (which represents LN) will be placed in STORE4 followed by '20'. For the second form, say 'SQRT(50)', the number '28' (which represents SQRT) will be placed in STORE4 followed by '50'.

JACK An integer which indicates the basic form of FUNCTION according to its elementary function composition.

KEK An integer which indicates whether FUNCTION involves a hyperbolic function. If its value is 1, then FUNCTION involves a hyperbolic function. If its value is 0, FUNCTION does not involve any hyperbolic function.

TYPE An integer whose value will be ≥-1 on completion of problem analysis. If its value is -1, the form 'number ** f(x)' was found in FUNCTION. If its value is a positive integer, FUNCTION was found to be a special function, and the value of TYPE is exactly the type number of the special function. If its value is 0, FUNCTION satisfies neither of the above cases.

As in our discussion of the differentiation portion of part 2, we shall illustrate some specific points with the aid of the following example.

Example 3.2

INT SQRT(COS(X))*SIN(X)*COS(X)*D*LN(10) - SIN(X - 1)

Part 2 (Integration). Generate internal form.

(a) Set NUMBER2 = 1 (Integration).

If the "function" contains more than one term break it into individual terms (see Section 2.3) and place these terms in the array PROBLEM.

Thus, for Example 3.2,

PROBLEM=(SQRT(COS(X))*SIN(X)*COS(X)*D*LN(10),SIN(X-1),

Set "problemno" = 1.

Set "multiple" equal to the number of terms.

(b) Set FUNCTION = PROBLEM(problemno).

Thus,

FUNCTION=SQRT(COS(X))*SIN(X)*COS(X)*D*LN(10)

(c) Inspect FUNCTION. Are there any of the following constructions present?

(I) 'number**(f(x))'

(II) 'function name(number)'

(III) 'SQRT(1 + M**2*SIN(f(x))**2)'

(where M is a positive integer)

(IV) 'SIN(LN(f(x)))' or 'COS(LN(f(x)))'

(V) 'LN(g(x) ± SQRT(f(x)**2 ± M**2))'

(VI) 'SQRT(f(x)**2 ± M**2)'

(VII) 'SQRT(M**2 ± f(x)**2)'

If any of the above constructions are discovered in FUNCTION do:

For (I). Replace 'number**f(x)' by 'EXP(f(x))' and set TYPE = -1. TYPE will be used in an ALTRAN routine to indicate that the actual argument of EXP is in fact 'LN(number)*f(x)'. Place '27' followed by the"number"in STORE4.

If the form 'number**f(x)' occurs more than once, then the value of "number" must be the same.

For (II). For each occurrence of the form 'function name(number)' do:

(1) Replace 'functionname(number)' by 'ZZ(I)',

where I is an integer starting from 1.

(2) Place the integer corresponding to the "functionname" in STORE4 followed by the "number".

(3) Add 1 to I.

Thus, for Example 3.2,

FUNCTION=SQRT(COS(X))*SIN(X)*COS(X)*D*ZZ(1)

STORE4 = (27,10,

For (III). Replace the form 'SQRT(1 + M**2*SIN(f(x))**2)' by an intermediate form '(1 + M**2*VV(f(x))**2)'. The operator 'VV' conveys to later steps that both SQRT and SIN are involved in the original form.

For (IV). Both of 'SIN(LN(f(x)))'and 'COS(LN(f(x)))' are special functions. Replace 'SIN(LN(f(x)))' by 'SIN(f(x))' or 'COS(LN(f(x)))' by 'COS(f(x))', set TYPE = 1000. (this indicates the FUNCTION is a special function).

For (V).

If signs = (+,+), set TYPE = -1.

If signs = (+,-), set TYPE = -2.

If signs = (-,+), set TYPE = -3.

If signs = (-,-), set TYPE = -4.

If g(x) and f(x) are elementary functions, they must be the same functions (their arguments are not checked here, but they must be equal. The processor will check whether the arguments are equal).

Thus, if g(x) = SIN(X-1) and f(x) = SIN(X-1), replace (V) by the character string 'LN(SIN(M,X-1,X-1))'. Otherwise if g(x) and f(x) are not elementary functions, replace (V) by the character string 'LN(M,g(x),f(x))'. These character strings are transformed again in later parts of the analyzer into their ALTRAN comprehensible form.

For (VI).

If sign = '+' set TYPE = -3

otherwise set TYPE = -2.

If f(x) is an elementary function, say f(x) = SIN(X-1), then replace (VI) by the character string

'SQRT(SIN(X**2 ± M**2,M,(X-1)))'

otherwise replace (VI) by the character string

'SQRT(f(x)**2 ± M**2,M,f(x))'

For (VII).

If sign = '+' set TYPE = -3

otherwise set TYPE = -1.

If f(x) is an elementary function, say f(x)=SIN(2-X), replace (VII) by the character string 'SQRT(SIN(M**2 ± X**2,M,(2-X)))'. Otherwise replace (VII) by the character string 'SQRT(M**2 ± f(x)**2,M,f(x))'

(d) Is there any nesting of elementary functions? If yes, is the nested function name acceptable to the package? If more than one top level function contains

a nested function name, then all the nested function names must be the same (the arguments of these nested functions are checked in the processor and they must also be the same). For example, the function

```
TAN(X)*SIN(TAN(X))*COS(TAN(X))*SEC(X)**2
```

satisfies this condition because the nested function names of SIN and COS are both equal to TAN.

Place the integer corresponding to the nested function name (Appendix E) in STORE6.

Thus,

```
STORE6 = (2,
```

For all top level functions which contain the nested function name, erase the nested function and replace the left bracket which follows by '<'.

Thus, for Example 3.2,

```
FUNCTION = SQRT(<X))*SIN(X)*COS(X)*D*ZZ(1)
```

Replace all other occurrences of the nested function name by a character string of the form 'ANA' where N is a positive integer starting from 1.

Thus,

```
FUNCTION = SQRT(<X))*SIN(X)*A1A(X)*D*ZZ(1)
```

Now, all function names that remain in the resulting "function", except those containing the nested function, must be identical to the function name appearing in the derivative of the nested function but distinct from the differential function. These function names are all replaced by 'A0A'.

Thus, for example 3.2

FUNCTION=SQRT(<X))*A0A(X)*A1A(X)*D*ZZ(1)

(e) Replace EXP by 'T0T' throughout FUNCTION (this will cause EXP to be the last function name to be replaced by a 'Z(I)').

(f) Express the elementary functions CSC, SEC, TAN, COT, CSCH, SECH, TANH and COTH in terms of SIN and COS or SINH and COSH. The transformations for TAN, COT, TANH and COTH are done in a little different way. For example, 'TAN(X)' is replaced by the character string 'SIN_COS(X)' rather than by the exact expression 'SIN(X)/COS(X)'. All other functions are transformed exactly. For example, 'SEC(X)' is replaced by '1/COS(X)'.

(g) Scan FUNCTION from left to right for a function name,

(1) Replace the function name by Z(I) and add 1 to I. Place the argument of the function name in STORE3.

(2) Scan FUNCTION from left to right. For each identical function name encountered do (1).

(3) Place the integer corresponding to the function name in STORE1 and place the total number of occurrences of the function name in STORE2.

(4) Add 1 to the "group number" to which the function name belongs. Group numbers are integer

variables used to determine whether the function composition of FUNCTION is within the limits allowed by the package. The variable GROUP1 is associated with the function EXP. GROUP2 is associated with the set of functions containing LN, LOG, SQRT and any inverse function. GROUP3 and GROUP4 are associated with trigonometric functions and hyperbolic functions, respectively.

(5) If any more distinct function names in FUNCTION repeat (g).

For Example 3.2, we obtain

STORE1 = (28,

STORE2 = (1,

STORE3 = (X,

FUNCTION = Z(1)*A0A(X)*A1A(X)*D*ZZ(1)

(h) Place the total number of distinct function names processed in (g) in STORE0.

Thus,

STORE0 = (1,

(i) Inspect "group number". If the elementary function composition of FUNCTION is beyond specified limits set an error flag. For example, if FUNCTION is a product of a trigonometric function and a hyperbolic function, both GROUP3 (representing trigonometric functions) and GROUP4 (representing hyperbolic functions) will be positive. When such a situation is recognized, the problem will be rejected

since FUNCTION is beyond the package capabilities.

(j) Replace 'A0A''s by 'Z_2' in FUNCTION, store '99' in STORE1 ('99' represents a dummy name) and place the arguments in STORE3. Likewise replace 'ANA''s by 'Z3', store '99' in STORE1 and place the arguments in STORE3.

Thus,

STORE1 = (28,99,99,

STORE3 = (X,X,X,

FUNCTION = Z(1)*Z2*Z3*D*ZZ(1)

(k) Determine the basic form of FUNCTION using "group number" and TYPE and place the integer corresponding to that basic form in JACK. The value of KEK is also set (see Definition of KEK).

Thus,

JACK = 14

KEK = 0

(l) If JACK = 1 and KEK = 0, is FUNCTION a special function? If yes, then set TYPE equal to the integer corresponding to the special function. Place relevant coefficients of the special function in STORE7 (see Appendix F). Replace the special function by '1'. If no, go to (m).

(m) If JACK $\neq$ 1 or KEK $\neq$ 0 set TYPE = 0.

Steps (l) and (m) complete part 2 (integration). At this point Example 3.2 would appear as follows:

FLAG = (0,0,2)

STORE0 = (1,

STORE1 = (28,99,99,

STORE2 = (1,

STORE3 = (X,X,X,

STORE4 = (27,10,

STORE6 = (2,

JACK = 14

KEK = 0

FUNCTION = Z(1)*Z2*Z3*D*ZZ(1)

Part 3. Complete the internal representation.

(a) Replace the variable by 'Z0' throughout FUNCTION and STORE3. Place the integer corresponding to the variable in ALPHACONST.

Thus, for Example 3.1,

STORE3 = (4 + A*Z0,1 - Z(4),Z0 - 2,Z0,

ALPHACONST = (23,

and for Example 3.2,

STORE3 = (Z0,Z0,Z0,

ALPHACONST = (23,

(b) Replace the alphabetic constants by Z1(I)'s throughout FUNCTION and STORE3 and place the integers corresponding to them in ALPHACONST. Generally, if an alphabetic constant is replaced by 'Z1(I)', the $(I+1)^{th}$ entry of ALPHACONST will contain the integer corresponding to that

alphabetic constant (note that, identical alphabetic constants in a given function are replaced by the same 'Z1(I)').

Thus, for Example 3.1,

STORE3 = (4 + Z1(1)*Z0,1 - Z(4),Z0 - 2,Z0,

ALPHACONST = (23,1,

and for Example 3.2,

FUNCTION = Z(1)*Z2*Z3*Z1(1)*ZZ(1)

ALPHACONST = (23,3,

(c) If no error flag do (1) otherwise do (2):

(1) Place '33' in the first entry of FLAG. If it is the first subproblem place "multiple" in the third entry of FLAG. If it is any other subproblem place '0' in the third entry of FLAG.

Thus, for Example 3.1,

FLAG = (33,0,1)

and for Example 3.2,

FLAG = (33,0,2)

(2) Place the error flag in the second entry of FLAG, the first entry of FLAG is set to '99'. Go to step (h).

(d) Place NUMBER2 in STORE0.

Thus, for Example 3.1,

STORE0 = (3,0)

and for Example 3.2,

STORE0 = (1,1)

(e) Close the stacks by adding a pair of '99' to the end of each stack.

(f) Output variables and stacks.

(g) If the problem processed is an integration problem and "problemno" is less then "multiple" add 1 to "problemno" and go to step(b) of part 2 (integration).

(h) Go to step(a) of part 1 for the next problem.

The complete list of variables and arrays produced by the analyzer for Example 3.1 and Example 3.2 are given in Figures 3.1 and 3.2.

```
FLAG        =    (33,0,1)
ALPHACONST  =    (23,0,99,99)
STORE0      =    (3,0)
STORE1      =    (26,1,100,2,99,99)
STORE2      =    (10,99,99)
STORE3      =    (4 + Z1(1)*Z0,1 - Z(4),Z0 - 2,Z0,99,99)
STORE4      =    (99,99)
STORE6      =    (0,1,0,0,99,99)
STORE7      =    (3,3,4,4,4,99,99)
FUNCTION    =    Z0*Z(1) + Z(2)*Z(3)**2
JACK        =    0
KEK         =    0
TYPE        =    0
```

Figure 3.1. : Output produced by the analyzer for Example 3.1.

```
FLAG          =    (33,0,2)
ALPHACONST    =    (23,3,99,99)
STORE0        =    (1,1)
STORE1        =    (28,99,99,99,99)
STORE2        =    (1,99,99)
STORE3        =    (Z0,Z0,Z0,99,99)
STORE4        =    (27,10,99,99)
STORE6        =    (2,99,99)
STORE7        =    (99,99)
FUNCTION      =    Z(1)*Z2*Z3*Z1(1)*ZZ(1)
JACK          =    14
KEK           =    0
TYPE          =    0
```

Figure 3.2 : Output produced by the analyzer for Example 3.2.

Before leaving this section we present two more examples and the output produced by the analyzer for each of them.

Example 3.3

```
DIFF SIN(X) - COS(X**2-1)*SIN(X-1)
```

and

Example 3.4

```
INT 16/(4*COS(X-1) - 5*SIN(X-1))
```

Variables	Example 3.3	Example 3.4
FLAG	(33,0,1)	(33,0,1)
ALPHACONST	(23,99,99)	(23,99,99)
STORE0	(3,0)	(2,1)
STORE1	(1,2,1,99,99)	(2,1,99,99)
STORE2	(99,99)	(1,1,99,99)
STORE3	(Z0,Z0**2-1,Z0-1,99,99)	(Z0-1,Z0-1,99,99)
STORE4	(99,99)	(99,99)
STORE6	(0,0,0,99,99)	(0,99,99)
STORE7	(3,3,3,3,99,99)	(4,-5,99,99)
FUNCTION	Z(1) - Z(2)*Z(3)	16*1
JACK	0	1
KEK	0	0
TYPE	0	100

Figure 3.3 : Output produced by the analyzer for Example 3.3 and Example 3.4.

It is worth noting that in Example 3.4 the "function" is a special function. The relevant coefficients of the "function" are stored in STORE7. The value of TYPE is 100 which is the type number corresponding to the special function.

3.3. The Processor-Differentiation.

In the processor step, all differentiation problems are handled by calling on a single ALTRAN procedure named SYMDIF. In designing this procedure we have made use of the ALTRAN built-in differentiation routine DIFF. This latter routine can be used to compute the derivative of any rational function in a simple variable.

Recall, from Section 3.2, that the "function" which is to be differentiated is translated into an internal representation by the analyzer. Elementary functions that appear in the "function" are replaced by simple variables, 'Z(I)'s. Thus we can consider the "function" to be a rational function, FCT, in the simple variables Z0, Z(1), Z(2),..., Z(N). Z0 is the variable with respect to which FCT is to be differentiated and Z(1), Z(2),..., Z(N) are in fact functions of Z0.

The derivative of the function FCT is obtained using the following rule:

Rule (I). $$\frac{dFCT}{dZ0} = \frac{\partial FCT}{\partial Z0} + \sum_{I=1}^{N} \frac{\partial FCT}{\partial Z(I)} \cdot \frac{dZ(I)}{dZ0}$$

The term $\frac{\partial FCT}{\partial Z0}$ can be obtained by making a call to the procedure DIFF with the statement DIFF(FCT,Z0). Similiarly, for each I, the term $\frac{\partial FCT}{\partial Z(I)}$ can be obtained by making the call DIFF(FCT,Z(I)). Thus, Rule (I) is equivalent to:

$$\frac{dFCT}{dZ0} = DIFF(FCT,Z0) + \sum_{I=1}^{N} DIFF(FCT,Z(I)) \cdot \frac{dZ(I)}{dZ0}$$

Finally, $\frac{dZ(I)}{dZ0}$ is to be obtained by using the chain rule as follows:

$$\text{Rule (II).} \quad \frac{dZ(I)}{dZ0} = ZZ(I) \cdot \frac{d}{dZ0}(\arg(Z(I)))$$

where ZZ(I) denotes $\frac{dZ(I)}{d(\arg(Z(I)))}$ (Z(I) is an elementary function), and arg(Z(I)) denotes the argument of the elementary function Z(I).

Substitute this equation into Rule (I) yields:

$$\text{Rule (III).} \quad \frac{dFCT}{dZ0} = DIFF(FCT,Z0) + \sum_{I=1}^{N} DIFF(FCT,Z(I)) \cdot ZZ(I) \cdot \frac{d}{dZ0}(\arg(Z(I))).$$

The actual elementary function(s) which correspond to each ZZ(I) are not determined in the processor step. The appropriate elementary function(s) are determined by the writer when the user comprehensible form is produced by matching subscripts between Z(I)'s and ZZ(I)'s.

Recall that the argument of any elementary function in the "function" is allowed to contain another function name, provided the Conditions (a)-(e) and Condition (f) stated in the Section 2.1 and Section 2.2 are all satisfied. Thus, Rules (I) and (II) may be used repeatedly during the computations of the derivative of a particular function. As an illustration of this last point, consider the following example:

```
DIFF 2*SIN(1-COS(X**2-1)).
```

The value of FCT is equal to '2*Z(1)' where 'Z(1)' represents SIN, the argument of SIN is represented by '1-Z(2)'.

Thus, by applying Rule (I),

$$\frac{dFCT}{dZ0} = 0 + \frac{\partial FCT}{\partial Z(1)} \cdot \frac{dZ(1)}{dZ0} \qquad (N = 1 \text{ in this case})$$

$$= 2 \cdot \frac{dZ(1)}{dZ0}$$

Applying Rule (II) on $\frac{dZ(1)}{dZ0}$ yields:

$$ZZ(1) \cdot \frac{d}{dZ0} (1-Z(2))$$

Here, $\frac{d}{dZ0} (1-Z(2))$ is to be computed using Rules (I) and (II) and thus we obtain

$$\frac{d}{dZ0} (1-Z(2)) = -ZZ(2) \cdot \frac{d}{dZ0} (Z0^2 - 1).$$

One final application of Rule (I) completes the computation of the derivative and we obtain

$$\frac{dFCT}{dZ0} = -2 \cdot ZZ(1) \cdot ZZ(2) \cdot 2 \cdot Z0$$

$$= -4 \cdot Z0 \cdot ZZ(1) \cdot ZZ(2) .$$

With this brief overview of how differentiation is performed we are ready to give a detailed description of SYMDIF. Before we proceed to that material, we first introduce the set of variables

used in that procedure.

M1 An integer array equal to STORE0

M4 An integer array equal to STORE6

M6 An integer array equal to STORE7

MB An algebraic array equal to STORE3

FCT An algebraic variable equal to FUNCTION

RESULT An algebraic variable used to contain the derivative of FCT

We again use Example 3.1 to show how the corresponding ALTRAN comprehensible form produced by the analyzer is processed in the differentiation routine SYMDIF.

Before the procedure SYMDIF is called, the variables and arrays defined above acquire the following values (see Figure 3.1). This is done through a read statement in the main procedure of the processor.

```
M1   = (3,0)
M4   = (0,1,0,0,99,99)
M6   = (3,3,4,4,4,99,99)
MB   = (4 + Z1(1)*Z0, 1 - Z(4),Z0 - 2,Z0,99,99)
FCT  = Z0*Z(1) + Z(2)*Z(3)**2
```

Several other variables and arrays, which are also read at this time are used for passing values to the writer. They will be discussed later in this section.

The algorithm employed in SYMDIF is given by the equation stated in Rule (III). The computation process can be described as follows:

1. Set RESULT = DIFF(FCT,Z0)
2. If M1(1) = 0 the derivative has been obtained and so return. (M1(1) denotes the number of top level function names in FCT).
3. Set SAVE1 = 0.
4. Set I = 1.
5. (a) Set OSAVE1 = DIFF(FCT,Z(I))*ZZ(I).
6. Set OSAVE2 = DIFF(MB(I),Z0) and SAVE3 = 0.
7. If M4(I) = 0 the argument of Z(I), namely MB(I), does not contain any function name, go to (g).
8. Set I1 = M4(I), K1 = M6(I).
9. Set J = 1.
10. (b) Set OSAVE3 = DIFF(MB(I),Z(K1 + J))*ZZ(K1 + J).
11. Set OSAVE4 = DIFF(MB(K1 + J), Z0) and SAVE 5 = 0.
12. If M4(K1 + J) = 0 the argument of Z(K1 + J), MB(K1+J), does not contain any function name, go to (f).
13. Set I2 = M4(K1 + J), K2 = M6(K1 + J).
14. Set K = 1
15. (c) Set OSAVE5 = DIFF(MB(K1 + J), Z(K2 + K))*ZZ(K2 + K)
16. Set OSAVE6 = DIFF(MB(K2 + K),Z0) and SAVE8 = 0.
17. If M4(K2 + K) = 0 the argument of Z(K2 + K), MB(K2 + K) does not contain any function name, go to (e).
18. Set I3 = M4(K2 + K), K3 = M6(K2 + K).
19. Set L = 1.
20. (d) Set OSAVE7 = DIFF(MB(K2 + K),Z(K3 + L))*ZZ(K3 + L).
21. Set OSAVE8 = DIFF(MB(K3 + L), Z0).

22. SAVE8 = SAVE8 + OSAVE7*OSAVE8.

23. Add 1 to L. If L is not greater than I3,go to (d).

24. (e) SAVE5 = SAVE5 + OSAVE5*(OSAVE6 + SAVE8).

25. Add 1 to K. If K is not greater than I2,go to (c).

26. (f) SAVE3 = SAVE3 + OSAVE3*(OSAVE4 + SAVE5)

27. Add 1 to J. If J is not greater than I1,go to (b).

28. (g) SAVE1 = SAVE1 + OSAVE1*(OSAVE2 + SAVE3).

29. Add 1 to I. If I is not greater than M1(1),go to (a).

30. RESULT = RESULT + SAVE1

31. Return.

In order to illustrate how the above algorithm works, we will follow the processing of Example 3.1.

Lines	Consequences
1.	RESULT = Z(1)
3.	SAVE1 = 0
4.	I = 1
5.	OSAVE1 = Z0*ZZ(1)
6.	OSAVE2 = Z1(1), SAVE3 = 0
7.	Since M4(1) = 0 go to line 28.
28.	SAVE1 = 0 + Z0*ZZ(1)*(Z1(1) + 0)
	= Z0*ZZ(1)*Z1(1).
29.	I = 2, since I is not greater than M1(1)(=3) go to line 5.
5.	OSAVE1 = Z(3)**2*ZZ(2).
6.	OSAVE2 = 0.
7.	Since M4(2) $\neq$ 0 continue on line 8.
8.	I1 = 1, K1 = 3.

9. J = 1

10. OSAVE3 = -ZZ(4).

11. OSAVE4 = 1, SAVE5 = 0.

12. M4(4) = 0 go to line 26.

26. SAVE3 = 0 + (-1)*(1 + 0) = -1 .

27. J = 2. J is greater than I1(=1).

28. SAVE1 = Z0*ZZ(1)*Z1(1)+Z(3)**2*ZZ(2)*
(0+(-1)) = Z0*ZZ(1)*Z1(1)-Z(3)**2*ZZ(2)*ZZ(4)

29. I = 3 go to 5.

5. OSAVE1 = 2*Z(2)*Z(3)*ZZ(3).

6. OSAVE2 = 1, SAVE3 = 0.

7. M4(3) = 0 go to line 28.

28. SAVE1 = Z0*ZZ(1)*Z1(1) - Z(3)**2*ZZ(2)
+ 2*Z(2)*Z(3)*ZZ(3)*(1 + 0)
= Z0*ZZ(1)*Z1(1)-Z(3)**2*ZZ(2)*ZZ(4)
+2*Z(2)*Z(3)*ZZ(3).

29. I = 4. I is greater than M1(1).

30. RESULT = Z(1) + Z0*Z1(1)*ZZ(1)
+ 2*Z(2)*Z(3)*ZZ(3)
- Z(3)**2*ZZ(2)*ZZ(4).

31. Return to the main procedure.

After the final value of RESULT is obtained, control is returned to the main procedure of the processor. The set of variables and arrays that are ready to be passed to the writer by the main procedure are:

STATUS	An integer array equal to FLAG
M0	An integer array equal to ALPHACONST
M1	As before
MA	An integer array equal to STORE1
MB	As before
RESULT	As before
M3	An integer array equal to STORE2

Thus, for Example 3.1, the values of the variables and arrays produced by the processor are as follows :

```
STATUS  =  (33,0,1)
M0      =  (23,0,99,99)
M1      =  (3,0)
MA      =  (26,1,100,2,99,99)
MB      =  (Z0*Z1(1) + 4, -(Z(4)-1), Z0 - 2, Z0, 99, 99)
RESULT  =  Z(1) + 2*Z(2)*Z(3)*ZZ(3) - Z(3)**2*ZZ(2)*ZZ(4) +
           Z0*Z1(1)*ZZ(1)
M3      =  (10,99,99)
```

Figure 3.4. : Output produced by the processor for Example 3.1.

Using the algorithm we have just described, the reader might compute the derivative for Example 3.3 in exactly the same way. The corresponding output produced by the processor is shown as follows.

```
STATUS  = (33,0,1)

M0      = (23,99,99)

M1      = (3,0)

MA      = (1,2,1,99,99)

MB      = (Z0, Z0**2 - 1, Z0 - 1, 99,99)

RESULT  = -(Z(2)*ZZ(3) + 2*Z(3)*Z0*ZZ(2) - ZZ(1))

M3      = (99,99)
```

Figure 3.5. : Output produced by the processor for Example 3.3.

3.4 The Processor - Integration

In Section 2.3, we presented the 15 forms of functions which can be integrated by SYMDIP. The integral of a given function matching any of these forms, except for Form 15, is computed by using a set of predetermined integration steps. As we indicated in Section 2.3, one step in computing the integral of a function of Form 2C was to perform an "integration by parts". A function matching Form 15 (i.e. a special function), is integrated with the aid of an integration table which provides the terms of the eventual integral. Only the coefficients that are associated with these terms are computed in the processor step.

The description of the general integration procedure will be presented in several parts. Before proceeding to that description, the set of variables which are involved in the processor are introduced. Variable names which are used in these definitions refer to variables which were defined in Section 3.2.

MA An integer array equal to STORE1.

MB An algebraic array of arguments equal to STORE3.

M0 An integer array equal to ALPHACONST.

M1 An integer array equal to STORE0.

M2 An integer array which is used for indirect representation of the function names in the computed integral.
If a function name in RESULT (see below for details) is replaced by 'Z(I)', where I is an integer greater than 6, then the $(I-6)^{th}$ entry of M2 will contain the integer corresponding to that replaced function name. M2 is initialized to 0 for each problem.

M3 An integer array equal to STORE2.

M4 An integer array equal to STORE6.

M5 An algebraic array of arguments. Each entry of M5 contains the argument of the function name which is indirectly represented in the computed integral. Generally, the I^{th} entry of M5 is the argument of the function to which the I^{th} entry of M2 corresponds, i.e. the function represented by Z(I+6) in RESULT. Like M2, array M5 is also initialized to 0 for each problem.

M6 An integer array equal to STORE7.

M7 An integer variable equal to TYPE.

M8 An integer array equal to STORE4.

LK An integer which denotes the next available location in M2 (and also in M5).

EXP An algebraic expression which contains the constant obtained by dividing the argument of the function EXP by the value of SER (see below).

KEK An integer variable same as before.

LAW An integer variable used to denote transformation of the integrand.

LAW = 2: The integrand has been transformed by Y = EXP(f(x)).

LAW = 3: The integrand has been transformed by Y = TAN(f(x)/2).

LAW = 4: The integrand has been transformed by Y = SQRT(f(x)).

LAW = 0: No transformation of any of the above kinds is made.

JACK An integer variable same as before.

SER An algebraic variable. The initial value of SER is set to the first entry of MB, except in the following cases:

Case (I). If JACK > 5 and JACK < 10, set SER = MB(2).

Case (II). If JACK > 10 and JACK < 14, set SER = MB(3).

During the computations of an integral the value of SER will be changed if there is a change of variable. For example, if the function to be integrated is:

SIN(X)*COS(X) ,

the initial value of SER is 'Z0' (the variable is always represented by 'Z0'). To integrate this function, SYMDIP converts it into the following function :

1/2*SIN(Y) .

where Y='2*X' by using an obvious trigonometric identity. The function will now be integrated with respect to Y, which is also represented by 'Z0'. The value of SER, in this case, is changed to '2*Z0'. As a second example, if the function to be integrated is

SINH(SIN(X-3))*COS(X-3) ,

SYMDIP will convert the function into

SINH(Y)

and integrate it with respect to Y=SIN(X-3). In this particular case, the value of SER which is initially set to 'Z0-3' is changed to 'Z0' (the new variable is also represented by 'Z0'). The integer which corresponds to the nested function name, SIN, is stored in MSER, and the argument, 'Z0-3', of the nested function is stored in NSER.

FCT An algebraic variable which is equal to FUNCTION.

PARM An algebraic variable which is used as follows:

(I) If the function to be integrated is a special function (Form 15) PARM will be set to the coefficient of the

second term of the integral, if that term exists. In this case, PARM will be used by the writer.

(II) If the function to be integrated belongs to or is converted to a function of Form 3, then PARM is set to the coefficient of the last term of the integral, if that term is to be obtained from an integration table. In this case, PARM will also be used by the writer.

(III) Otherwise, PARM is used to contain the coefficient of the next term of the integral.

RESULT An algebraic variable used to contain the integral. Function names are represented in RESULT as follows:

Type 1. Direct representation. A function name is directly represented according to the following rules:

1. Z(1): Represents SIN if KEK=0 and SINH if KEK=1
2. Z(2): Represents COS if KEK=0 and COSH if KEK=1
3. Z(3): Represents EXP
4. Z(4): Represents LN
5. Z(5): Represents SQRT
6. Z3: Represents TAN

(the argument of each of these functions is found in SER)

Type 2. Indirect representation. For each Z(I), I > 6, Z(I) represents the function name to which the $(I-6)^{th}$ entry of M2 corresponds. The argument of the function name is M5(I-6).

Type 3. Dummy representation. Z(6) is used to represent SER in RESULT. The use of Z(6) in place of SER prevents terms such as SER**7 from being expanded unnecessarily.

Thus, if RESULT has the following value

2*Z(3)*Z(5)*Z(6) with SER = '7*Z0';

it will be translated by the writer into

'2*EXP(7*Z0)*SQRT(7*Z0)*(7*Z0)', and Z0 is a variable whose form is determined by the value of MSER (see below).

OSER An algebraic variable used when DELAY=1 (see below). OSER is the variable with respect to which the second part of the FUNCTION is integrated. For example, if the function SIN(X)*COS(4*X) is to be integrated, it will first be converted into SIN(5 * X)/2+SIN(-3 * X)/2 by using the rule SIN(A+B)+SIN(A-B)=2SIN(A)*COS(B). The second part of the function, 1/2*SIN(-3*X), is then integrated with respect to Y='-3*X'. Thus, OSER='-3*Z0'.

OEXP An algebraic variable used when DELAY=1 (see below). OEXP is set to the constant obtained by dividing the argument of the function EXP by the value of OSER.

DELAY An integer variable which has the following meaning:

DELAY=1: The function to be integrated has been converted into the sum of two parts by one of the following rules:

(I) sin(A+B)+sin(A-B)=2sin(A)cos(B)

(II) sin(A+B)-sin(A-B)=2sin(B)cos(A)

(III) cos(A+B)+cos(A-B)=2cos(A)cos(B)

(IV) cos(A+B)-cos(A-B)=-2sin(A)sin(B)

Each of these 2 parts will be integrated separately. The final integral of the function is then given as the sum of these integrals.

DELAY=2: The value of DELAY is changed from 1 to 2 when both parts of the sum mentioned above match Form 2B but with different values of M.

DELAY=0: No conversion.

OPARM An algebraic variable used when DELAY=1. OPARM is used exactly as PARM except that it is applied to the second part of FCT produced by one of the transformations described in the definition of DELAY.

MSER An integer which indicates nesting. If MSER is 0, the function does not involve any nesting of functions. If MSER is not 0, MSER is the integer corresponding to the nested function name.

NSER An algebraic variable. If MSER is 0 NSER is dummy. Otherwise NSER contains the argument of the nested function name to which MSER corresponds.

STATUS An integer array equal to FLAG. The various possible values of STATUS(1) are interpreted as follows:

STATUS(1)=33: The status of the computations is acceptable at the current moment, or the integral has been completed.

STATUS(1)=55: A form checking in FCT is successful and subsequent steps may continue.

STATUS(1)>99: The value of STATUS(1) indicates the next procedure to be called.

STATUS(1)=99: An error is detected and the computations are terminated. In this case, STATUS(2) contains the integer which corresponds to the error.

In the program, STATUS is declared to have subscripts 0,1,2. For purposes of exposition, we use subscripts of 1,2, and 3 in our description.

A,B,C The rational numbers in the algebraic expression A*X**2+B*X+C.

I An integer variable. During a procedure call I contains the integer corresponding to the function name which occurs in the definition of Forms 2A, 2B, or 2C (see Section 2.3). For

Forms 4 and 6 (DELAY=1), since the function is first converted into the sum of two parts, I contains the integer corresponding to the function name which appears in these parts.

N An integer variable which contains the power of Z(1).

N1 An integer variable equal to N.

N2 An integer variable which contains the power of Z(2).

M The power of the variable with respect to which the function is integrated. For example, in the function X*LN(X), M is equal to 1.

OM An integer variable used when DELAY=1. It is similar to M but applied on the second term of the sum.

D1 An algebraic variable equal to Z(1)**N1

D2 An algebraic variable equal to Z(2)**N2

D3 An algebraic variable equal to FCT/(D1*D2).

Having defined the set of variables which are used for integration in the main procedure of the processor, we can proceed with the description of the integration portion of the processor. The integration procedure can be logically divided into the following component parts which we list in their order of execution.

PART 1. Initialize variables.

PART 2. If the function to be integrated involves nesting of functions or if it is a special function, check the form of the function.

If the function involves nesting of functions, perform the change of variable.

PART 3. If the function is a special function call the necessary routines to compute coefficients of the integral. After

the coefficients are obtained, the next part to be executed is part 7.

PART 4. If the function is not a special function, unify the representation of function names in FCT which have the same argument.

PART 5. Determine the values of N, N1, N2, D1, D2, and D3. If the function has Form 14 change the value of JACK to 14. Compute the integral of the function by making necessary subprocedure calls.

PART 6. If the integral has been computed by a transformation of variables, substitute back into the original variables (see LAW above). If present, replace SQRT(1) and SQRT(0) in RESULT by 1 and 0 respectively.

PART 7. Output the integral and variables that are needed by the writer and proceed to the next problem.

With this brief introduction to the various parts of the integration procedure in mind, we now proceed to a more detailed description. As in the discussion of the analyzer in Section 3.2, specific points will be illustrated by showing the values of variables associated with Example 3.2 as it is being processed.

PART 1. Read STATUS,M0. If STATUS(1) = 999 terminate the processor step. If STATUS(1) = 99 write STATUS,M0 and go to part 1.

Thus, we have for Example 3.2,

STATUS = (33,0,0)

M0 = (23,0,99,99)

M1 = (1,1)

MA = (28,99,99,99,99)

M3 = (1,99,99)

MB = (Z0,Z0,Z0,99,99)

M8 = (27,10,99,99)

M4 = (2,99,99)

M6 = (99,99)

FCT = Z(1)*Z2*Z3*Z1(1)*ZZ(1)

JACK = 14

KEK = 0

M7 = 0

Set PARM = 1

Set SER according to the description given previously.

Initialize other variables.

Thus, for Example 3.2,

SER = 'Z0'

M2 = 0

M5 = 0

LK = 0

LAW = 0

MSER = 0

NSER = 0

DELAY = 0

RESULT = 0

PART 2. If M7 < 1 and M4(1) = 0 go to part 3. Otherwise do the following:

If all the entries except the last two of MB (array of arguments) are not equal set an error flag and terminate the problem (so, go back to part 1 again). Divide FCT by the derivative of SER with respect to 'Z0'. If FCT still contains 'Z0' set an error flag.

Thus, for Example 3.2,

FCT = Z(1)*Z2*Z3*Z1(1)*ZZ(1)

(FCT remains unchanged for this particular case).

(a) If M4(1) = 0 go to part 3. Otherwise do:

Set OM = M4(1)-KEK * 2

Thus, for Example 3.2,

OM = 2

(1) If OM is equal to 5 (the nested function is TAN or TANH) or OM equal to 6 (the nested function is COT or COTH) set FCT = FCT/Z2**2 (in the case of TAN, the derivative of TAN(X) is SEC(X)**2, and SEC(X) is represented by 'Z2' in FCT. The other 3 cases are similar).

(2) Otherwise if OM is equal to 3 (the nested function is CSC or CSCH) or OM equal to 4 (the nested function is SEC or SECH) set FCT = FCT/(Z2*Z3) (in the case of CSC, the derivative of CSC(X) is -CSC(X)*COT(X), and the functions CSC(X) and COT(X) are represented by 'Z3' and 'Z2', respectively, in FCT. The other 3 cases are similar, except the sign of the derivative, which is treated later).

(3) Otherwise, if OM is equal to 26 (the nested function is EXP) set FCT = FCT/Z3 (note that the derivative of EXP(X) is EXP(X), and it is represented by 'Z3').

(4) Otherwise if OM is equal to 27 (the nested function is LN) set FCT = FCT*MB(1) (note that the derivative of LN(X) is 1/X).

(5) Otherwise, if OM is less than 3 (the nested function is SIN, SINH, COS or COSH) Set FCT = FCT/Z2 (in the case of SIN, the derivative of SIN(X) is COS(X), which is represented by 'Z2'. The other 3 cases are similar, except possibly the sign of the derivative).

Thus, for Example 3.2,

FCT = Z(1)*Z3*Z1(1)*ZZ(1)

Now, if FCT contains any more 'Z2' set an error flag. If OM is equal to 2 (see (5)), or 3 (see (2)) or 6 (see (1)) adjust the sign of FCT where appropriate. Replace 'Z3' by 'Z0' throughout FCT and set:

MSER = M4(1)

NSER = SER

SER = Z0

By now, the necessary procedure to change the variable of integration is complete.

Thus, for Example 3.2 ,

FCT = -Z0*Z(1)*Z1(1)*ZZ(1)

SER = Z0

MSER = 2

NSER = Z0

So, FCT is now expressed in terms of the new variable, Y = COS(X), which is also represented by 'Z0'. Note that the argument of Z(1) (i.e. the function SQRT) is now equal to Y, since SER = 'Z0'.

PART 3. If M7 < 1 go to part 4 otherwise the function to be integrated is a special function. Call procedure SPEFCT which determines one of the four procedures that could handle the particular function. These four procedures are: SPEC1, SPEC2, SPEC3, and SPEC4. For details refer to the documentations of these procedures, as well as, that of the procedure SPEFCT. After part 3 is done, go to part 7.

PART 4. If JACK ≠ 1,2,10,14 go to part 5. If the function contains identical function names which have the same arguments, this part replaces all these function names (Z(I)'s) by a suitable Z(I). Note that identical function names are represented by Z(I)'s in FCT with consecutive values of I. For example, if the function is given as

10+SIN(X+3)*EXP(2*X+6)*SIN(X+3)

then FCT is equal to

10*Z(1)*Z(3)*Z(2).

Since Z(1) and Z(2) represent identical function names having the same arguments, they are all replaced by Z(1). In this particular case, Z(3) is adjusted to Z(2), and FCT becomes

Once such a change is made, the relevant entries of MB and M3 are also adjusted. For the above case, the value of MB is adjusted from (Z0+3,Z0+3,2*Z0+6,99,99) to (Z0+3,2*Z0+6,2*Z0+6,99,99) (the third entry now becomes dummy). Likewise the value of M3 is adjusted from (2,1,99,99) to (1,1,99,99).

PART 5. This is the main part of the integration procedure for functions which are not special functions.

(a) Get the degree of Z(1) in FCT and place it in N1.

Get the degree of Z(2) in FCT and place it in N2.

Set N = N1.

Set D1 = Z(1) ** N1.

Set D2 = Z(2) ** N2.

Set D3 = FCT / (D1 * D2).

Thus for Example 3.2,

N1 = 1

N2 = 0

N = 1

D1 = Z(1)

D2 = 1

D3 = Z0 * Z1(1) * ZZ(1)

(b) If JACK $\neq$ 1 go to (c) otherwise :

If the trigonometric or hyperbolic functions appear in factors containing a single term,go to (c).

Otherwise set JACK equal to 15 (Form 14).

(c) Call subprocedure according to the value of JACK. For

Example 3.2, the procedure named SQRT0 is called. Procedure SQRT0 is a general routine which identifies various forms of functions involving only SQRT. (See the documentation for details). For Example 3.2, the function is transformed by U = SQRT(Y).

Thus,

FCT = -2*Z0**4*Z1(1)*ZZ(1)

(Recall that Y = COS(X)).

Set LAW = 4 and STATUS(1) = 104. Since the value of STATUS(1) is 104, the next procedure to be called is RATFCT which integrates all functions of a simple variable. Note that FCT is to be integrated with respect to the new variable U (also represented by 'Z0').

(d) If STATUS(1) is less than 100 go to (e). Call subroutine according to the value of STATUS(1). For our particular example, procedure RATFCT is called. This procedure in turn calls the ALTRAN built-in integration routine, namely PINT. Generally, if the integral can not be completed, the procedure RATFCT needs to call another procedure, RATFUN, to integrate the remaining part of the function.

Thus, for Example 3.2,

RESULT = -2*Z0**5*Z1(1)*ZZ(1)/5

Since the integral is completed by procedure RATFCT the value of STATUS(1) is set to 33, which indicates the integration has been finished.

(e) If STATUS(1) = 55 FCT has an acceptable form.

Set STATUS(1) = 33 and call subroutine according to the value of JACK.

PART 6. If LAW is not equal to 0 change the variable in RESULT back into the original variable, according to the following:

LAW=2 : Replace 'Z0' in FCT by 'Z(3)' (i.e. EXP).

LAW=3 : Replace 'Z0' in FCT by 'Z3' (i.e. TAN).

LAW=4 : Replace 'Z0' in FCT by 'Z(5)' (i.e. SQRT).

Thus, for Example 3.2,

```
RESULT = -2*Z(5)**5*Z1(1)*ZZ(1)/5
```

Note that the argument of any function name which is directly represented (Z(3),Z(5),Z3, etc.) in RESULT is SER (see SER and RESULT). So the argument of Z(5), in our example, is equal to 'Z0' (since SER = 'Z0'), and which is in fact COS(Z0), as indicated by the values of MSER and NSER.

Replace SQRT(1) and SQRT(0) in RESULT by 1 and 0 respectively.

PART 7. Output variables and arrays and go to part 1.

Thus, for Example 3.2,

```
STATUS  = (33,55,99) (55 is a dummy integer)
M0      = (23,0,99,99)
M1      = (1,1)
M2      = 0
M5      = 0
```

```
RESULT = -2*Z(5)**5*Z1(1)*ZZ(1)/5
MSER   = 2
PARM   = 1
M6     = (99,99)
KEK    = 0
M7     = 0
SER    = Z0
NSER   = Z0
M8     = (27,10,99,99)
```

Figure 3.6. : Output produced by the processor for Example 3.2.

Similarly, for Example 3.4, the set of variables and arrays which are printed at the end of the processor step are as follows:

```
STATUS  = (33,18,1)
M0      = (23,99,99)
M1      = (2,1)
M2      = 0
M5      = 0
RESULT  = 16
MSER    = 0
PARM    = 1
M6      = (4,-5,41,99)
KEK     = 0
M7      = 12
SER     = Z0-1
NSER    = 0
M8      = (99,99)
```

Figure 3.7 . : Output produced by the processor for Example 3.4.

3.5 The Writer

The last job step in the execution of SYMDIP is the writer. As its name implies, the main function of the writer is to translate results produced by the processor into user comprehensible form and print them. It also prints appropriate error messages if a problem could not be solved.

As in our discussion of the analyzer, the functions of the writer can be divided into several logical parts. These are:

PART 1. Read the problem and print it with some necessary statements associated with the problem.

PART 2. (Integration only) Determine whether it is a subproblem.

PART 3. Determine if the problem has been successfully solved; if it was not, find the appropriate error message and print it.

PART 4. Translate the computed result by replacing terms representing function names in the result by their actual function names. In view of the differences between the handling of differentiation and integration in SYMDIP we will discuss part 4 in two parts.

PART 5. Complete the translation by replacing terms representing the variable and all the alphabetic constants in RESULT by the actual letters or expressions they represent.

In our detailed description of these parts we will continue to use Examples 3.1 and 3.2 which were first introduced in Section 3.2. The results for these problems as computed by the processor are given in Figure 3.4 and Figure 3.5. The following discussion will show how these internal results are translated by the final job step of the

package. Before we proceed to that description, we first explain what we mean by "concatenation".

Definition 3.2. Concatenation is an operation for combining two or more strings to form a new string. It is indicated by specifying two or more string-valued operands separated from each other by at least one blank. (In SNOBOL4 both numeric and non-numeric quantities can be used as strings). Thus, the statement

```
COMBINE = 'CEN' 'TURY' '30'
```

yields

```
COMBINE = 'CENTURY 30'.
```

We also introduce the meaning of some variables, as follows:

MO	Same as described in Section 3.3 and 3.4.
TERM	A character string which contains a term of FUNCTION.
STATUS	Same as described in Section 3.3 and 3.4.
RESULT	A character string equal to RESULT described in Section 3.3 and 3.4.
PROBLEM	The original problem stated by the user, except that the 'WITH RESPECT TO "variable"'part has been removed (by the analyzer).
TERMNO	If the original function has been separated into several terms, TERMNO is used to indicate the current term which is processed.
COUNT	An integer variable which is initialized to 0 at the start of the SNOBOL4 program.

FUNCTION A character string equal to FCT(see Section 3.3).

LETTERS The character string 'ABC...Z'.

ERROR The array of error messages.

TEXT1 The array of elementary functions.

TEXT2 An array used for substitution of Z(I) by a function name which is directly represented.

TEXT3 The integration table containing the terms associated with the integrals of special functions.

TEXT5 An array used for substitution of Z(I) by a function name which is indirectly represented.

TECH The array of derivatives of elementary functions.

M An array consisting of 12 elements. In the SNOBOL4 language, the elements of M are denoted by M<1> M<2>, M<3>, etc. For purposes of exposition, we shall denote them by M(1), M(2), M(3) in the main body of the thesis.

M(1) A character string equal to M1.

M(2) A character string for differentiation, M(2) is equal to MA. For integration M(2) is equal to M2.

M(3) A character string. For differentiation, M(3) is equal to MB. For integration M(3) is equal to M5.

M(4) A character string equal to RESULT.

M(5) For differentiation, M(5) is a character string equal to M3. For integration, M(5) is a character string equal to MSER.

M2 An integer array which contains all the entries of M(2).

M3 An algebraic array which contains all the entries of M(3).

PART 1. Read STATUS, MO. If reading fails, a system error has occurred, set an error flag and go to step (g) of part 5. If STATUS(1) = 999, all the problems and their solutions are printed, go to step (h) of part 5.

Thus, for Example 3.1,

STATUS = (33,0,1)

MO = (23,0,99,99)

and for Example 3.2,

STATUS = (33,0,2)

MO = (23,3,99,99)

(a) Read PROBLEM from the "input file" and add 1 to COUNT. Print the character string 'PROBLEM' followed by COUNT. Obtain the "variable" of differentiation and integration from the character string 'ABC...Z' using the first entry of MO.

(b) Print PROBLEM.

(c) Print the string (formed by concatenation) 'WITH RESPECT TO' "variable".

Thus, at the end of part 1, the lines printed for Example 3.1 are:

```
PROBLEM 1
DIFFERENTIATE EXP(4+A*X)+SIN(1-COS(X))*10**(X-2)**2
WITH RESPECT TO X
```

and the lines printed for Example 3.2 are:

```
PROBLEM 2
INTEGRATE SQRT(COS(X))*SIN(X)*COS(X)*D*LN(10)-SIN(X)
WITH RESPECT TO X
```

PART 2. Is the problem a subproblem , i.e. is STATUS(3) $\neq$ 1 ? If no, go to step (g) of part 2. If yes, is this the first subproblem , i.e. is STATUS(3) > 1 ? If no go to step (c) of part 2. Let XX denote the value of STATUS(3). XX now contains the integer which is equal to the number of terms in the user's original function.

Set TERMNO = 1.

(a) Print 'THE FUNCTION YOU HAVE PROVIDED HAS BEEN SEPARATED INTO' XX 'TERMS' .

(b) Print 'WHICH HAVE BEEN INTEGRATED SEPARATELY'.

(c) Read TERM.

(d) Print 'TERM' TERMNO.

(e) PRINT TERM. Add 1 to TERMNO.

(f) Set MESSAGE = 'THE INTEGRAL IS :' and go to part3.

(g) Set MESSAGE = 'THE RESULT IS :'.

Thus at the end of part 2, no new lines are printed for Example 3.1 but the value of MESSAGE is

'THE RESULT IS :' .

For Example 3.2 the lines printed up to this part are :

```
PROBLEM 2
INTEGRATE SQRT(COS(X))*SIN(X)*COS(X)*D*LN(10)-SIN(X)
WITH RESPECT TO X
THE FUNCTION YOU HAVE PROVIDED HAS BEEN SEPARATED INTO
2 TERMS WHICH HAVE BEEN INTEGRATED SEPARATELY
```

and the value of MESSAGE is

'THE INTEGRAL IS :' .

PART 3. Is the problem successfully solved, that is, STATUS(1) = 33? If yes, read M1 and go to part 4. If no, get the error number which is contained in STATUS(2). If this number is greater than 16, it is a run-time error so go to (d). Otherwise, it is an input error do:

(a) Print '*** INPUT ERROR *** ATTEMPT AT PROBLEM SOLUTION CANCELLED *** '.

(b) Print '***' error message '***' , where error message is obtained from the array of messages, ERROR, using the error number as an index into ERROR.

(c) go to part 1.

(d) print '*** RUN-TIME ERROR *** INTEGRATION ON THE FUNCTION TERMINATED ***'

(e) Print '*** THE FUNCTION IS BEYOND THE PACKAGE CAPABILITIES ***' .

(f) Print '***' error message '***' , where error message is obtained as in step (b).

(g) go to part 1.

PART 4. (Differentiation) Do this part when the second entry of M1 is 0.

Read M(2),M(3),M(4),M(5).

If reading fails, go to step (g) of part 5.

Thus, for Example 3.1,

```
M(2) = '(26,1,100,2,99,99)'
M(3) = '(4+Z0*Z1(1),1-Z(4),Z0-2,Z0,99,99)'
M(4) = 'Z(1)+Z0*Z1(1)*ZZ(1)+2*Z(2)*Z(3)*
       ZZ(3)-Z(3)***2*ZZ(2)*ZZ(4)'
```

M(5) = '(10,99,99)'

Assign RESULT = M(4) and SAVE = M(5).

Thus,

RESULT = 'Z(1)+Z0*Z1(1)*ZZ(1)+2*Z(2)*Z(3)*
ZZ(3)-Z(3)**2*ZZ(2)*ZZ(4)'.

(a) Scan M(2) from left to right. For each complete string of integer encountered place it in the current available entry of the array M2.

Process M(3) as above , but all individual strings are placed in array M3.

Thus, for Example 3.1,

M2(1) = '26'

M2(1) = '1'

M2(3) = '100'

M2(4) = '2'

and

M3(1) = '4+Z0*Z1(1)'

M3(2) = '1-Z(4)'

M3(3) = 'Z0-2'

M3(4) = 'Z0'

(all other entries of M2 and M3 remain 0 as initialized).

Set LK = 1.

(b) If M2(LK) = 0 go to step (c).

Otherwise do the following:

(1) If M2(LK) = 100

Remove the left most entry of M5 and assign it to C0.

Replace 'ZZ(LK)' in RESULT by the string (formed by concatenation) CO '**(' M3(LK) ')*LN(' CO ')'

(2) If M2(LK) ≠ 100

Replace 'ZZ(LK)' in RESULT by the function name obtained by using the value of M2(LK) as an index into the array TECH. Replace '#' by '(' M3(LK) ')' In RESULT ('#' is used as an intermediate text throughout TECH).

Add 1 to LK. If LK is less than 13 go to step (b).

Thus, for Example 3.1,

```
RESULT = 'Z(1)+Z0*Z1(1)*EXP(4+Z0*Z1(1)) +
          2*Z(2)*Z(3)*10**(Z0-2)*LN(10)-
          Z(3)**2*COS(1-Z(4))*(-SIN(Z0))'
```

(c) Set LK = 1.

(d) If M2(LK) = 0 go to step (e).

Otherwise do the following:

(1) If M2(LK) = 100

Remove the left most entry of SAVE and assign it to CO.

Replace 'Z(LK)' in RESULT by the string

CO '**(' M3(LK) ')'.

(2) If M2(LK) ≠ 100

Replace 'Z(LK)' in RESULT by TEXT1(M2(LK)) '(' M3(LK) ')'

Add 1 to LK. If LK is less than 13 go to step (d).

Thus, for Example 3.1,

```
RESULT = 'EXP(4+Z0*Z1(1))+Z0*Z1(1)*EXP
          (4+Z0*Z1(1))+2*SIN(1-COS(Z0))*
```

```
10**(Z0-2)*10**(Z0-2)*LN(10)-
10**(Z0-2)*COS(1-COS(Z0))*(-SIN(Z0))'
```

(e) go to step (b) of part 5.

Before we discuss the integration portion of part 4, we introduce several new variables which will be used. The variables used in the definitions refer to variables introduced in Section 3.4.

M(6) A character string equal to PARM

M(7) A character string equal to M6

M(8) A character string equal to KEG

M(9) A character string equal to M7

M(10) A character string equal to SER

M(11) A character string equal to NSER

M(12) A character string equal to M8

PART 4. (Integration) Do this part when the second entry of M1 is 1.

Read $M(2),\ldots, M(12)$ If reading fails go to step (g) of part 5.

Thus, for Example 3.2, we have:

```
M(2)  = 0
M(3)  = 0
M(4)  = -2*Z(5)**5*Z1(1)*ZZ(1)/5
M(5)  = 2
M(6)  = 1
M(7)  = (99,99)
M(8)  = 0
M(9)  = 0
```

M(10) = Z0

M(11) = Z0

M(12) = (27,10,99,99)

Assign RESULT = M(4)

Thus,

RESULT = -2*Z(5)**5*Z1(1)*ZZ(1)/5 .

Process M(2) and M(3) as in step (a) of part 4 (Differentiation).

Thus, for Example 3.2,

M2 = 0

M3 = 0

(a) If M(9) is less than 1 go to (b).

Otherwise do the following:

(1) If M(9) < 3000 (i.e. FUNCTION is a special function)

Get the coefficients of the eventual integral from M(7). Attach the coefficients to the corresponding "terms" of the eventual integral of FUNCTION ("terms" are taken from the array TEXT3 using M(9) as a key)

Go to part 5 .

(2) If M(9) > 2999

Get the last term, LTERM, of the integral from TEXT3 (the coefficient of this term is PARM).

Attach to RESULT this last term along with its coefficient PARM. So, we set

RESULT = RESULT '+' M(6) '*' LTERM

(b) If M(9) > 0 and M(9) < 3000, the integral of the function has been obtained, go to part 5. Otherwise do the following:

(1) Set SAVE = 1

(2) If the left most entry of M(12) is '99' go to (3).

Otherwise

Remove the left two entries of M(12) and place them in C0 and C1 respectively.

Replace 'ZZ(SAVE)' by the string

TEXT1 (C0) '(' C1 ')'

(note that C1 is the integer argument of the function name contained in TEXT1(C0))

Add 1 to SAVE and go to step (2)

Thus, for Example 3.2,

RESULT = - 2*Z(5)**5*Z1(1)*LN(10)/5

(3) Replace 'Z2' by 'LN(10)' throughout RESULT.

Replace 'Z(6)' by the string

'(' M(10) ')'

throughout RESULT.

(4) If there is a 'Z(N)' in RESULT (N is a positive integer) do the following:

(I) If N < 6 do :

If N < 3 and M(8) = 1 set COND = 'H'

otherwise set COND = '' (empty string).

Replace 'Z(N)' by the string

TEXT2(N) COND '(' M(10) ')'

throughout RESULT.

Go to step (4).

(II) Otherwise

Set N = N - 6.

Replace 'Z(N)' by the string

TEXT1(M2(N)) '(' M3(N) ')'

throughout RESULT.

Go to step (4).

Thus, for Example 3.2,

RESULT = -2*SQRT(Z0)**5*Z1(1)*LN(10)/5

Part 5. Replace 'Z3' by the string 'TAN(' M(10) ')' throughout RESULT.

(a) If M(5) = 0 go to (b),otherwise

(1) Replace 'Z0' by '?' throughout M(11).

(2) Get the function name to which the integer M(5) corresponds from the string TEXT5 by using M(5) as a key. Replace all 'Z0' in RESULT by M(11).

(3) Replace all '?' in RESULT by 'Z0'.

Thus, for Example 3.2,

RESULT = -2*SQRT(COS(Z0))**5*Z1(1)*LN(10)/5

(b) Let IN be the first entry of M0. Get the character which is represented by 'Z0' from the string LETTERS by using IN as a key. Call this character CHAR. Replace 'Z0' by CHAR throughout RESULT.

Thus, for Example 3.1,

```
RESULT = 'EXP(4+X*Z1(1))+X*Z1(1)*EXP(4+X*Z1(1)
         +2*SIN(1-COS(X))*10**(X-2)*
         10**(X-2)*LN(10)-10**(X-2)**2
         *COS(1-COS(X))*(-SIN(X))'
```

and for Example 3.2,

RESULT = '-2*SQRT(COS(X)**5*Z1(1)*LN(10)/5 '

(c) If there is a 'Z1(N)' in RESULT, where N is an integer do: Get the character which is represented by 'Z1(N)' from the string LETTERS by using the $(N+1)^{th}$ entry of M0 as a key. Call this character CHAR. Replace all 'Z1(N)' by CHAR. Go to (c).

(d) If it is an integration problem, set RESULT = RESULT '+ CONSTANT'.

Thus, for Example 3.1,

```
RESULT = 'EXP(4+X*A)+X*A*EXP(4+X*A)
         +2*SIN(1-COS(X))*10**(X-2)*
         10**(X-2)*LN(10)-10**(X-2)**2
         *COS(1-COS(X))*(-SIN(X))'
```

and for Example 3.2,

RESULT ='-2*SQRT(COS(Z0))**5*D*LN(10)/5 + CONSTANT'

(e) Print MESSAGE and RESULT.

(f) Go to part 1.

(g) A system error has occurred do:

(1) Print '***THE RUN WAS ABNORMALLY TERMINATED***'.

(2) Print '***THE SYSTEM ERROR OCCURRED WHEN THE ABOVE PROBLEM OR THE ONE AFTER IT WAS BEING COMPUTED***'.

(3) Read error number from ERROR (ALTRAN error message file).

(4) Print '***' error message '***'.

(Where error message is obtained from the array ERROR using the error number as a key.)

(5) Go to part 1.

(h) Print '*****ALL PROBLEMS ATTEMPTED*****' .

3.6 Interfaces Between Job Steps

As we indicated in Chapter 2, our package is composed of 3 distinct job steps. These job steps always execute in sequential order. The first job step is the analyzer, the second job step is the processor, and the final job step is the writer. Each of these job steps executes independently of the others. It follows that the values of variables in a particular job step are completely unknown to the subsequent job steps. Communication between any two job steps can be established, however, by using temporary files (data sets). Once a data set is created by a job step, it can be used by any subsequent job step. When all steps of a job have been completed, the data sets are automatically deleted from the storage device. In SYMDIP we have made use of several data sets, for transmission of information between different job steps of our package. These include a data set which contains the problems provided by the user, a data set which contains the ALTRAN comprehensible form for each of these problems, a data set which contains the results computed by the ALTRAN processor, and finally, a data set which contains the ALTRAN system error messages (if any). Thus, there are four temporary data sets used for transferring of data during the execution of the package. A general description of each of these data sets is given in Figure 3.8.

Data Set Name	Creator	Contents	Users
INPUT	Analyzer	User's Problems	Writer
CODE	Analyzer	ALTRAN Comprehensible Forms	Processor
RESULT	Processor	Derivatives or Integrals	Writer
ERROR	Processor	Error Messages	Writer

Figure 3.8: Temporary data sets used by SYMDIP

The data set INPUT is used to contain the set of original problems provided by the user. When a problem is processed by the analyzer, it is first written as a single line record on this data set (see Section 3.2). INPUT is used by the writer when it prints the individual problems and their solutions (See Section 3.5, step (a) of part 1).

The data set CODE contains the ALTRAN comprehensible forms translated from the user problems by the analyzer. Once a problem has been translated, the analyzer writes the set of variables and arrays it produces on this data set (see Section 3.2, step(f)of part 3). The actual formats of the records on CODE can be seen in Figures 3.1, 3.2 and 3.3. The order of the variables and arrays on this data set is exactly the same as shown in those figures. Each of these variables and arrays is printed in an individual record of the data set. The processor reads the ALTRAN comprehensible form of a problem from CODE and manipulates it to produce the solution of that problem.

The data set RESULT, as its name implies, contains the computed results of the problems attempted by the processor. The records printed on this data set are shown on Figure 3.4 - Figure 3.7 in Sections 3.3 and 3.4. RESULT is used by the writer which translates the results

into user comprehensible form.

The data set ERROR contains any system error messages generated by the ALTRAN system during the processor step. If an ALTRAN system error has occurred (see part 1, Section 3.5), the writer will read ERROR. The error number written on this data set is searched (see step (g) of part 5 of Section 3.5). This number will be converted into a user comprehensible error message which is then printed along with other error message statements.

The JCL for creating these files is given as part of Appendix J. In the normal operation of the program by a calculus student, the simplified JCL of Appendix I would be used. As can be seen by comparing Appendix I and J, the student would not need to be aware of any of this file detail or the various job steps which were being executed.

CHAPTER FOUR

OVERLAY STRUCTURE

4.1 Introduction

4.2 Multi-Region Overlays and the VS-1 Operating System

4.3 Constructing an Overlay Structure

4.4 Test Results Contrasting Various Modes of Operation

4.1. Introduction.

In Chapter 1, we noted that the ALTRAN system requires a relatively large amount of storage for execution of even a trivial program. Since the major portion of our package was written in the ALTRAN language, this chapter discusses how the main storage management facility of the operating system can be used to reduce the storage requirements during execution of a computer program. In this section we explain why it is desirable to reduce the main memory requirements of our package. We also give an overview of two memory management techniques, overlay and virtual storage, which are used to reduce main memory requirements. In Section 2, we describe the multi-region overlay concept available in the VS-1 operating system. Some examples are given to demonstrate specific points. In Section 3, we discuss the implementation of overlays within our package. Finally, in the last section we compare the various overlay structures we have designed to be used with the package. The set of elementary differentiation and integration problems that have been chosen for use in this comparison is listed in Appendix G.

It was obvious from the very beginning of this project that the usefulness of the final package would depend in part on how long the user was required to wait for his results. Tests of the completed package indicate that it can solve a typical first year calculus problem in less than 1 minute of CPU time. Because of various system overheads it is more efficient to solve several problems in one run and when this is done the average solution time is usually reduced to about 20 seconds per problem.

Clearly the required CPU time is sufficiently small to provide problem solution in a reasonable period of time.

Unfortunately, CPU time requirements represent only one factor in determining how long it will take for the user to obtain his results. The second factor is program size. In the case of SYMDIP, this is determined by the processor (ALTRAN) step which requires 354K bytes. Thus, SYMDIP must be run in a class-C partition on the current University of Victoria computer system. Figure 4.1 gives turn-around statistics collected by the University of Victoria Computer Center for class-C jobs for several months during the 1975-76 term.

Turn-around time (min)	Number of jobs completing processing in less than specified turn-around time.		
	OCT 1975	FEB 1976	MARCH 1976
0 - 30	106	180	252
30 - 60	10	33	35
60 - 90	1	6	7
90 - 120	0	2	1
120- 150	1	1	0
150- 180	0	1	0
> 180	0	1	1
Total	118	224	296
Average turn-around	5	9	10

Figure 4.1.: Turn-around statistics for class-C jobs of less than 5 min. CPU time.

The turn-around times given in Figure 4.1 represent the actual elapsed time that a program was active in the computer. Assuming that the physical handling of the input deck and printed output did not take longer than 30 minutes,a casual look at Figure 4.1 suggests that quite reasonable turn-around for SYMDIP could be expected. Taking a more careful look at the data in Figure 4.1, however,we note the following. The total number of class-C jobs which have been run is relatively small as compared with the average total of 7,000 OS jobs which are run by the computer center each month. A small change in the number of class-C jobs which were run (relative to the total number of OS jobs run) has had a significant impact on the turn-around times for class-C.

Given these facts, it is reasonable to ask what if SYMDIP were to be used by even a small group of calculus students? It is likely that the total number of class-C jobs would at least double the March figure of Figure 4.1. It is also very likely that the average turn-around would increase to at least 20 minutes and more likely the figure would be 30 to 60 minutes. This would occur because there is currently only one class-C partition available on the University of Victoria computer system. Because of the limited amount of main memory available it is unlikely that a second class-C partition could be created. Thus, we must look for some other way to obtain reasonable turn-around for SYMDIP if it is to be used.

Figure 4.2 shows turn-around statistics for class-B programs for the same time periods as Figure 4.1. A class-B program is one which occupies

no more than 256K bytes of main memory. It is clear from Figure 4.2 that significant changes in the numbers of class-B programs has little effect on turn-around time. This happens because there are usually several class-B partitions active in the computer at the same time.

Turn-around time (min.)	Number of jobs completing processing in less than specified turn-around time.		
	OCT 1975	FEB 1976	MARCH 1976
0 - 30	1357	1449	2417
30 - 60	110	161	198
60 - 90	32	48	54
90 -120	6	15	17
120-150	10	3	10
150-180	2	1	1
> 180	13	11	11
Total	1530	1688	2708
Average turn-around	8	6	6

Figure 4.2. : Turn-around statistics for class-B jobs of less than 5 min. CPU time.

It would appear that adding 300 more class-B jobs would not change the turn-around statistics significantly. Thus, if some way could be found to run SYMDIP in a class-B partition, it would appear to be worth trying to implement the package in that way.

Two methods have been devised for reducing the main memory needs of a computer. The first of these was the overlay technique which allowed the programmer to incorporate storage allocation procedures into his program. Before the availability of this technique,programs larger than the

maximun allowable size could not be executed. The overlay facility allowed a programmer to run his oversized computer program by breaking it up into several independent parts which shared the main memory during execution. This was done by having part of the program(one segment) stored on an auxiliary memory device while the other part was in main memory. We note one major feature associated with the use of an overlay structure, namely that the programmer decides(has complete control over) which parts of his program are loaded into main memory,when they are loaded, and where they are placed.

In the past decade, both the processing speeds and memory capacity of the largest computer systems have increased significantly. Consequently, the problem today is no longer one of finding space for most programs but instead it has become one of having enough active programs in the computer at one time to keep the machine busy. One approach to solving this problem is the virtual storage operating system.

In a virtual storage operating system, the storage space is divided into fixed size blocks, called pages. Programs are loaded by the operating system onto pages in the virtual storage. The size of the virtual storage is physically limited by the amount of auxiliary memory available and by the performance characteristics of the computer's I/O hardware.

When a program becomes active, only those pages that are needed are loaded into the main memory. The remaining pages continue to be stored on auxiliary storage. During the execution of the program, pages from auxiliary storage are loaded into the main storage when they are referenced. At the same time, pages in main storage will be released and their locations will then be allocated to the required pages. When this occurs, "overlay of pages" is said to have occurred. Pages that are released from main

storage are returned to the virtual storage, if necessary.

There are several advantages to a virtual storage system. Because overlay of pages is done by the operating system, programmers are able to execute a program which is larger than the main storage size of their system without any additional effort on their part. That is, the operating system decides what should be loaded, when it should be loaded and where it should be loaded. Since pages are loaded into main storage only as required, storage space is used more efficiently and system throughput may be increased.

In a multi-programming environment, virtual storage is divided into partitions. Before a program can be executed, it must first be loaded into one of these partitions. Consequently, programs larger than the maximun size of the partitions cannot be executed. In this case, the overlay facility should be considered.

Ordinarily, when a load module produced by the linkage editor is executed, all of the control sections(procedures and subprocedures) in the module remain in virtual storage(more precisely, in a partition of virtual storage) throughout execution. When storage space is not at a premium, this is the most efficient way to execute a program. However, if a program is beyond the limits of the virtual storage(or its partition), the programmer should consider using the overlay facilities of the linkage editor. When the linkage editor overlay facility is requested the load module is structured so that, at execution time, certain procedures are loaded into virtual storage and then into main storage only when referenced. The way in which an overlay module is structured depends on the relationships among the various procedures within the module. Two procedures

that do not reference each other can overlay each other. Such procedures are independent. They can be assigned the same load addresses.

Procedures may be collected into groups, called segments. A segment is the smallest functional unit (containing one or more procedures) that can be loaded into virtual storage during execution. Procedures that are required all of the time are placed in a special segment called the root segment. This segment remains in virtual storage throughout execution of an overlay program. The way in which segments are loaded into virtual storage during execution is affected by the user providing suitable linkage editor control statements. The user chooses the overlayable portions of the program, and the system arranges to load the required portions when needed during execution of the program. Once this is done, an overlay structure is said to have been created.

An overlay structure is workable only when the overlaying segments can in fact share the storage space. That is, any two procedures in different overlaying segments do not reference each other. Put another way, the procedures must be independent. If this assumption is violated, the overlay structure is invalid. The overlay rule can be stated briefly as follows :

> Rule : No two procedures or subprocedures in different overlaying segments reference each other, either directly or indirectly.

The overlay facility allows the virtual storage space to be used efficiently. However, because independent parts of a program are loaded into virtual storage dynamically, overhead will certainly occur, and

therefore must be handled with care. If an overlay structure is not constructed appropriately, the advantage of the overlay structure will be completely lost. In general, overlays should be used only where the size of the program is beyond the limits posed by the user or by the system.

In the case of SYMDIP, an overlay structure is developed because we expect better user performance if we run in a class-B rather than a class-C partition.

Because SYMDIP is to be used on the University of Victoria IBM 370/145 computer with the VS-1 operating system, all of our discussion in the rest of this chapter will refer to this system, unless otherwise stated.

4.2. Multi-Region overlays and the VS-1 Operating System.

The main ideas of overlay in the VS-1 operating system can be summarized as follows:

(a) There are two possible levels of overlay in VS-1.

(b) The operating system performs overlay of main storage space by paging.

(c) The programmer requests overlay of virtual storage space by constructing an overlay structure.

Since overlay of main storage space is completely in the control of the operating system, the following discussion relates only to the overlay of virtual storage space which is controllable by the programmer. We start by describing an overlay structure.

An overlay structure can be visualized as a tree, called an overlay

tree. The root node of the overlay tree is the root segment (see Section 4.1). One or more branches may extend from the root node to other nodes. Each of these nodes represents a segment of the program, and each of these branches represents references to any procedures in that segment. Similarly, from any of these nodes could originate one or more branches running down to other nodes, and so on. For convenience, we call the originating node of a particular branch the father of the node to which the branch goes. The latter is called the son of the former. Thus we can classify all the nodes of the overlay tree into levels. The son of a node is assigned a level number which is one more than that of its father. For convenience, the root of the tree always has level number equal to 0.

Each procedure will now be in one and only one node of the tree and the number of levels in the tree is determined by the way the program is divided into segments.

During execution, if a particular segment is to be executed, any segments between it and the root segment must also be in storage. This set of segments is called a path. The length of the longest path of the overlay tree determines the amount of virtual storage required by the program.

A program with an overlay structure is usually called an overlay program. To design an overlay structure, the programmer should select the procedure that will receive control at the beginning of execution, plus any other procedures that should always remain in main storage. These procedures form the root segment. The rest of the structure is developed by determining the dependencies of the remaining procedures

and how they can use the same virtual storage locations at different times during execution.

In order to illustrate how an overlay structure can be established, assume we are running a program which is composed of a main procedure and 7 subprocedures. Further assume that references between these procedures are given as follows.

Example 4.1.

Main procedure calls SUB1, SUB3, SUB5.

SUB1 calls SUB2 and SUB4.

SUB5 calls SUB6 and SUB7.

The main procedure must be in the root segment. The root has 3 sons, namely SUB1, SUB3, SUB5. SUB1 has 2 sons, namely SUB2 and SUB4. SUB5 has 2 sons, namely SUB6 and SUB7. The depth of the tree is 2. Since SUB2, SUB4, SUB6 and SUB7 are leaves (a leaf is a node on the tree without a son), they form level 2. SUB1, SUB3 and SUB5 are at depth 1, they form level 1. By definition, the main procedure forms level 0. The overlay structure can be constructed as shown in Figure 4.3.

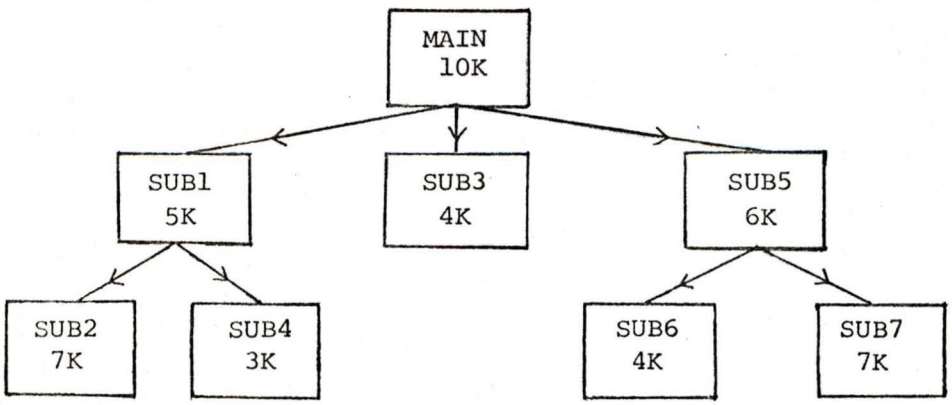

Figure 4.3.: An acceptable overlay structure for Example 4.1.

In Figure 4.3, SUB1, SUB3 and SUB5 will occupy the same storage locations but only one of them is loaded into storage at any given time. This is also true for the pair SUB2 and SUB4, and for the pair SUB6 and SUB7.

There are 8 possible paths on the tree. The longest path is MAIN-SUB5-SUB7 and the virtual storage required for execution of the program with this overlay structure is therefore 23K bytes. If the program were not in overlay, the virtual storage required would be 46K bytes.

The reader may notice that Figure 4.3 is, in fact, directly derived from the relationships between the subprocedures of the program. The arrows shown in the figure indicate both subprocedure calls and executable paths. Since the procedure linkages for Example 4.1 are not complicated, the overlay structure shown in Figure 4.3 is constructed very easily. Unfortunately, it is frequently the case that relationships between various routines in a program are so complicated that constructing an overlay structure becomes very difficult. Figure 4.3 shows only one of the possible overlay structures for Example 4.1. Another overlay structure can be obtained by placing the main procedure (MAIN), SUB1 and SUB5 in the root segment, and considering each of the remaining subprocedures as a son of this root segment. The resulting overlay structure is shown in Figure 4.4.

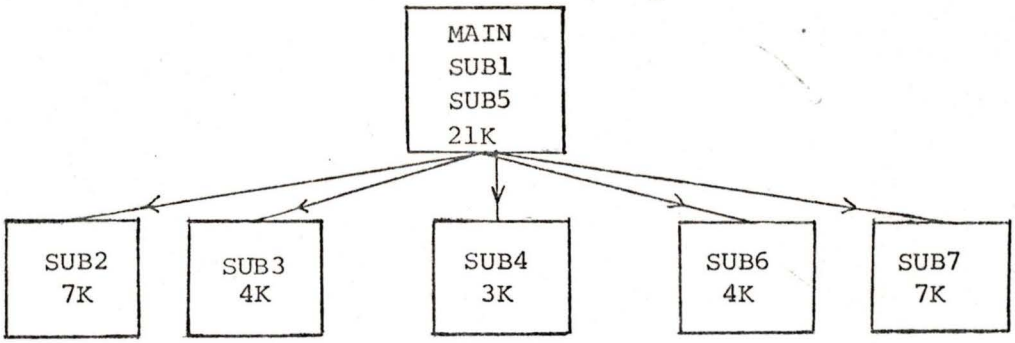

Figure 4.4.: An acceptable overlay structure for Example 4.1.

The overlay structure shown in Figure 4.4 now requires 28K bytes of storage space for execution of the program.

In the VS-1 operating system, an overlay structure may consist of several independent parts, called regions. A region is a contiguous area of virtual storage within which segments can be loaded independently of paths in other regions. An overlay structure with more than one region is called a multi-region overlay structure. Currently, a maximum of 4 regions is allowed.

A multi-region overlay structure is appropriate when several paths need the same procedure. In a single region overlay structure, such a procedure would normally need to be placed in the root segment so that it can be called from any segment. However, the root segment might now become so large that the benefit of overlay is greatly reduced. This problem may be solved by placing the frequently referenced procedure in a separate region.

To illustrate the construction of a multi-region overlay structure, consider the following example.

Example 4.2.

Procedure calls as in Example 4.1, and in addition,

SUB3 calls SUB2, SUB4, SUB6 and SUB7.

The overlay structure shown in Figure 4.3 is no longer acceptable because none of the subprocedures called by SUB3 are on a path containing SUB3. The overlay structure of Figure 4.4 is also unacceptable for basically the same reason. If SUB3 is placed in the root segment in Figure 4.3, as shown in Figure 4.5, then an acceptable overlay structure is created.

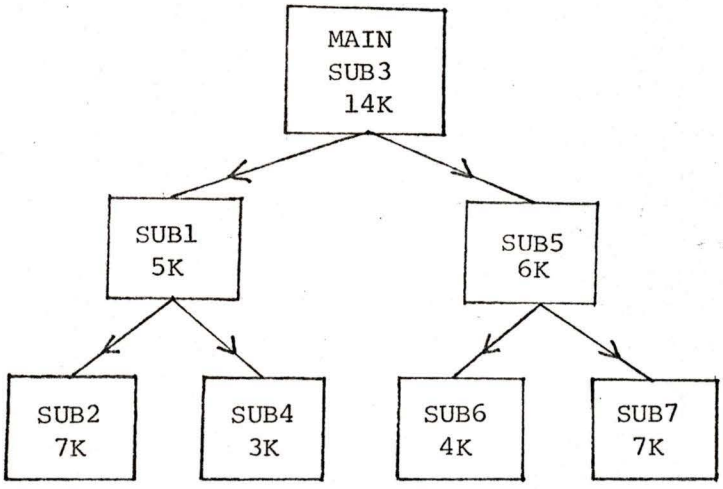

Figure 4.5: An acceptable overlay structure for Example 4.2.

The overlay structure just constructed requires 27K bytes of virtual storage space for execution of the program because this is the longest path on the new overlay tree. A better solution is to place the routines SUB2, SUB4, SUB6 and SUB7 into another region, as shown in Figure 4.6.

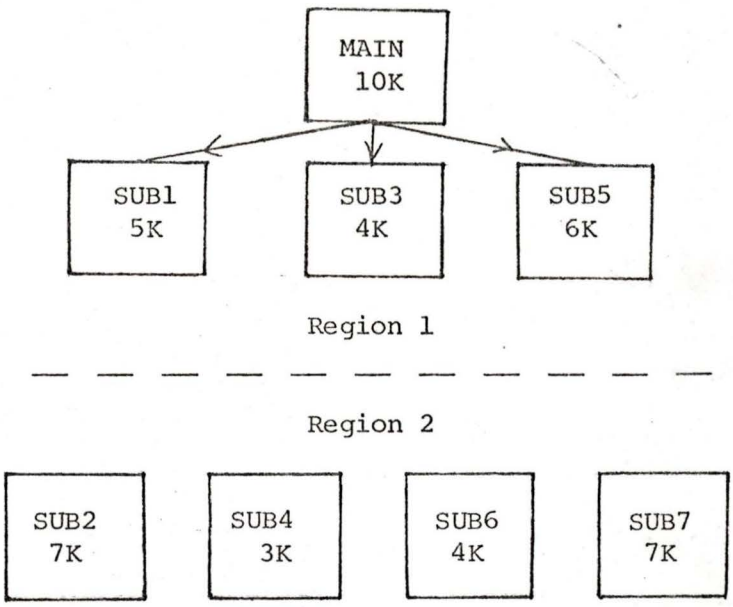

Figure 4.6. : An acceptable multi-region overlay structure for Example 4.2.

With this overlay structure, any procedure in Region 1 can call any procedure in Region 2. In addition, the amount of virtual storage space required for execution of the program is 23K bytes, which is the same as in Example 4.1.

The design of a multi-region overlay structure is generally not easy. Because each region must observe the overlay rule, the segments must be arranged very carefully to avoid a significant increase in overhead. On the other hand, a multi-region overlay structure could increase segment loading efficiency, that is, processing can continue in one region while the next subpath to be executed is being loaded into another region [8].

We now proceed to the discussion of a method to produce a multi-region overlay structure which can greatly reduce the storage requirements of our package.

4.3. Constructing an Overlay Structure.

In this section, we describe how we developed on overlay structure for our package which significantly reduced its storage requirements. This was not a trivial task since the complete package consists of approximately 280 subprocedures and each of these procedures calls (on average) 6 other procedures. Thus, there were approximately 1680 procedure linkages to be dealt with in designing an overlay structure.

The overlay design process contained three steps. The first step was to distribute the subprocedures into several levels. The second step was to determine segments from levels. The third step was to arrange these segments into regions.

Step 1. Create levels.

The set of approximately 250 subprocedures in the ALTRAN package were distributed into 26 levels, as follows :

Level 0. Contained the set of subprocedures each of which did not call any other procedure.

Level 1. Contained the set of subprocedures each of which called only procedures in level 0.

Level 2. Contained the set of subprocedures each of which called only subprocedures in level 0 or 1 with at least one call being in level 1.

Generally, for $K > 2$, level K contained the set of subprocedures each of which called only subprocedures in level 0, 1,...,K - 1 with at least one call being in level K - 1. It is clear that each of the

approximately 250 subprocedures was associated with one and only one level, and subprocedures in the same level obviously never called each other, either directly or indirectly. Because of this property, we could randomly split a level into two or more independent segments.

The program we used to determine the levels is listed in Appendix K.

Step 2. Determine segments.

The most trivial way to create a multi-region overlay structure is to select the levels with largest procedures and split each of them into two or more segments, each of these levels can then form a region. Obviously, the overlay structure thus obtained is by no means an efficient one because independent segments are obtained randomly.

Step 2 was handled by us as follows. The levels were inspected to identify procedures performing functions of the same kind. Note that procedures that perform similiar functions usually link among themselves with only a single link to external procedures. One obvious example is the set of ALTRAN read routines. In this way , five sets of procedures were obtained. Several procedures have been deleted from these sets to obtain necessary independence among some segments(with the aid of levels it was easy).

Step 3. Form regions.

The segments obtained above were used to form regions. The following criteria were applied :

(a) Independent segments were placed in the same region

provided swapping among these segments was not significant (this can be determined with the aid of the levels). Any two segments that were not independent were placed in different regions.

(b) Segments placed in the same region should have almost equal size. Thus, if necessary, one or more procedures should be removed from a segment to form a new segment.

The overlay structure thus obtained is composed of the following regions.

Region 4. Contains 2 segments.

Segment 1 contains ALTRAN error handling routines.

Segment 2 contains general routines for manipulating pointers, blocks, stacks etc.

Region 3. Contains 3 segments.

Each of these segments contains general routines.

Region 2. Contains 4 segments.

Segment 1 contains ALTRAN read routines.

Segment 2 contains ALTRAN write routines.

Segment 3 contains general routines.

Segment 4 contains routines for manipulating specialized functions, such as the procedure NTRM which returns the number of terms in a given expression.

Region 1 is constructed in the following way :

The root segment of region 1 contains the main procedure of SYMDIP and all the necessary procedures in the ALTRAN library which were not included in any of the above regions.

The remaining 24 procedures written by us were placed in segments other than the root segment of region 1.

Construction of the overlay structure in this way has many advantages. For example, the read and write ALTRAN routines are less frequently used in the package than many of the general routines, therefore the swapping activity involved should not be significant. Likewise, the ALTRAN error handling routines of region 4 will be used very infrequently and so swapping activity should be very low here also.

4.4. Test Results Contrasting Various Modes of Operation.

In this section, we present a comparison of several overlay structures we developed to run with SYMDIP. Each of these overlay structures contains a certain number of regions described in the previous section. A set of differentiation and integration problems which were taken from the MATH 130 examination papers for fall and spring of session 1975-1976 were used as test problems. These problems are listed in Appendix G.

The purpose of this comparison was to enable us to understand the relative effectiveness of these overlay structures in reducing the virtual storage requirement, as well as, their influence on the execution time required to solve those problems. For this reason, the package was also run without any overlay structure. The comparison is shown in Figure 4.7.

Test No.	Regions	Execution time	Storage used
1	No overlay	9 min. 49 sec.	354K
2	Region 1	9 min. 50 sec.	312K
3	Region 1,2	10 min. 54 sec.	272K
4	Region 1,2,3	11 min. 12 sec.	270K
5	Region 1,2,3,4	11 min. 47 sec.	256K

Figure 4.7. : Comparison of overlay structures.

Figure 4.7 shows that both region 1 and region 2 gave a relatively large amount of reduction in the virtual storage requirements of SYMDIP. On the other hand, the figure also shows that region 1 gave the smallest amount of overhead while region 2, when it is included in any overlay structure, would give the largest amount of overhaed. One way to evaluate the effectiveness of an overlay structure is to consider the ratio of the increase in CPU time to the reduction in storage space required. Figure 4.7 indicates that region 1 created the smallest amount of overhead for each unit of the storage space it reduced . The corresponding ratio for region 3 is the largest among the 4 regions. Since the tests were performed on a set of 30 problems it is very obvious that a problem can be solved completely within 1 minute of CPU time on the average. Moreover, a problem can also be solved using 256K bytes of virtual storage, that is, a class-B partition at the University of Victoria Computer Center.

In order to understand the functional ability of the package more closely, we have also run the package with a set of 18 harder differentiation and integration problems. Without any overlay structure, the problems were successfully solved with approximately 9 minutes of execution time. The set of these 18 harder problems is listed in Appendix H.

CHAPTER FIVE

CONCLUSIONS

5.1. Accomplishments

5.2. Extensions

5.1. Accomplishments.

Recall, from Chapter 1 that the basic goal of this thesis was the design and implementation of a computer program for symbolic differentiation and integration. It was hoped that the package which was developed could be used as a teaching tool in a typical first year calculus class. Specifically, the package should be able to solve a problem using less than 1 minute of CPU time and less than 256K bytes of main storage space on the university IBM 370/145 computer. This section summarizes how these goals are met by our computer program SYMDIP.

Because SYMDIP sucessfully solved all but one of the 30 problems taken from the recent examination papers for Mathematics 130, we can reasonably assume that most of the problems that can be found in a typical first year calculus text can be solved by SYMDIP. Because nesting of elementary functions is allowed in a function which is to be differentiated or integrated, SYMDIP can also solve a broad range of difficult and complicated problems not usually seen in a calculus text. The 18 problems of Appendix H clearly illustrate this.

If a computer program is to be used as a teaching tool, it must be easy to use and easy to understand. In particular, the user must be able to use the package after only a very brief introduction to it. Chapter 2 dealt with this problem. It is clear that the SYMDIP user is allowed to pose his problem in a natural and simple form. The solutions produced by SYMDIP are also presented in a natural and easily read form. SYMDIP also provides sufficient and meaningful error messages if an error occurs. Finally, the JCL statements that are required to activate the program are also simple. The SYMDIP user

is not required to understand the structure of the program or the programming language in which it is written. Because of these features of SYMDIP, we believe students would be interested in using it as an aid to learning.

The final major concern in using a computer program as a teaching tool is related to the cost. Naturally, the computer program must be inexpensive to use in terms of the CPU time and storage space. The user must also be able to obtain the solution to his problem without long delays. Because the overlay version of SYMDIP is able to run in a class-B partition, the storage constraint imposed on SYMDIP has been met. Since SYMDIP can solve a problem in approximately 20 seconds, the CPU time constraint is also met. Moreover, a SYMDIP user can expect to obtain the solution of his problem relatively quickly because a class-B partition is used.

5.2. Extensions .

The discussion in Section 5.1 indicates that all of the goals we have set at the beginning of this project have been met by SYMDIP. This section discusses some possible extensions that can be made to enlarge the functional capabilities of SYMDIP while continuing to satisfy the basic design goals given above.

As has been indicated in Chapter 2, most rational functions with denominator of degree greater than 2 can not be integrated by SYMDIP. These rational functions can be made integrable by SYMDIP by writting several routines to generate the required partial fraction expansions. Routines of this type could be designed around the built-in ALTRAN

procedure ASOLVE (see Section 1.3). Many rational functions would still not be integrable because only partial fraction expansion with integer coefficients could be generated.

Other features of ALTRAN could be used to expand the SYMDIP capabilities. For example, ALTRAN permits the evaluation of an algebraic expression, that is, variables that appear in an algebraic expression can be replaced by specific numeric values. Thus, it would be possible to expand SYMDIP so that it could compute the numerical derivative at a given point or evaluate a definite integral. This would, however, require considerable programming effort because a function is transformed from one form to another as it is being processed by SYMDIP. Moreover, the transformation can become very complicated when the function involves nesting. If this feature were implemented, it would also be necessary to check denominators of expressions and arguments of all the functions to verify that they are within appropriate ranges.

The last and possibly the most useful expansion of SYMDIP we shall mention is related to the form of solutions it produces. For teaching purposes, it is desirable that not only the solution of the problem is returned to the user, but also the method that is used to compute the solution is provided. Thus techniques such as changes of variables or integration by parts might be indicated. It would also be useful if SYMDIP could provide several problems that could be solved in the same way as the problem that has just been solved. Since the differentiation and integration routines in SYMDIP use a set of predetermined methods once the form of the function is obtained,

the extension of SYMDIP in this direction is also possible.

In Section 1.3, we indicated that because of the limited time available, we would write the major portion of our program using the general purpose algebraic language ALTRAN. It would be interesting to do a critical comparison of our package with at least one other package designed to do symbolic integration. The best comparison package might be SIN (symbolic integrator) which was developed by MOSES [10]. SIN was written using LISP and a little known language named SCHATCHEN. A version of SIN was developed for use with certain IBM 360 operating systems. Implementation of SIN on our system would probably not be too difficult unless it contains features unique to the M.I.T. project MAC system on which it was developed.

REFERENCES

1. AHO,A.V., HOPCROFT, J.E., and ULLMAN, J.D. (1974). The design and analysis of computer algorithms. Reading, Mass. : Addison-Wesley.

2. BARTON, D. and FITCH, J.P. (1972). A review of algebraic manipulative programs and their application. The computer journal, Vol. 15, pp. 362-381.

3. BROWN, W.S. (1973). ALTRAN, User's Manual. Murray Hill. N.J. : Bell Laboratories.

4. GRIES, D. (1971). Compiler construction for digital computers. N.Y. : John Wiley & Sons.

5. GRISWOD, R.E., POAGE, J.F., and POLANSKY, I.P. (1971). The SNOBOL4 programming language. Englewood Cliffs, N.J. : Prentice-Hall.

6. HALL, A.D., JR. (1971). The ALTRAN system for rational function manipulation-A survey. CACM, Vol. 14, pp. 517-521.

7. HALL, A.D. (1971). ALTRAN, Installation and Maintenance Manual. Murray Hill, N.J. : Bell laboratories.

8. IBM. Linkage editor and loader. IBM File No. GC 26-3813-4.

9. McCARTHY, J. (1960). Recursive functions of symbolic expressions and their computation by machine. CACM, Vol. 3, pp. 184-195.

10. MOSES, J. (1971). Symbolic integration: The stormy Decade. CACM, Vol. 14, pp. 548-560.

11. SHERWOOD, G.E.F., and TAYLOR, A.E. (1942). Calculus. Englewood Cliffs, N.J. : Prentice-Hall.

12. TSICHRITZIS, D.C., and BERNSTEIN, P.A. (1974). Operating systems. New York, N.Y. : Academic Press.

13. WEISSMAN, C. (1968). LISP 1.5 primer. Belmont, Cal. : Dickenson Publishing Co.

APPENDICES

```
-UNLIST
*
*
*                          ******************************
*                          *                            *
*                          *          ANALYSER          *
*                          *                            *
*                          ******************************
*
*
*                     STRINGS PATTERNS AND OPTIONS
*
*
*
          INPUT('INPUT',3,80)
          &TRIM = 1
          &ANCHOR = 0
          TECH =   'SINH13COSH14CSCH15SECH16TANH17COTH18'
+                  'SIN1COS2CSC3SEC4TAN5COT6'
+                  'ASINH19ACOSH20ACSCH21ASECH22ATANH23ACOTH24'
+                  'ASIN7ACOS8ACSC9ASEC10ATAN11ACOT12SQRT28'
+                  'TT26LN27VV6AA99BB99CC100EXP26LOG29'
          TEXT6 =  'SIN1COS2CSC3SEC4TAN5COT6EXP26LN27SQRT28'
+                  'SINH13COSH14CSCH15SECH16TANH17COTH18'
          NESTER1 =  'SIN.COS.COS.SIN.CSC.COT.SEC.TAN.TAN.'
+                    'SEC.COT.CSC.SINH.COSH.COSH.'
+                    'SINH.CSCH.COTH.SECH.TANH.TANH.SECH.COTH.CSCH.'
          DIGITS = '0123456789'
          LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
          SPECIAL = '?"=::>_@#%$<.|&,¬'
          OPERATORS = '+-*/'
          CONTENT = SPAN(DIGITS LETTERS OPERATORS '()')
          CONTEXT = SPAN(DIGITS LETTERS OPERATORS '(_)')
          TWO = ANY(LETTERS) ANY(LETTERS)
          THREE = TWO ANY(LETTERS)
          FOUR = THREE ANY(LETTERS)
          FIVE = FOUR ANY(LETTERS)
          STRIKE = (FIVE | FOUR | THREE | TWO)
          OPERATIONS = 'DIFFERENTIATE' | 'DIFF' |
+         'INTEGRATE' | 'INT'
          K1 = ('SIN' | 'COS')
          K2 = ('*' | '/') . C3
          K3 = ('+' | '-') . C1
          K4 = ('+' | '-' | '') . C4
          STYLE1 = K1 . C0 '(LN' BREAK('?') . C6
          STYLE2 = '(' K4 SPAN(DIGITS) . C0 K3
+         ((SPAN(DIGITS) . C2 '*') | '')
          STYLE3 = 'Z(1)' K2 '(1' K3 ((SPAN(DIGITS) . C2
+         '**2*') | '') 'Z(2)**2)?'
          STYLE4 = 'SQRT' ('(1' ('+' | '-') ((SPAN(DIGITS)
+         '**2*') | '')) . KING1 'SIN(' BREAK('?') . KING2
          COUNT1 = 0
          COUNT2 = 0
*
*
*                     STARTING OF PROGRAM
*
*
*
```

```
START   KEEP = INPUT                                          :F(EXIT)
        KEEP OPERATIONS                                       :F(START)
STRT1   COUNT2 = 1
        STACK1 = KEEP
STRT2   KEEP = INPUT                                          :F(STRT3)
        KEEP OPERATIONS                                       :S(INTI1)
        STACK1 = STACK1 KEEP                                  :(STRT2)
STRT3   KEEP =
*
*
*                    INITIALIZATION OF VARIABLES
*
*
INTI1   TYPE = 0
        FLAG = 0
        PART = ARRAY(10,0)
        PROBLEM = ARRAY(10,0)
        PROBLEMNO = 1
        MULTIPLE = 1
        PAIR = ARRAY(2,0)
        COUNT0 = 0
        NUMBER2 = 0
        NUMBER3 = 0
        STORE4 = ''
        STORE6 = 0
        TEMP = 0
        COLLECT =
*
*
*                    SYNTACTIC ANALYSIS
*
*
ANLY1   OUTPUT('OUTPUT',10,'(1X,400A1)')
        INTEGROR = '?'
        STACK1 POS(0) SPAN(' ') =
        STACK1 POS(0) OPERATIONS . C4 ' ' =                   :S(ANLY2)
        OUTPUT = STACK1
ANLY2   STACK2 = STACK1
        STACK2 ('WITH' | 'W.R.T.') . C1 = '|'                 :F(ANLY3)
        STACK2 BREAK('|') . FUNCTION '|' = C1                 :(ANLY4)
ANLY3   FUNCTION = STACK2
ANLY4   FUNCTION SPAN(' ') =                                  :S(ANLY4)
        FUNCTION POS(0) '+' =
ANLY5   STACK2 = IDENT(C4,'DIFF') 'DIFFERENTIATE ' FUNCTION
+                                                             :S(ANLY6)
        STACK2 = IDENT(C4,'INT') 'INTEGRATE ' FUNCTION        :S(ANLY6)
        STACK2 = C4 ' ' FUNCTION
ANLY6   OUTPUT = STACK2
        STACK2 POS(0) ('+' | '*' | '/')                       :S(ERR8)
        IDENT(FUNCTION,'')                                    :S(ERR1)
        FUNCTION ANY(SPECIAL) . C2                            :S(ERR9)
        STACK2 = FUNCTION
        C1 = 0
ANLY7   STACK2 ANY('()') . C0 =                               :F(ANLY8)
        C1 = IDENT(C0,'(') C1 + 1
        C1 = IDENT(C0,')') C1 - 1
        LT(C1,0)                                              :S(ERR6)
```

```
+                                                                    F(ANLY7)
ANLY8   EQ(C1,0)                                                     :F(ERR6)
        STACK2 = FUNCTION
ANLY9   STACK2 SPAN(OPERATORS) . C0 =                                :F(ANLY10)
        LT(SIZE(C0),2)                                               :S(ANLY9)
        IDENT(C0,'**')                                               :F(ERR8)
+                                                                     S(ANLY9)
ANLY10  STACK2 = FUNCTION
        STACK2 ANY(OPERATORS) ')'                                    :S(ERR8)
        STACK2 '(' ('+' | '*' | '/')                                 :S(ERR8)
        STACK2 ANY(OPERATORS) RPOS(0)                                :S(ERR8)
        STACK2 ('(' | ')(' )                                         :S(ERR8)
        FUNCTION = ':' FUNCTION ':'
        STACK1 'WITH' SPAN(' ') 'RESPECT' SPAN(' ') 'TO'
+       SPAN(' ') ANY(LETTERS) . INTEGROR =                          :S(ANLY11)
        STACK1 'W.R.T.' SPAN(' ') ANY(LETTERS) . INTEGROR
+                                                                    :S(ANLY11)
        FUNCTION NOTANY(LETTERS) ANY(LETTERS) . INTEGROR
+       NOTANY(LETTERS)                                              :S(ANLY11)
        INTEGROR = 'X'
ANLY11  FUNCTION (ANY(LETTERS) SPAN(LETTERS)) . C2
+       NOTANY('(')                                                  :S(ERR20)
        FUNCTION ANY(LETTERS) ANY(DIGITS)                            :S(ERR24)
        FUNCTION ANY(DIGITS) ANY(LETTERS)                            :S(ERR24)
        FUNCTION NOTANY(LETTERS) ANY(LETTERS) . C2 '('               :S(ERR25)
        FUNCTION SPAN(DIGITS) '('                                    :S(ERR26)
ANLY12  FUNCTION NOTANY(LETTERS) . C0 '(' SPAN(DIGITS) . C1
+       ')' = C0 C1                                                  :S(ANLY12)
ANLY13  FUNCTION ':' =                                               :S(ANLY13)
        NUMBER2 = DIFFER(C4,'DIFF')
+       DIFFER(C4,'DIFFERENTIATE') 1                                 :F(DIFF1)
*
*
*                   IF THE INTEGRAND HAS MORE THAN ONE TERM
*                   EACH TERM IS STORED IN ARRAY PROBLEM
*                   EACH OF THEM IS TREATED AS AN INDEPENDENT FUNCTION
*
*
        C1 = 0
        C3 = 0
        FUNCTION STRIKE                                              :F(SPEC1)
        COUNT5 = 1
        FUNCTION = FUNCTION '+'
        FUNCTION POS(0) NOTANY('+-') . C2 = '+' C2
TERM1   FUNCTION ANY('()') . C0 = 'ε'                                :F(TERM6)
        EQ(C1,0)                                                     :F(TERM2)
        FUNCTION ('+' | '-' | '=') . C2 'ε' = C2 '%'                 :S(TERM3)
        FUNCTION 'ε' = '<'                                           :(TERM3)
TERM2   IDENT(C0,'(')                                                :F(TERM4)
        FUNCTION 'ε' = '#'
TERM3   C1 = C1 + 1                                                  :(TERM1)
TERM4   C1 = C1 - 1
        EQ(C1,0)                                                     :F(TERM5)
        FUNCTION '%'                                                 :F(TERM4D)
        FUNCTION 'ε' ('+' | '-' | '=')                               :F(TERM4C)
        FUNCTION ('%' BREAK('ε') . C4 'ε') = '¬'
        C4 TWO                                                       :S(TERMO)
```

```
          FUNCTION '¬' = '<' C4 '>'                                  :(TERM1)
TERM0     IDENT(C2,'-')                                              :F(TERM4A)
          C3 = C3 + 1
          FUNCTION '-¬' = '=' C3 '=¬=' C3 '='
TERM4A    C4 '#' = '('                                               :S(TERM4A)
TERM4B    C4 '$' = ')'                                               :S(TERM4B)
          FUNCTION '¬' = C4                                          :(TERM1)
TERM4C    FUNCTION '%' = '<'
TERM4D    FUNCTION '&' = '>'                                         :(TERM1)
TERM5     FUNCTION '&' = '$'                                         :(TERM1)
TERM6     FUNCTION '<' BREAK('>') . C0 '>' = '¬' COUNT5 '>'          :F(TERM7)
          PART<COUNT5> = '(' C0 ')'
          COUNT5 = COUNT5 + 1                                        :(TERM6)
TERM7     EQ(C3,0)                                                   :S(TERM7E)
TERM7A    FUNCTION '=' C3 '=' BREAK('=') . C4 '=' C3 '=' = '|'
          C4 POS(0) ('+' | '-')                                      :S(TERM7B)
          C4 = '+' C4
TERM7B    C4 '+' = '.'                                               :S(TERM7B)
TERM7C    C4 '-' = '+'                                               :S(TERM7C)
TERM7D    C4 '.' = '-'                                               :S(TERM7D)
          FUNCTION '|' = C4
          C3 = C3 - 1                                                :(TERM7)
TERM7E    COUNT5 = 0
TERM8     FUNCTION ('+' | '-') . C2 BREAK('+-') . C1 =               :F(TERM9)
          C1 STRIKE                                                  :S(TERM8A)
          COLLECT = COLLECT C2 C1                                    :(TERM8)
TERM8A    COUNT5 = COUNT5 + 1
          C2 = IDENT(C2,'+')
          PROBLEM<COUNT5> = C2 C1                                    :(TERM8)
TERM9     IDENT(COLLECT,'')                                          :S(TERM9A)
          COUNT5 = COUNT5 + 1
          COLLECT POS(0) '+' =
          PROBLEM<COUNT5> = COLLECT
TERM9A    COUNT2 = GT(COUNT5,1) COUNT5
          MULTIPLE = COUNT5
TERM10    FUNCTION = PROBLEM<PROBLEMNO>
TERM11    FUNCTION '¬' SPAN(DIGITS) . C0 '>' = PART<C0>              :S(TERM11)
TERM12    FUNCTION '$' = ')'                                         :S(TERM12)
TERM13    FUNCTION '#' = '('                                         :S(TERM13)
          TEMP = 0
          FLAG = 0
          TYPE = 0
          COUNT0 = 0
          STORE4 =
          PROBLEM<PROBLEMNO> = FUNCTION
*
*
*                    DECTECT SPECIAL FUNCTIONS AND CONSTRUCTIONS
*
*
SPEC1     FUNCTION SPAN(DIGITS) . C0 '**' ANY(LETTERS) . C1
+         (NOTANY(LETTERS) | '') . C2 = C0 '**(' C1 ')' C2           :S(SPEC1)
          FUNCTION SPAN(DIGITS) . C0 '**(' = 'EXP('                  :F(SPEC3)
          TEMP = 1
          COUNT0 = COUNT0 + 1
          STORE4 = STORE4 '27,' C0 ','
SPEC2     FUNCTION SPAN(DIGITS) . C1 '**(' = 'EXP('                  :F(SPEC3)
```

```
          IDENT(C0,C1)                                                    :F(ERR21)
+                                                                          S(SPEC2)
SPEC3     FUNCTION STRIKE . C0 '(' SPAN(DIGITS) . C1 ')' = '#'
+                                                                         :F(SPEC4)
          TECH C0 SPAN(DIGITS) . C0                                      :F(ERR5)
          COUNT0 = COUNT0 + 1
          STORE4 = STORE4 C0 ',' C1 ','
          FUNCTION '#' = '@(' COUNT0 ')'                                 :(SPEC3)
SPEC4     STORE6 = 0
          FUNCTION = '?' FUNCTION '?'
          FUNCTION STYLE4 = KING1 'VV(' KING2                            :S(ARGM1)
          FUNCTION STYLE1 = '#'                                          :F(INTE1)
          FUNCTION STRIKE                                                :S(SPEC5)
          TYPE = 1000
          C6 ')' =
          FUNCTION '#' = C0 C6                                           :(ARGM1)
SPEC5     FUNCTION '#' = C0 '(LN' C6                                     :(CHCK1)
*
*
*                    FORM IDENTIFICATION OF
*
*                      LN(X+SQRT(X**2+5**2))
*                      LN(X+SQRT(X**2-5**2))
*                      LN(X-SQRT(X**2+5**2))
*                      LN(X-SQRT(X**2-5**2))
*                      LN(X+SQRT(5**2+X**2))
*                      LN(X-SQRT(5**2+X**2))
*                      SQRT(X**2+5**2)
*                      SQRT(X**2-5**2)
*                      SQRT(5**2+X**2)
*                      SQRT(5**2-X**2)
*
*
*
INTE1     FUNCTION '/LOG' = '(1/Z2)/LN'                                  :S(INTE1)
INTE2     FUNCTION 'LOG' = 'Z2*LN'                                       :S(INTE2)
INTE3     FUNCTION '**2' = '_'                                           :S(INTE3)
INTE4     FUNCTION 'SQRT' = '>'                                          :S(INTE4)
          FUNCTION ('LN(' CONTEXT . C4
+         '>(' CONTENT . C5 '_' ('+' | '-') . C2
+         ((SPAN(DIGITS) . C3 '_') | '1' . C3) '))') . C0                :S(INTE5)
          FUNCTION ('LN(' CONTEXT . C4 '>(' ('1' . C3 | (
+         SPAN(DIGITS) . C3 '_')) '+' CONTENT . C5 '_))') . C0
+                                                                         :F(INTE6)
          C2 = '+'
INTE5     C4 LEN(1) . C1 RPOS(0) =                                       :F(ERR2)
          C6 = 0
          C7 = 0
          TYPE = IDENT(C1,'+') IDENT(C2,'+') -1
          TYPE = IDENT(C1,'+') IDENT(C2,'-') -2
          TYPE = IDENT(C1,'-') IDENT(C2,'+') -3
          TYPE = IDENT(C1,'-') IDENT(C2,'-') -4                          :(FINE1)
INTE6     FUNCTION '>(' (('1' . C0 | (SPAN(DIGITS) . C0 '_'))
+         ('+' | '-') . C1 CONTENT . C4 '_') . C2 ')'                    :F(INTE7)
          TYPE = IDENT(C1,'+') -3
          TYPE = IDENT(C1,'-') -1                                        :(FINE3)
```

```
INTE7    FUNCTION '>(' (CONTENT . C4 '_' ('+' | '-') . C1 ('1' . C0
+        | (SPAN(DIGITS) . C0 '_'))) . C2 ')'                    :F(CHCK1)
         TYPE = IDENT(C1,'+') -3
         TYPE = IDENT(C1,'-') -2                                 :(FINE3)
*
*
*                    THE FOLLOWING CHECK ANY NESTING
*                    IN THE ABOVE FORMS
*
*
FINE1    C4 '*' = '**2'                                          :S(FINE1)
         C4 POS(0) STRIKE . C6
         C5 POS(0) STRIKE . C7 =
         IDENT(C6,C7)                                            :F(ERR4)
         GT(SIZE(C6),1)                                          :F(FINE2)
         C4 C6 '(' = C6 '(' C3 ',' C5 ','
         FUNCTION C0 = 'LN(' C4 ')'                              :(CHCK1)
FINE2    FUNCTION C0 = 'LN(' C3 ',' C5 ',' C4 ')'                :(CHCK1)
FINE3    C4 POS(0) STRIKE . C6                                   :F(FINE4)
         C4 C6 '(' = C6 '(Z0**2' C1 C0 '**2' ',' C0 ','
         FUNCTION '>(' C2 ')' = 'SQRT(' C4 ')'                   :(CHCK1)
FINE4    FUNCTION '>(' C2 ')' = 'SQRT(' C2 ',' C0 ',' C4 ')'
*
*
*                    DETECT NESTING IN AN INTEGRAND
*
*
CHCK1    FUNCTION '_' = '**2'                                    :S(CHCK1)
CHCK2    FUNCTION '>' = 'SQRT'                                   :S(CHCK2)
         N = 0
         FUNCTION STRIKE . C0 '(' STRIKE . C1 '('
+        = C0 '(<'                                               :F(CHCK6)
CHCK3    FUNCTION STRIKE . C3 '(' STRIKE . C4 '('
+        = C3 '(<'                                               :F(CHCK4)
         IDENT(C1,C4)                                            :F(ERR15)
+                                                                 S(CHCK3)
CHCK4    C1 ('LN' | 'SQRT' | 'EXP')                              :S(CHCK5)
         NESTER1 C1 ',' BREAK(',') . C2                          :F(ERR14)
         FUNCTION C2 '(' NOTANY('<') . C4 = 'A0AΣ(' C4           :F(ERR7)
CHCK5    N = N + 1
         FUNCTION C1 '(' NOTANY('<') . C4 = 'A' N 'AΣ(' C4       :S(CHCK5)
         FUNCTION STRIKE . C2 '(' NOTANY('<')                    :S(ERR13)
         TEXT6 C1 SPAN(DIGITS) . STORE6
CHCK6    FUNCTION 'EXP' = 'TOT'                                  :S(CHCK6)
*
*
*                    TRANSFORMATION OF FUNCTION
*
*
TRAN1    FUNCTION '/TAN' = '*COT'                                :S(TRAN1)
TRAN2    FUNCTION '/COT' = '*TAN'                                :S(TRAN2)
TRAN3    FUNCTION '/SEC' = '*COS'                                :S(TRAN3)
TRAN4    FUNCTION '/CSC' = '*SIN'                                :S(TRAN4)
TRAN5    FUNCTION NOTANY('A') . C0 'CSC' = C0 '1/SIN'            :S(TRAN5)
TRAN6    FUNCTION NOTANY('A') . C0 'SEC' = C0 '1/COS'            :S(TRAN6)
TRAN7    FUNCTION NOTANY('A') . C0 'COTH' = C0
+        'COSH_SINH'                                             :S(TRAN7)
```

```
TRAN8    FUNCTION NOTANY('A') . C0 'TANH' = C0
+        'SINH_COSH'                                          :S(TRAN8)
TRAN9    FUNCTION NOTANY('A') . C0 'COT' = C0
+        'COS_SIN'                                            :S(TRAN9)
TRAN10   FUNCTION NOTANY('A') . C0 'TAN' = C0
+        'SIN_COS'                                            :S(TRAN10)
+                                                             F(ARGM1)
*
*                DETECT FUNCTION OF THE FORMS
*                5 * X * 12 ** (SIN(X) - X ** 3)
*
*
DIFF1    FUNCTION SPAN(DIGITS) . C0 '**' ANY(LETTERS) . C1
+        (NOTANY(LETTERS) | '') . C2 = C0 '**(' C1 ')' C2    :S(DIFF1)
DIFF2    FUNCTION SPAN(DIGITS) . C0 '**(' = 'CC' C0 '('      :S(DIFF2)
*
*
*                DERIVING INTERNAL FORMS
*
*
ARGM1    N = 1
         KEK = 0
         JACK = 0
         LEVEL = 0
         GROUP1 = 0
         GROUP2 = 0
         GROUP3 = 0
         GROUP4 = 0
         COUNT4 = 0
         STORE1 =
         STORE2 =
         STORE3 =
         STORE7 =
         ALPHACONST =
         NUMBER1 = 0
         INDEX = ''
ARGM2    FUNCTION STRIKE . C0
+        = '#'                                               :F(BODY3)
         NUMBER1 = EQ(LEVEL,0) NUMBER1 + 1
ARGM3    TECH C0 SPAN(DIGITS) . C5                           :F(ERR5)
         NUMBER1 = EQ(C5,99) NUMBER1 - 1
         STORE1 = STORE1 C5 ','
         PAIR<N> = PAIR<N> + 1
         COUNT5 = 1
ARGM4    FUNCTION '#' (BREAK(')') ')') . C1                  :F(ERR6)
         C4 = 0
         C2 = C1
         EQ(NUMBER2,0)                                      :S(ARGM5)
         C2 INTEGROR                                        :F(ERR22)
         C2 '∂'                                             :S(ERR23)
ARGM5    C2 ANY('()') . C3 = 'ε'                            :F(ARGM6)
         C4 = IDENT(C3,')') C4 - 1                          :S(ARGM5)
         C4 = C4 + 1                                        :(ARGM5)
ARGM6    EQ(C4,0)                                           :S(ARGM7)
         FUNCTION '#' (C1 (BREAK(')') ')') . C2) . C1       :S(ARGM5)
+                                                            F(ERR6)
ARGM7    C2 = C1
```

```
          C2 POS(0) SPAN(DIGITS) . C3                                  :F(ARGM8)
          STORE2 = STORE2 C3 ','
ARGM8     C2 BREAK('(') '(' =
          C2 ')' RPOS(0) =
          COUNT4 = LT(COUNT4,12) COUNT4 + 1                            :F(ERR61)
          INDEX = INDEX C2 ','
          FUNCTION '_#' C1 = C1 '/Z(' COUNT4 ')'
          FUNCTION '#_' = 'Z(' COUNT4 ')/'
          C2 '<' =                                                     :F(ARGM9)
          FUNCTION '#' C1 ')' = 'Z(' COUNT4 ')'                        :(ARGM10)
ARGM9     FUNCTION '#' C1 = 'Z(' COUNT4 ')'
ARGM10    EQ(NUMBER2,0)                                                :S(ARGM2)
          C2 STRIKE . C2                                               :S(ERR3)
          STORE3 = STORE3 C2 ','
          FUNCTION NOTANY('A') . C6 C0 NOTANY('H') . C3 =
+         C6 '#' C3                                                    :F(BODY1)
          COUNT5 = COUNT5 + 1                                          :(ARGM4)
*
*
*                    COMPLEXITY ANALYSIS OF INTEGRAND
*
*
BODY1     IDENT(C0,'AA')                                               :S(NEST2)
          IDENT(C0,'BB')                                               :S(NEST5)
          GROUP3 = LT(C5,7) GROUP3 + 1                                 :S(BODY2)
          GROUP4 = LT(C5,19) GT(C5,12) GROUP4 + 1                      :S(BODY2)
          GROUP1 = EQ(C5,26) GROUP1 + 1                                :S(BODY2)
          GROUP2 = GROUP2 + 1
BODY2     STORE2 = STORE2 COUNT5 ','                                   :(ARGM2)
BODY3     EQ(NUMBER2,0)                                                :F(BODY4)
          N = 2
          LEVEL = 1
          STORE3 = STORE3 FUNCTION ','
          STORE6 = STORE6 PAIR<N> ','
          STORE7 = STORE7 COUNT4 ','
          PAIR<N> = 0
          FUNCTION = ''
          INDEX BREAK(',') . FUNCTION ',' =                            :S(ARGM2)
+                                                                       F(FORM1)
BODY4     FUNCTION 'T0T' = 'TT'                                        :F(BODY6)
BODY5     FUNCTION 'T0T' = 'TT'                                        :S(BODY5)
+                                                                       F(ARGM2)
BODY6     GT(GROUP2,1)                                                 :S(ERR10)
          GT(GROUP2,0) GT(GROUP1 + GROUP3 + GROUP4,0)                  :S(ERR11)
          GT(GROUP3,0) GT(GROUP4,0)                                    :S(ERR12)
*
*
*                       NESTING
*
*
NEST1     EQ(STORE6,0)                                                 :S(LAST1)
          C6 = C5
          FUNCTION 'A0A' = 'AA'                                        :F(NEST3)
+                                                                      S(ARGM2)
NEST2     FUNCTION 'Z(' COUNT4 ')' = 'Z2'
NEST3     N = 1
NEST4     FUNCTION 'A' N 'A' = 'BB'                                    :F(NEST6)
```

```
+                                                                    S(ARGM2)
NEST5   FUNCTION 'Z(' COUNT4 ')' = 'Z3'
        N = N + 1                                                    :(NEST4)
NEST6   C5 = C6
*
*
*                       INTEGRAND CLASSIFICATION
*
*
*
LAST1   STORE6 = STORE6 ','
        EQ(GROUP2,1)                                                 :F(LAST2)
        JACK = EQ(C5,27) TYPE + 10                                   :S(FORM1)
        JACK = EQ(C5,18) TYPE + 14                                   :S(FORM1)
        KEK = LT(C5,25) GT(C5,18) 1
        JACK = 4                                                     :(FORM1)
LAST2   EQ(GROUP1,1)                                                 :F(LAST3)
        KEK = GT(GROUP4,0) 1
        JACK = 3
        JACK = GT(GROUP3 + GROUP4,0) 2                               :(FORM1)
LAST3   GT(GROUP3 + GROUP4,0)                                        :F(LAST4)
        KEK = GT(GROUP4,0) 1
        JACK = 1                                                     :(PRCE1)
LAST4   JACK = 5                                                     :(FORM1)
*
*
*                       DETECT THE SPECIAL FUNCTIONS
*                       WITH FIXED COEFFICIENTS
*
*
PRCE1   EQ(JACK,1) EQ(KEK,0)                                         :F(FORM1)
        C0 = 1
        C1 =
        C2 = 1
        C4 =
        C5 =
        C6 = 0
        FUNCTION 'Z(1)/(1' K3 'Z(2))**2' = '1'                       :S(RICE6)
        FUNCTION 'Z(1)/((1' K3 'Z(2))**2)' = '1'                     :S(RICE6)
        FUNCTION 'Z(1)/(1' K3 'Z(2))' = '1'                          :F(PRCE2)
        STORE2 SPAN(DIGITS) . C7
        TYPE = EQ(C7,2) 300                                          :S(MODE1)
        TYPE = 60                                                    :(MODE1)
PRCE2   FUNCTION '/(Z(1)*(1' K3 'Z(2)))' =                           :F(PRCE3)
        STORE2 SPAN(DIGITS) . C7
        TYPE = EQ(C7,2) 400                                          :S(MODE1)
        TYPE = 76                                                    :(MODE1)
PRCE3   FUNCTION 'Z(1)/(Z(2)*(1' K3 'Z(3)))' = '1'                   :F(PRCE4)
        TYPE = 64                                                    :(MODE1)
PRCE4   FUNCTION 'Z(1)/(Z(3)*(1' K3 'Z(2)))' = '1'                   :F(PRCE5)
        TYPE = 88                                                    :(MODE1)
PRCE5   FUNCTION '/((Z(1)' K3 'Z(2))**2)' =                          :S(PRCE6)
        FUNCTION '/(Z(1)' K3 'Z(2))**2' =                            :F(PRCE7)
PRCE6   TYPE = 72                                                    :(MODE1)
PRCE7   FUNCTION '/(Z(1)' K3 'Z(2))' =                               :F(PRCE8)
        TYPE = 68                                                    :(MODE1)
PRCE8   FUNCTION 'Z(1)/(Z(2)' K3 'Z(3))' = '1'                       :F(PRCE9)
        TYPE = 80                                                    :(MODE1)
```

```
PRCE9     FUNCTION 'Z(1)/(Z(3)' K3 'Z(2))' = '1'                    :F(PRCE10)
          TYPE = 84                                                 :(MODE1)
PRCE10    FUNCTION '/(' (SPAN(DIGITS) . C0 | '1')
+         K3 ((SPAN(DIGITS) . C2 '*') | '') 'Z(1)**2)' =            :F(RICE1)
          TYPE = 750                                                :(MODE1)
*
*
*                    DETECT THE SPECIAL FUNCTIONS WITH
*                    VARIABLE COEFFICIENTS
*
*
RICE1     FUNCTION '/(' ((K4 SPAN(DIGITS) . C0 '*') | '')
+         'Z(1)' K3 ((SPAN(DIGITS) . C2 '*' ) | '')
+         'Z(2))' =                                                 :F(RICE2)
          TYPE = 100                                                :(MODE1)
RICE2     FUNCTION '/(' STYLE2 'Z(1))**2)' =                        :S(RICE3)
          FUNCTION '/' STYLE2 'Z(1))**2' =                          :F(RICE4)
RICE3     TYPE = 600                                                :(MODE1)
RICE4     FUNCTION '/' STYLE2 'Z(1))' =                             :F(RICE5)
          TYPE = 200                                                :(MODE1)
RICE5     FUNCTION 'Z(1)/(' STYLE2 'Z(2))**2)' = '1'                :S(RICE6)
          FUNCTION 'Z(1)/' STYLE2 'Z(2))**2' = '1'                  :F(RICE7)
RICE6     TYPE = 500                                                :(MODE1)
RICE7     FUNCTION 'Z(1)/' STYLE2 'Z(2))' = '1'                     :F(RICE8)
          TYPE = 300                                                :(MODE1)
RICE8     FUNCTION '/(Z(1)*' STYLE2 'Z(2)))' =                      :F(RICE9)
          TYPE = 400                                                :(MODE1)
RICE9     FUNCTION '/(' ((SPAN(DIGITS) . C0 '**2') | '1')
+         K3 ((SPAN(DIGITS) . C2 '**2*') | '')
+         'Z(1)**2)' =                                              :F(RICE10)
          TYPE = 700                                                :(MODE1)
RICE10    FUNCTION '/(' ((SPAN(DIGITS) . C0 '**2*')
+         | '1*' | '') 'Z(1)**2' K3 ((SPAN(DIGITS) . C2
+         '**2*') | '1*' | '') 'Z(2)**2)' =                         :F(RICE11)
          TYPE = 900                                                :(MODE1)
RICE11    FUNCTION STYLE3 = '1'                                     :F(FORM1)
          TYPE = 800
*
*
*                    DETECT VALIDITY OF SPECIAL FUNCTIONS
*
*
MODE1     FUNCTION 'Z3'                                             :S(ERR13)
          C4 = IDENT(C4,'+') ''
          C1 = IDENT(C1,'+') ''
          EQ(TYPE,800)                                              :F(MODE2)
          C0 = C2
          C5 = C1
          C1 '-' =
          TYPE = IDENT(C3,'/') TYPE + 5
MODE2     TYPE = LT(TYPE,100) IDENT(C1,'-') TYPE + 1
          STORE7 = C4 C0 ',' C1 C2 ','
          C1 = GE(TYPE,800) LE(TYPE,805) C5
          TYPE = GE(TYPE,700) LE(TYPE,900) IDENT(C1,'-') TYPE + 1
*
*
*                    GETTING INTERNAL FORMS
```

```
*
*
FORM1    FUNCTION INTEGROR NOTANY('(0123') . C1 = 'Z0' C1   :S(FORM1)
FORM2    STORE3 INTEGROR NOTANY('(0123') . C1 = 'Z0' C1     :S(FORM2)
         STORE0 = '(' NUMBER1 ',' NUMBER2 ')'
         NUMBER5 = 0
         C0 = INTEGROR                                      :(FORM6)
FORM3    FUNCTION ANY(LETTERS) . C0 NOTANY('0123(')         :S(FORM4)
         STORE3 ANY(LETTERS) . C0 NOTANY('0123(')           :F(FORM7)
FORM4    FUNCTION C0 NOTANY('012(') . C1 = 'Z1(' NUMBER5 ')'
+        C1                                                 :S(FORM4)
FORM5    STORE3 C0 NOTANY('0123(') . C1 = 'Z1(' NUMBER5
+        ')' C1                                             :S(FORM5)
FORM6    LETTERS BREAK(C0) . C1 C0
         C1 = SIZE(C1)
         NUMBER5 = LT(NUMBER5,6) NUMBER5 + 1                :F(ERR62)
         ALPHACONST = ALPHACONST C1 ','                     :(FORM3)
FORM7    EQ(NUMBER2,0)                                     :F(PRNT1)
         STORE3 BREAK(',') . FUNCTION ',' =
         STORE6 BREAK(',') ',' =
*
*
*                    OUTPUTING STACKS
*
*
PRNT1    STORE1 = '(' STORE1 '99,99)'
         STORE2 = '(' STORE2 '99,99)'
         STORE3 = '(' STORE3 '99,99)'
         STORE4 = '(' STORE4 '99,99)'
         ALPHACONST = '(' ALPHACONST '99,99)'
         STORE6 = '(' STORE6 '99,99)'
         STORE7 = '(' STORE7 '99,99)'
         C0 = EQ(FLAG,0) 33                                 :S(PRNT2)
         C0 = 66
         C0 = NE(FLAG,1) NE(FLAG,2) NE(FLAG,4) 99
PRNT2    EQ(NUMBER2,1) NE(MULTIPLE,1)                       :F(PRNT3)
         OUTPUT = PROBLEM<PROBLEMNO>
PRNT3    OUTPUT('OUTPUT',2,'(1X,80A1)')
         FLAG = '(' C0 ',' FLAG ',' COUNT2 ')'
         OUTPUT = FLAG
         OUTPUT = ALPHACONST
         OUTPUT = STORE0
         OUTPUT = STORE1
         OUTPUT = STORE2
         OUTPUT = STORE3
PRNT4    FUNCTION '@(' = 'ZZ('                              :S(PRNT4)
         OUTPUT = STORE4
         OUTPUT = STORE6
         OUTPUT = STORE7
         COUNT1 = COUNT1 + 1
         FUNCTION '?' =
         FUNCTION '?' =
         TYPE = LT(TYPE,10) 0
         TYPE = EQ(TEMP,1) -1
         FUNCTION 'Z(1)' EQ(TYPE,1000) = '1'
         OUTPUT = FUNCTION
         OUTPUT = JACK
```

```
          OUTPUT = KEK
          OUTPUT = TYPE
PRNT5     COUNT2 = 0
          OUTPUT('OUTPUT',10,'(1X,132A1)')
          PROBLEMNO = LT(PROBLEMNO,MULTIPLE) PROBLEMNO + 1   :S(TERM10)
          IDENT(KEEP,'')                                     :S(EXIT)
+                                                             F(STRT1)
*
*
*                    SET OF ERROR NUMBERS
*
*
*
ERR1      FLAG = 1                                           :(ERROR)
ERR2      FLAG = 2                                           :(ERROR)
ERR3      FLAG = 3                                           :(ERROR)
ERR4      FLAG = 4                                           :(ERROR)
ERR5      FLAG = 5
          C2 = C0                                            :(ERROR)
ERR6      FLAG = 6                                           :(ERROR)
ERR7      FLAG = 7
          C2 = C1                                            :(ERROR)
ERR8      FLAG = 8                                           :(ERROR)
ERR9      FLAG = 9                                           :(ERROR)
ERR10     FLAG = 10                                          :(ERROR)
ERR11     FLAG = 11                                          :(ERROR)
ERR12     FLAG = 12                                          :(ERROR)
ERR13     FLAG = 13                                          :(ERROR)
ERR14     FLAG = 14
          C2 = C1                                            :(ERROR)
ERR15     FLAG = 15                                          :(ERROR)
ERR20     FLAG = 20                                          :(ERROR)
ERR21     FLAG = 21                                          :(ERROR)
ERR22     FLAG = 22                                          :(ERROR)
ERR23     FLAG = 23                                          :(ERROR)
ERR24     FLAG = 24                                          :(ERROR)
ERR25     FLAG = 25                                          :(ERROR)
ERR26     FLAG = 26                                          :(ERROR)
ERR61     FLAG = 61                                          :(ERROR)
ERR62     FLAG = 62                                          :(ERROR)
ERR63     FLAG = 63
ERROR     OUTPUT = EQ(NUMBER2,1) NE(MULTIPLE,1) PROBLEM<PROBLEMNO>
          OUTPUT('OUTPUT',2,'(1X,80A1)')
          ALPHACONST = IDENT(INTEGROR,'?') '(23,'            :S(ERROR1)
          LETTERS BREAK(INTEGROR) . C1 INTEGROR
          C1 = SIZE(C1)
          ALPHACONST = '(' C1 ','
ERROR1    NE(FLAG,3) NE(FLAG,5) NE(FLAG,7) NE(FLAG,9)
+         NE(FLAG,13) NE(FLAG,14) NE(FLAG,20) NE(FLAG,25)    :S(ERROR3)
          ALPHACONST = ALPHACONST '100,'
ERROR2    C2 LEN(1) . KN =                                   :F(ERROR3)
          LETTERS BREAK(KN) . C1 KN
          C1 = SIZE(C1)
          ALPHACONST = ALPHACONST C1 ','                     :(ERROR2)
ERROR3    ALPHACONST = ALPHACONST '99,99)'
          FLAG = '(' 99 ',' FLAG ',' COUNT2 ')'
          OUTPUT = FLAG
```

```
          OUTPUT = ALPHACONST                                    :(PRNT5)
*
*
*
EXIT      OUTPUT('OUTPUT',10,'(1X,400A1)')
          OUTPUT = 'THIS IS THE END OF YOUR INPUT'
          OUTPUT('OUTPUT',2,'(1X,80A1)')
          OUTPUT = '(999,0,' COUNT1 ')'
          OUTPUT = '(23,99,99)'
END
```

APPENDIX B

ALTRAN PROCEDURES AND THEIR DESCRIPTIONS

1. Main procedure.
2. Procedure SYMDIF
3. Procedure SPEFCT, SPEC1, SPEC2, SPEC3, SPEC4
4. Procedure SQRT0, SQRT1, SQRT2, SQRT3, SQRT4, SQRT5
5. Procedure RATFCT, RATFUN
6. Procedure GETPWR
7. Procedure ARCFCT
8. Procedure EXPFCT, EXP, EXP2
9. Procedure HYPTRI
10. Procedure LOGFCT
11. Procedure TRIHY1, TRIHY2, TRIHYP

PROCEDURE MAIN

THIS IS THE MAIN PROCEDURE OF THE PROCESSOR OF THE PACKAGE
THE FUNCTIONS OF THE PROCEDURE INCLUDE THE FOLLOWING :

1. IT READS IN THE ALTRAN COMPREHENSIBLE FORMS PRODUCED BY THE ANLYZER OF THE PACKAGE WHICH TRANSLATES THE USER PROBLEMS
2. IF THE FUNCTION IS TO BE INTEGRATED THE MAIN PROCEDURE SIMPLIFIES THE FUNCTION WITH AN APPROPRIATE CHANGE OF VARAIBLE
3. IT DETERMINES THE ROUTINE THAT CAN DIFFERENTIATE OR INTEGRATE THE PARTICULAR FUNCTION, USING THE VARIABLES AND ARRAYS PASSED FROM THE ANALYZER OF THE PACKAGE
4. WHEN THE SOLUTION OF A PROBLEM IS OBTAINED IT WRITES THE COMPUTED RESULT ALONG WITH OTHER NECESSARY INFORMATION ON A FILE NEEDED BY THE WRITER OF THE PACKAGE WHERE THE RESULT WILL BE TRANSLATED INTO A USER COMPREHENSIBLE FORM

MOST VARIABLES THAT ARE USED IN THE PROCESSOR ARE EXTERNAL VARIABLES
THESE VARIABLES CONTAIN INFORMATION NEEDED BY THE PROCESSOR DURING
THE COMPUTATIONS OF THE DERIVATIVE AND INTEGRAL , AND WHEN THE COMPUTATIONS
ARE COMPLETED SOME OF THEM ARE PASSED TO THE WRITER
SEVERAL LOCAL VARIABLES ARE ALSO USED BY THE MAIN PROCEDURE TO PASS
NECESSARY VALUES TO THE ROUTINES CALLED BY THE PROCEDURE
IN THE FOLLOWING WE LIST THE SET OF ALL OF THE EXTERNAL VARIABLES USED
IN THE PROCESSOR STEP OF THE PACKAGE AS WELL AS THESE LOCAL VARIABLES
USED FOR TRANSFERRING VALUES BETWEEN ROUTINES.

INTERNAL VARIABLES	DESCRIPTION
A	THE INTEGER COEFFICIENT OF X**2 IN THE EXPRESSION A*X**2+B*X+C
B	THE INTEGER COEFFICIENT OF X IN THE EXPRESSION A*X**2+B*X+C
C	THE INTEGER CONSTANT IN THE EXPRESSION A*X**2+B*X+C
I	AN INTEGER VARIABLE. DURING THE SUBROUTINE CALLS I CONTAINS THE INTEGER CORRESPONDING TO THE FUNCTION NAME IN TYPE 2A, 2B, OR 2C(SEE SECTION 2.3). FOR TYPE 4 AND TYPE 6 (DELAY = 1), SINCE THE "FUNCTION" IS FIRST CONVERTED INTO THE SUM OF TWO PARTS, I CONTAINS THE INTEGER CORRESPONDING TO THE FUNCTION NAME WHICH APPEARS IN BOTH PARTS OF THE SUM.
M	THE POWER OF THE VARIABLE WITH RESPECT TO WHICH THE FUNCTION IS INTEGRATED
N	AN INTEGER VARIABLE EQUAL TO N1
OM	AN INTEGER VARIABLE USED WHEN DELAY EQUAL TO 1. IT IS SIMILIAR TO M BUT APPLIED ON THE SECOND PART OF THE FUNCTION.
D1	AN ALGEBRAIC VARIABLE EQUAL TO Z(1)**N1
D2	AN ALGEBRAIC VARIABLE EQUAL TO Z(2)**N2
D3	AN ALGEBRAIC VARIABLE EQUAL TO FCT/(D1*D2)
N1	AN INTEGER VARIABLE WHICH CONTAINS THE POWER OF Z(1)
N2	AN INTEGER VARIABLE WHICH CONTAINS THE POWER OF Z(2)

EXTERNAL VARIABLE	DESCRIPTION
MA	AN INTEGER ARRAY EQUAL TO STORE1
MB	AN ALGEBRAIC ARRAY OF ARGUMENTS EQUAL TO STORE3
M0	AN INTEGER ARRAY EQUAL TO ALPHACONST
M1	AN INTEGER ARRAY EQUAL TO STORE0
M2	AN INTEGER ARRAY WHICH IS USED FOR INDIRECT REPRESENTATION OF THE FUNCTION NAMES ON THE COMPUTED INTEGRAL. IF A FUNCTION NAME ON THE EXTERNAL VARIABLE "RESULT" (SEE BELOW FOR DETAILS) IS REPLACED BY Z(I),WHERE I IS AN INTEGER GREATER THAN 6, THEN THE (I-6) TH ENTRY OF M2 CONTAINS THE INTEGER CORRESPONDING TO THAT REPLACED FUNCTION NAME. M2 IS INITIALIZED TO 0 FOR EACH PROBLEM.
M3	AN INTEGER ARRAY EQUAL TO STORE2
M4	AN INTEGER ARRAY EQUAL TO STORE6
M5	AN ALGEBRAIC ARRAY OF ARGUMENTS. EACH ENTRY OF M5 CONTAINS THE ARGUMENT OF THE FUNCTION NAME WHICH IS INDIRECTLY REPRESENTATED ON THE COMPUTED INTEGRAL. GENERALLY, THE I TH ENTRY OF M5 IS THE ARGUMENT OF THE FUNCTION TO WHICH THE I TH ENTRY OF M2 CORRESPONDS, I.E. THE FUNCTION REPRESENTED BY Z(I+6) ON THE "RESULT". M5 IS ALSO INITIALIZED TO 0 FOR EACH PROBLEM.

M6 AN INTEGER ARRAY EQUAL TO STORE7
M7 AN INTEGER VARIABLE EQUAL TO TYPE
M8 AN INTEGER ARRAY EQUAL TO STORE4
LK AN INTEGER VARIABLE WHICH DENOTES THE NEXT AVAILABLE LOCATION OF M2(AND SO OF M5)
EXP AN ALGEBRAIC VARIABLE WHICH IS SET TO THE CONSTANT OBTAINED BY DIVIDING THE ARGUMENT OF THE FUNCTION 'EXP' BY THE VALUE OF SER(SEE BELOW)
KEK AN INTEGER VARIABLE SAME AS BEFORE
LAW AN INTEGER VARIABLE USED TO DENOTE TRANSFORMATION OF THE INTEGRAND.
IF LAW EQUAL TO 2 THE INTEGRAND HAS BEEN TRANSFORMED BY Y = EXP(F(X))
IF LAW EQUAL TO 3 THE INTEGRAND HAS BEEN TRANSFORMED BY Y = TAN(F(X)/2)
IF LAW EQUAL TO 4 THE INTEGRAND HAS BEEN TRANSFORMED BY Y = SQRT(F(X))
OTHERWISE LAW EQUAL TO 0 WHICH INDICATES NO TRANSFORMATION OF THE ABOVE KINDS HAS BEEN PERFORMED.
JACK AN INTEGER VARIABLE SAME AS BEFORE
SER AN ALGEBRAIC VARIABLE. THE INITIAL VALUE OF SER IS SET TO THE FIRST ENTRY OF MB, EXCEPT IN THE FOLLOWING CASES :
CASE I. JACK > 5 AND JACK < 10 SET SER = MB(2).
CASE II. JACK > 10 AND JACK < 14 SET SER = MB(3)
DURING THE COMPUTATIONS OF AN INTEGRAL THE VALUE OF SER WILL BE CHANGED IF THERE IS A CHANGE OF VARIABLE.
IN BRIEF, IF THE "FUNCTION" DOES NOT INVOLVE NESTING OF FUNCTIONS SER IS THE VARIABLE WITH RESPECT TO WHICH THE "FUNCTION" IS INTEGRATED. OTHERWISE IF THERE IS NESTING OF FUNCTIONS THEN SER IS FIRST SET TO 'Z0' AND IT MAY ALSO BE CHANGED PROVIDED THERE IS A CHANGE OF VARIABLE.
MSER AND NSER INDICATES THE NESTED FUNCTION AND ITS ARGUMENT RESPECTIVELY.
PARM AN ALGEBRAIC VARIABLE WHICH IS USED AS FOLLOWS :
I. IF THE "FUNCTION" IS A SPECIAL FUNCTION, PARM WILL BE SET TO THE COEFFICIENT OF THE SECOND TERM OF THE INTEGRAL, IF THAT TERM EXISTS. IN THIS CASE PARM WILL BE USED BY THE WRITER.
II. IF THE "FUNCTION" BELONGS TO OR IS CONVERTED TO FUNCTION TYPE 3, THEN PARM IS SET TO THE COEFFICIENT OF THE LAST TERM OF THE INTEGRAL, IF THAT TERM IS TO BE OBTAINED FROM THE INTEGRATION TABLE. IN THIS CASE PARM WILL ALSO BE USED BY THE WRITER.
III. OTHERWISE PARM IS USED TO CONTAIN THE COEFFICIENT OF THE NEXT TERM OF THE INTEGRAL WHEN THE INTEGRAL IS BEING COMPUTED. AFTER THE INTEGRAL IS OBTAINED PARM WILL BECOME A DUMMY VARIABLE.
MSER AN INTEGER VARIABLE WHICH INDICATES NESTING OF FUNCTIONS.
IF MSER EQUAL TO 0 THERE IS NO NESTING.
IF MSER IS POSITIVE MSER IS THE INTEGER CORRESPONDING TO THE NESTED FUNCTION.
NSER AN ALGEBRAIC VARIABLE. IF MSER = 0 NSER IS DUMMY.
IF MSER IS POSITIVE NSER IS THE ARGUMENT OF THE NESTED FUNCTION.
OEXP AN ALGENRAIC VARIABLE USED WHEN DELAY = 1(SEE BELOW). OEXP IS SET TO THE CONSTANT OBTAINED BY DIVIDING THE ARGUMEST OF THE FUNCTION EXP BY THE VALUE OF OSER.
OSER AN ALGEBRAIC VARIABLE USED WHEN DELAY = 1(SEE BELOW). OSER IS THE VARIABLE WITH RESPECT TO WHICH THE SECOND PART OF THE FUNCTION IS INTEGRATED.
OPARM AN ALGEBRAIC VARIABLE USED WHEN DELAY = 1(SEE BELOW). OPARM IS USED EXACTLY AS PARM EXCEPT THAT IT IS APPLIED ON THE SECOND PART OF THE FUNCTION.

DELAY AN INTEGER VARIABLE WHICH HAS THE FOLLOWING MEANINGS :

A. DELAY = 1. THE FUNCTION HAS BEEN CONVERTED INTO THE SUM OF TWO PARTS BY ONE OF THESE RULES :

1. SIN(A+B) + SIN(A-B) = 2*SIN(A)*COS(B)
2. SIN(A+B) - SIN(A-B) = 2*SIN(B)*COS(A)
3. COS(A+B) + COS(A-B) = 2*COS(A)*COS(B)
4. COS(A+B) - COS(A-B) = -2*SIN(A)*SIN(B)
5. SINH(A+B) - SINH(A-B) = 2*SINH(A)*COSH(B)
6. SINH(A+B) - SINH(A-B) = 2*SINH(B)*COSH(A)
7. COSH(A+B) + COSH(A-B) = 2*COSH(A)*COSH(B)
8. COSH(A+B) - COSH(A-B) = 2*SINH(A)*SINH(B)

THE INTEGRALS OF THESE TWO PARTS OF THE FUNCTION ARE OBTAINED SEPERATELY. THE FINAL INTEGRAL OF THE FUNCTION IS GIVEN AS THE SUM OF THESE INTEGRALS.

B. DELAY = 2. THE VALUE OF DELAY IS CHANGED FROM 1 TO 2 WHEN THE TWO COMPONENT PARTS OF THE FUNCTION MATCH THE TYPE 2B(MORE PRECISELY N = 1) BUT BOTH HAVE DIFFERENT VALUE OF M(SEE TYPE 4 AND 6). NOTE THAT IF THE VALUES OF M ARE EQUAL THE INTEGRAL OF THE SECOND PART OF THE FUNCTION CAN BE OBTAINED FROM THE FIRST BY AN OBVIOUS CHANGE OF VARIABLE. OTHERWISE THE SECOND INTEGRAL HAS TO BE INTEGRATED INDEPENDENTLY OF THE FIRST.

C. DELAY = 0. DUMMY.

RESULT AN ALGEBRAIC VARIABLE USED TO CONTAIN THE INTEGRAL. REPRESENTATION OF FUNCTION NAMES ON THE "RESULT" CAN BE CLASSIFIED INTO 3 TYPES :

TYPE 1. DIRECT REPRESENTATION. A FUNCTION NAME IS DIRECTLY REPRESENTED ON THE "RESULT" ACCORDING TO THE FOLLOWING :

A. Z(1) REPRESENTS SIN IF KEK = 0
IT REPRESENTS SINH IF KEK = 1
B. Z(2) REPRESENTS COS IF KEK = 0
IT REPRESENTS COSH IF KEK = 1
C. Z(3) REPRESENTS EXP
D. Z(4) REPRESENTS LN
E. Z(5) REPRESENTS SQRT
F. Z3 REPRESENTS TAN

FOR ALL THESE FUNCTIONS THE ARGUMENT OF EACH OF THEM IS SER.

TYPE 2. INDIRECT REPRESENTATION. FOR EACH Z(I) WHERE I > 6, Z(I) REPRESENTS THE FUNCTION NAME TO WHICH THE (I-6) TH ENTRY OF M2 CORRESPONDS. THE ARGUMENT OF THE FUNCTION NAME IS M5(I-6).

TYPE 3. DUMMY REPRESENTATION. Z(6) IS USED AS DUMMY REPRESENTATION WHICH REPRESENTS SER ITSELF.

STATUS AN INTEGER ARRAY EQUAL TO FLAG. DURING THE COMPUTATIONS OF AN INTEGRAL, THE VALUE OF STATUS INDICATES THE STATUS OF THE COMPUTATIONS OF THE INTEGRAL. THE VARIOUS POSSIBLE VALUES OF STATUS ARE INTERPRETED AS BELOWS :

A. STATUS(0) = 33. THE STATUS OF THE COMPUTATIONS AT THE CURRENT MOMENT IS OKAY, OR THE INTEGRAL HAS BEEN COMPLETED.

B. STATUS(0) = 55. A FORM CHECKING ON THE "FUNCTION" IS SUCESSFUL AND COMPUTATIONS(PERHAPS ANOTHER FORM IDENTIFICATION) MAY GO AHEAD.

C. STATUS(0) = 99. IF THIS VALUE TURNS OUT WHEN THE VALUE IS EXPECTED TO BE 33,55(SEE (A) AND (B) ABOVE), OR GREATER THAN 99(SEE (D) BELOW) THEN THE STATUS OF THE COMPUTATIONS IS IN AN ERRONEOUS STATE. IN THIS CASE THE COMPUTATIONS ARE IMMEDIATELY TERMINATED. THE APPROPRIATE ERROR NUMBER IS ALSO ASSIGNED TO STATUS(1).

D. STATUS(0) IS GREATER THEN 99. THE VALUE OF STATUS(0) INDICATES THE NEXT PROCEDURE TO BE CALLED.

THE SECOND ENTRY IS USED TO CONTAIN THE ERROR NUMBER.

STMT NO	DESCRIPTION
13	IF THE STATUS IS BAD REJECT THE PROBLEM.
14 - 17	IF END OF INPUT RETURN
18 - 26	INITIALIZE THE EXTERNAL VARIABLES.
29 - 33	CALL SYMDIF FOR DIFFERENTIATION
36 - 39	GET THE ARGUMENT OF A FUNCTION AND SET TO SER
40 - 69	PREPOCESS THE SPECIAL FUNCTION OR THE FUNCTION INVOLVING NESTING OF FUNCTIONS
42 - 45	ALL THE ARGUMENTS SHOULD BE THE SAME
46 - 68	TRANSFORM THE INTEGRAND BY Y = F(X) OR Y = OP(F(X)) THE LATTER CASE IS FOR FUNCTIONS INVOLVING NESTING OF FUNCTIONS
52	THE NESTED FUNCTION IS TAN COT TANH OR COTH THE DERIVITIVE OF THE NESTED FUNCTION SHOULD APPEAR
53	THE NESTED FUNCTION IS CSC SEC CSCH OR SECH
54 - 59	THE OTHER CASES
55	THE NESTED FUNCTION IS EXP
56	THE NESTED FUNCTION IS LN
57	THE NESTED FUNCTION IS SQRT
58	THE NESTED FUNCTION IS SIN COS SINH OR COSH
62 - 63	ADJUST SIGN WHEN THE NESTED FUNCTION IS COS CSC OR COT
64 - 67	STORE THE NESTED FUNCTION AND SET SER TO Z0 WHICH ALSO REPRESENTS THE NEW VARIABLE OF INTEGRATION
70 - 73	CALL SUBROUTINES TO INTEGRATE THE SPECIAL FUNCTION
74 - 93	UNIFY THE REPRESENTATION OF THE IDENTICAL FUNCTIONS WHICH HAVE THE SAME ARGUMENT. THE FUNCTION PROCESSED HERE INVOLVES EXP,TRI. FUNCTIONS OR HYP. FUNCTIONS.
94 - 101	SAME AS ABOVE BUT APLLIED ON THE FUNCTION WHICH INVOLVES LN OR SQRT.
102 - 107	ASSIGN VALUES TO LOCAL VARIABLES FOR SUBROUTINE CALL
109 - 115	IDENTIFY THE EXACT TYPE OF THE FUNCTION THAT ONLY INVOLVES TRIGONOMETRIC FUNCTIONS OR HYPERBOLIC FUNCTIONS
116 - 139	SUBROUTINE CALLS FOR INTEGRATION
140 - 150	IF TRANSFORMATION OF VARIABLE HAS BEEN MADE THIS PART TRANSFORMS THE INTEGRAL BACK INTO ITS ORIGINAL VARIABLE
150 - 154	SET SQRT(1) = 1 AND SQRT(0) = 0
155 - 157	PRINT THE RESULTS FOR THE SNOBOL WRITER
158	PROCESS THE NEXT PROBLEM

```
      PROCEDURE MAIN
         ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1) &
         D1,D2,D3,MB,M5,SER,OSER,PARM,OPARM,EXP,OEXP,FCT,RESULT,NSER
         EXTERNAL MA,MB,M1,M2,M3,M4,M5,M6,M7,LK,KEK,LAW,SER,JACK,OSER,
         PARM,OPARM,EXP,OEXP,DELAY,RESULT,FCT,STATUS
         INTEGER M0,M1,M2,M3,M4,M6,M7,M8,MA,LK,KEK,LAW,JACK,DELAY,STATUS,
         I,J,M,N,P,N1,N2,OM,TEST,MSER
         ARRAY M0,M1,M3,M4,M6,M8,MA,MB
         ARRAY (0:2) STATUS
         ARRAY (6) M2,M5
         RATIONAL A,B,C
START:   READ (2) STATUS,M0
MAIN1:   IF(STATUS(0) == 99) GO TO MAIN4
         IF(STATUS(0)==999) DO
               WRITE (3) STATUS
               RETURN
         DOEND
         M2 = 0
         M5 = 0
         MSER = 0
         NSER = 0
         RESULT=0
         PARM = 1
               LK = 0
               LAW = 0
               DELAY = 0
         READ (2) M1,MA,M3,MB,M8,M4,M6,FCT,JACK,KEK,M7
# DIFFERENTIATION
      IF(M1(2).NE.1) DO
               SYMDIF
               WRITE (3) STATUS,M0,M1,MA,MB,RESULT,M3
               GO TO START
      DOEND
         STATUS(0) = 99
# INTEGRATION
         IF(JACK > 10 .AND. JACK < 14) J = 3
         ELSE IF(JACK > 5 .AND. JACK < 10) J = 2
         ELSE J = 1
         SER = MB(J)
         IF(M4(1) .NE. 0 .OR. M7 > 0) DO
                     STATUS(1) = 16
                     DO I = J , 100
                           IF(MB(I + 1) .EQ. 99) GO TO MAIN2
                           IF(MB(I) .NE. MB(I + 1)) GO TO MAIN4
                     DOEND
MAIN2:               STATUS(1) = 17
                     FCT = FCT / DIFF(MB(J),Z0)
                     SPLIT(FCT,Z0,D1,D2)
                     IF(D2 .NE. 1) GO TO MAIN4
               IF(M4(1) .NE. 0) DO
                     OM = M4(1) - KEK * 12
                     IF(OM .EQ. 5 .OR. OM .EQ. 6) FCT = FCT / Z2 ** 2
                     ELSE IF(OM .EQ. 3 .OR. OM .EQ. 4) FCT = FCT / (Z2 * Z3)
                     ELSE DO
                                 IF(M4(1) .EQ. 26) FCT = FCT / Z3
                                 IF(M4(1) .EQ. 27) FCT = FCT * MB(J)
                                 IF(M4(1) .EQ. 28) FCT = FCT * Z3 * 2
                                 ELSE IF(OM .LT. 3) FCT = FCT / Z2
                     DOEND
                     SPLIT(FCT,Z2,D1,D2)
                     IF(D2 .NE. 1) GO TO MAIN4
                     IF(M4(1) .EQ. 2 .OR. M4(1) .EQ. 3 .OR. M4(1) .EQ. 6)  &
                                       FCT = -FCT
                     FCT = FCT(Z3 = Z0)
                     MSER = M4(1)
                     NSER = MB(J)
                     SER = Z0
               DOEND
         DOEND
```

```
      IF(M7 > 0) DO
            SPEFCT
            GO TO MAIN4
      DOEND
      IF(JACK .EQ. 1 .OR. JACK .EQ. 2) DO
            STATUS(1) = JACK + 35
            M = 0
            OM = M3(1) + M3(2)
            IF(M1(1) .EQ. 3 .AND. (OM + M3(3)) .NE. 3) GO TO MAIN4
            ELSE IF(JACK .EQ. 2 .AND. M3(2) .NE. 1) GO TO MAIN4
            DO J = 1 , M1(1)
                  DO I = 1 , M3(J) - 1
                        IF(MB(I+M) .EQ. MB(I+M+1)) FCT=FCT(Z(I+M)=Z(J))
                        ELSE IF(JACK .EQ.1 .AND. OM .NE. 101) GO TO MAIN4
                        ELSE IF(JACK .EQ. 2 .AND. OM .NE. 3) GO TO MAIN4
                        ELSE GO TO MAIN5
                  DOEND
                  FCT = FCT(Z(M3(J)+M) = Z(J))
                  M = M + M3(1)
            DOEND
            MB(2) = MB(M3(1) + 1)
            M3(1) = 1
            IF(M3(2) .NE. 99) M3(2) = 1
      DOEND
      ELSE IF(JACK == 4 .OR. JACK == 10 .OR. JACK == 14) DO
            STATUS(1) = 52
            DO I = 2 , M3(1)
                  IF(MB(I-1) .NE. MB(I)) GO TO MAIN4
                  FCT = FCT(Z(I) = Z(1))
            DOEND
            M3(1) = 1
      DOEND
MAIN5:      N1 = DEG(FCT,Z(1))
            N2 = DEG(FCT,Z(2))
            N = N1
            D1 = Z(1) ** N1
            D2 = Z(2) ** N2
            D3 = FCT / (D1 * D2)
      STATUS(1) = 20
      IF(JACK.EQ.1) DO
            SPLIT(D3,Z(1),D1,D2)
            IF(D2.NE.1) JACK = 15
            SPLIT(D3,Z(2),D1,D2)
            IF(D2.NE.1) JACK = 15
            STATUS(0) = 105
      DOEND
```

```
                    IF(JACK == 2) EXP1(I,N1,N2,D3)
                ELSE IF(JACK == 3) EXPFCT(M,D3)
                ELSE IF(JACK == 4) ARCFCT(M,N,I,A,B,N1,D3)
                ELSE IF(JACK == 5) RATFCT
                ELSE IF(JACK<11.AND.JACK>5) LOGFCT(A,B,M,N,I,D3)
                ELSE IF(JACK<15.AND.JACK>10) SQRT0(M,N,I,A,B,C,D3)
                ELSE IF(JACK == 15)TRIHYP
                    IF(STATUS(0)>99) DO
MAIN3:                          IF(STATUS(0) == 100) SQRT1(B,C)
                        IF(STATUS(0) == 101) SQRT2(M,N,A,B)
                        ELSE IF(STATUS(0) == 102) SQRT3(M,N,A,B,C)
                        ELSE IF(STATUS(0) == 104) RATFCT
                        ELSE IF(STATUS(0) == 105) HYPTRI(I,M,OM,N1,N2,D3)
                            IF(STATUS(0) == 109) SQRT4(M,N,I,B)
                        ELSE IF(STATUS(0) == 110) SQRT5(M,N,I,B)
                            IF(STATUS(0)>99) GO TO MAIN3
                    DOEND
                    IF(STATUS(0)==55) DO
                            STATUS(0)=33
                            IF(JACK == 1) TRIHY1(N1,N2)
                        ELSE IF(JACK == 2) TRIHY2(M,N,I,OM)
                        ELSE IF(JACK == 5) RATFUN
                        ELSE IF(JACK == 6) EXP2(I,N1)
                    DOEND
                IF(LAW.NE.0) DO
                    IF(LAW == 3) SER = SER / 2
                    IF(LAW == 2) P = 3
                    ELSE IF(LAW == 4) P = 5
                    IF(LAW == 3) RESULT = RESULT(Z0 = Z3)
                    ELSE RESULT = RESULT(Z0 = Z(P))
                    DO I=1,LK
                        IF(LAW == 3) M5(I) = M5(I)(Z0 = Z3)
                        ELSE M5(I) = M5(I)(Z0 = Z(P))
                    DOEND
                DOEND
        DO I = 1 , LK
            IF(M2(I) .EQ. 28 .AND. M5(I) .EQ. 1) RESULT = RESULT(Z(I+6)=1)
            IF(M2(I) .EQ. 28 .AND. M5(I) .EQ. 0) RESULT = RESULT(Z(I+6)=0)
        DOEND
MAIN4:  IF(STATUS(0) == 99) WRITE (3) STATUS,M0
        ELSE WRITE (3) STATUS,M0,M1,M2,M5,RESULT,MSER,PARM,M6,
        KEK,M7,SER,NSER,M8
        GO TO START
    END
```

```
                PROCEDURE SYMDIF

THIS PROCEDURE DIFFERENTIATES ALL THE FUNCTIONS ACCEPTABLE TO THE PACKAGE

STMT  NO              DESCRIPTION

 10               PARTIALLY DIFFERENTIATE THE FUNCTION W.R.T. Z0
                  BY CONSIDERING ANY FUNCTION NAME AS A CONSTANT
 11               IF NO FUNCTION NAME RETURN
 13 - 41          PARTIALLY DIFFERENTIATE THE FUNCTION NAMES W.R.T. Z0
                  STARTING FROM THE TOP LEVEL TO THE LOWEST LEVEL
                  ZZ(I) CONTAINS THE DERIVITIVE OF THE FUNCTION NAME Z(I)
                  IN THE FUNCTION
                  FOR EXAMPLE
                  IF Z(I) = SIN(X) ZZ(I) WILL BE COS(X)
 13 - 41          DIFFERENTIATE THE FUNCTION NAMES AT THE TOP LEVEL
 19 - 37          DIFFERENTIATE THE FUNCTION NAMES AT THE THIRD
                  LOWEST LEVEL
 25 - 37          DIFFERENTIATE THE FUNCTION NAMES AT THE SECOND
                  LOWEST LEVEL
 31 - 35          DIFFERENTIATE THE FUNCTION NAMES AT THE LOWEST LEVEL
 42               THE FINAL RESULT IS CONTAINED IN RESULT


 PROCEDURE SYMDIF
    ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1) &
                              MB,FCT,RESULT,SAVE1,SAVE3,
                              SAVE5,SAVE8,OSAVE1,OSAVE2,OSAVE3,OSAVE4,
                                          OSAVE5,OSAVE6,OSAVE7,OSAVE8
    EXTERNAL M1,MB,M4,M6,FCT,RESULT
    INTEGER ARRAY M1,M4,M6
    INTEGER I,J,K,L,I1,I2,I3,J1,K1,K2,K3
    ARRAY MB
    RESULT = DIFF(FCT,Z0)
    IF(M1(1)==0) RETURN
    SAVE1=0
    DO I = 1,M1(1)
          I1=M4(I)
          K1=M6(I)
          SAVE3=0
          OSAVE1 = DIFF(FCT,Z(I))*ZZ(I)
          OSAVE2 = DIFF(MB(I),Z0)
          DO J=1,I1
                OSAVE3 = DIFF(MB(I),Z(K1+J))*ZZ(K1+J)
                OSAVE4=DIFF(MB(K1+J),Z0)
                I2=M4(K1+J)
                K2=M6(K1+J)
                SAVE5=0
                DO K=1,I2
                      OSAVE5=DIFF(MB(K1+J),Z(K2+K))*ZZ(K2+K)
                      OSAVE6=DIFF(MB(K2+K),Z0)
                      I3=M4(K2+K)
                      K3=M6(K2+K)
                      SAVE8=0
                      DO L=1,I3
                            OSAVE7=DIFF(MB(K2+K),Z(K3+L))*ZZ(K3+L)
                            OSAVE8=DIFF(MB(K3+L),Z0)
                            SAVE8=SAVE8+OSAVE7*OSAVE8
                      DOEND
                      SAVE5=SAVE5+OSAVE5*(OSAVE6+SAVE8)
                DOEND
                SAVE3=SAVE3+OSAVE3*(OSAVE4+SAVE5)
          DOEND
          SAVE1=SAVE1+OSAVE1*(OSAVE2+SAVE3)
    DOEND
    RESULT=RESULT+SAVE1
 END
```

PROCEDURE SPEFUN

THIS PROCEDURE HANDLES THE SET OF ALL SPECIAL FUNCTIONS LISTED IN THE APPENDIX F. THE VALUE OF M7 IS USED TO CALL THE ROUTINE WHICH CAN INTEGRATE THE PARTICULAR FUNCTION

STMT NO	DESCRIPTION
9 - 14	THE INTEGRAND HAS THE FORM 1/(A+A*OP(X)**2) WHERE OP IS SIN COS SINH OR COSH
15 - 19	SUBROUTINE CALLS

```
PROCEDURE SPEFCT
   ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1) &
   D1,D2,D3,MB,FCT,PARM
   EXTERNAL MB,M6,M7,FCT,PARM,STATUS
   INTEGER M6,M7,STATUS
   ARRAY MB,M6
   ARRAY (0:2) STATUS
   STATUS(1) = 18
        IF(M7 == 750 .OR. M7 == 751) DO
             IF(M6(1) .NE. M6(2) .AND. M6(1) .NE. -M6(2)) RETURN
             M7 = M7 - 50
             FCT = FCT / M6(1)
             M6 = 1
        DOEND
        IF(M7 .LT. 92) SPEC1
        ELSE IF(((M6(1) .EQ. M6(2)) .OR. (M6(1) .EQ. -M6(2))) .AND.
        ((M7 .GE. 900 .AND. M7 .LT. 1000) .OR. M7 .LT. 800)) SPEC2
        ELSE IF(M7 .LT. 700) SPEC3
        ELSE SPEC4
END
```

PROCEDURE SPEC1

THIS PROCEDURE HANDLES THE SET OF THE SPECIAL FUNCTIONS THAT
HAVE EQUAL COEFFICIENTS . FCT INITIALLY CONTAINS THE COEFFICIENT
THAT IS ASSOCIATED WITH THE PARTICULAR FUNCTION
THE PROCEDURE SET IN TWO EXTERNAL VARIABLES RESULT AND PARM THE
COEFFICIENTS OF THE EVENTUAL INTEGRAL PROVIDED FROM THE INTEGRATION TABLE

STMT NO	DESCRIPTION
7 - 8	FORM CHECKING
9 - 19	THE FUNCTION HANDLED HERE IS ANY OF THE FOLLOWING FORMS 1 F'(X)/(A**2+-A**2*OP(F(X))**2) 2 F'(X)/(A**2*OP1(F(X))**2+-B**2*OP2(F(X))**2) WHERE OP IS SIN OR COS AND EITHER OF OP1 AND OP2 IS SIN AND THE OTHER COS M7 WILL EVENTUALLY HAVE THE TYPE NUMBER OF THE SPECIAL FUNCTION AND RESULT CONTAINS THE FACTOR OF THE COEFFICIENT OF THE EVENTUAL INTEGRAL THAT DOES NOT INVOLVE A FUNCTION NAME (E.G. SQRT) THE OTHER FACTOR OF THE COEFFICIENT IS CONTAINED IN M6, AN EXTERNAL ARRAY
20 - 35	THE INTEGRAND HANDLED HERE HAS EITHER OF THE FOLLOWING FORMS 1 F'(X)/(A+A*OP(F(X))) 2 F'(X)*OP(F(X))/(A+A*OP(F(X))) 3 F'(X)/(OP(F(X))*(A+A*OP(F(X)))) 4 F'(X)/(A+A*OP(F(X)))**2 5 F'(X)*OP(F(X))/(A+A*OP(F(X)))**2 WHERE OP IS SIN OR COS M7 AND RESULT ARE THE SAME AS ABOVE WHILE PARM CONTAINS THE FACTOR OF THE COEFFICIENT OF THE SECOND TERM OF THE EVENTUAL INTEGRAL (WHERE APPLICIABLE) THAT DOES NOT INVOLVE A FUNCTION NAME LIKEWISE M6 CONTAINS THE FACTORS OF ANY COEFFICIENT THAT INVOLVE A FUNCTION NAME

```
PROCEDURE SPEC1
   ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1) FCT,PARM,RESULT
   INTEGER ARRAY M3,MA
   EXTERNAL M3,M7,MA,FCT,PARM,RESULT,STATUS
   INTEGER M7,STATUS
   ARRAY (0:2) STATUS
   IF(M3(2) .EQ. 99) RETURN
   IF(M7 .GE. 64 .AND. M7 .LE. 67 .AND. M3(2) .NE. 2) RETURN
   ELSE IF(M7 .GE. 80 .AND. M7 .LE. 83 .AND. M3(2) .EQ. 2) RETURN
   IF(MA(1) .EQ. 2 .AND. (M7 .LT. 68 .OR. M7 .GT. 73)) M7 = M7 + 2
   IF(MA(1) .EQ. 2 .AND. M7 .EQ. 69) FCT = -FCT
   ELSE DO
        IF(M7 .GT. 71) FCT = FCT / 2
        IF(M7 .EQ. 85 .OR. M7 .EQ. 87) FCT = -FCT
        IF(M7 .GT. 84) M7 = M7 - 4
        IF(M7 .EQ. 80 .OR. M7 .EQ. 83 .OR. M7 .EQ. 85 .OR.
             M7 .EQ. 87) PARM = -FCT
        ELSE IF(M7 > 73) PARM = FCT
   DOEND
   IF(M7 .EQ. 60 .OR. M7 .EQ. 63 .OR. M7 .EQ. 77 .OR.
        M7 .EQ. 78 .OR. M7 .EQ. 66 .OR. M7 .EQ. 67 .OR. M7 .EQ.
             86 .OR. M7 .EQ. 87) RESULT = -FCT
   ELSE RESULT = FCT
   STATUS(0) = 33
END
```

PROCEDURE SPEC2

THIS PROCEDURE SET IN TWO EXTERNAL VARIABLES RESULT AND
PARM THE COEFFICIENTS OF THE TWO TERMS OF THE EVENTUAL INTEGRAL
THE COMPLETE INTEGRAL IS GIVEN IN THE FINAL STEP OF THE PACKAGE
WHERE THESE COEFFICIENTS ARE USED TO ATTACH TO THE
APPROPRIATE TERMS. THE SET OF SPECIAL FUNCTIONS THAT ARE TREATED HERE ARE :

FORM	M7 (FORM NUMBER)
F'(X)*SIN(F(X))/(1+-COS(F(X)))	60,61
F'(X)*COS(F(X))/(1+-SIN(F(X)))	62,63
F'(X)SIN(F(X))/(COS(F(X))*(1+-COS(F(X))))	64,65
F'(X)*COS(F(X))/(SIN(F(X))*(1+-SIN(F(X))))	66,67
F'(X)/(SIN(F(X))+-COS(F(X)))	68,69
F'(X)/((SIN(F(X))+-COS(F(X)))**2)	72,73
F'(X)/(SIN(F(X))*(1+-COS(F(X))))	76,77
F'(X)/(COS(F(X))*(1+-SIN(F(X))))	78,79
F'(X)*SIN(F(X))/(SIN(F(X))+-COS(F(X))))	80,81
F'(X)*COS(F(X))/(SIN(F(X))+-COS(F(X))))	82,83
F'(X)*SIN(F(X))/(COS(F(X))*(1+-SIN(F(X))))	88,89
F'(X)*COS(F(X))/(SIN(F(X))*(1+-COS(F(X))))	90,91

STMT NO	DESCRIPTION
7 - 9	FORM CHECKING
10 - 19	THE VARIABLE FCT INITIALLY HAS THE COEFFICIENT THAT IS ASSOCIATED WITH THE SPECIAL FUNCTION ADJUST M7 TO THE TYPE NUMBER OF THE INTEGRAND AND IT IS TO BE USED BY THE SNOBOL WRITER RESULT AND PARM RESPECTIVELY ARE SET TO VALUES OF THE COEFFICIENTS OF THE FIRST AND SECOND TERM OF THE EVENTUAL INTEGRAL (BOTH OF THE TERMS OF THE INTEGRAL ARE STORED IN THE ARRAY IN THE SNOBOL WRITER

```
  PROCEDURE SPEC2
     ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1) FCT,PARM,RESULT
     INTEGER M3,M6,M7,MA,STATUS
     EXTERNAL M3,M6,M7,MA,FCT,PARM,RESULT,STATUS
     ARRAY M3,M6,MA
     ARRAY (0:2) STATUS
     IF(M7 .GE. 900 .AND. M3(2) .EQ. 2) RETURN
     ELSE IF(M3(2) .LT. 900 .AND. M3(2) .NE. 99) RETURN
     IF(M7 .GE. 700 .AND. M7 .LE. 901) DO
          IF(M6(1) .NE. M6(2)) RETURN
          RESULT = FCT / M6(1) ** 2
          IF(M7 .EQ. 700 .OR. M7 .EQ. 901) RESULT = RESULT / 2
          IF(M7 .EQ. 700 .OR. M7 .EQ. 701) M7 = M7 + MA(1) * 2 - 1
          ELSE M7 = M7 + 50
          IF(M7 .EQ. 704 .OR. (M7 .EQ. 951 .AND. MA(1) .EQ. 2)) &
               RESULT = -RESULT
          IF(M7 .EQ. 951 .AND. MA(1) .EQ. 2) M6(1) = M6(2)
          GO TO AMPX1
     DOEND
          FCT = FCT / M6(1)
          IF(M7 .GT. 400) FCT = FCT / M6(1)
          IF(M6(1) .EQ. -M6(2)) M7 = M7 + MA(1) + 2
          ELSE M7 = M7 + MA(1)
          IF(M7 > 451) DO
               RESULT = FCT / 2 * (-1) ** MA(1)
               PARM = FCT / 6 * (-1) ** (MA(1) + 1)
               IF(M7 .EQ. 601 .OR. M7 .EQ. 604) RESULT = -RESULT
               IF(M7 .EQ. 601 .OR. M7 .EQ. 602) PARM = -PARM
               GO TO AMPX1
          DOEND
          RESULT = FCT
          PARM = FCT * (-1) ** (MA(1) + 1)
          IF(M7 .EQ. 201 .OR. M7 .EQ. 204 .OR.
               M7 .EQ. 303 .OR. M7 .EQ. 304) RESULT = -RESULT
AMPX1:  STATUS(0) = 33
     END
```

PROCEDURE SPEC3

THIS PROCEDURE SET IN TWO EXTERNAL VARIABLES RESULT AND
PARM THE COEFFICIENTS OF THE TWO TERMS OF THE EVENTUAL INTEGRAL
THE COMPLETE INTEGRAL IS GIVEN IN THE FINAL STEP OF THE PACKAGE
WHERE THESE COEFFICIENTS ARE USED TO ATTACH TO THE
APPROPRIATE TERMS. THE SET OF SPECIAL FUNCTIONS THAT ARE TREATED HERE ARE :

FORM	M7 (FORM NUMBER)
F'(X)/(A*SIN(F(X))+B*COS(F(X)))	100
F'(X)/(A+B*SIN(F(X)))	200
F'(X)/(A+B*COS(F(X)))	200
F'(X)*SIN(F(X))/(A+B*SIN(F(X)))	300
F'(X)*COS(F(X))/(A+B*COS(F(X)))	300
F'(X)/(SIN(F(X))*(A+B*SIN(F(X)))	400
F'(X)/(COS(F(X))*(A+B*COS(F(X)))	400
F'(X)*SIN(F(X))/(A+B*SIN(F(X)))**2	500
F'(X)*COS(F(X))/(A+B*COS(F(X)))**2	500
F'(X)/(A+B*SIN(F(X)))**2	600
F'(X)/(A+B*COS(F(X)))**2	600

STMT NO	DESCRIPTION
7 - 46	THE VARIABLE FCT INITIALLY HAS THE COEFFICIENT THAT IS ASSOCIATED WITH THE SPECIAL FUNCTION ADJUST M7 TO THE TYPE NUMBER OF THE INTEGRAND AND IT IS TO BE USED BY THE SNOBOL WRITER RESULT AND PARM RESPECTIVELY ARE SET TO FACTORS OF THE COEFFICIENTS OF THE FIRST AND SECOND TERM OF THE EVENTUAL INTEGRAL THAT DOES NOT INVOLVE ANY FUNCTION NAME AND THE OTHER FACTORS ARE STORED IN THE ARRAY M6 BOTH TERMS OF THE EVENTUAL INTEGRAL ARE STORED IN THE ARRAY IN THE SNOBOL WRITER

```
      PROCEDURE SPEC3
         ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1) FCT,PARM,RESULT
         INTEGER M3,M6,M7,MA,STATUS
         EXTERNAL M3,M6,M7,MA,FCT,PARM,RESULT,STATUS
         ARRAY M3,M6,MA
         ARRAY (0:2) STATUS
         IF(M7 .EQ. 100) DO
               IF(M3(2) .EQ. 99) RETURN
               M6(3) = M6(1)**2 + M6(2)**2
               IF(MA(1) .EQ. 1) M6(1) = M6(2)
               M7 = 12
               RESULT = FCT
               GO TO SRIN1
         DOEND
         IF(M3(2) .NE. 99) RETURN
               M6(3) = M6(1)**2 - M6(2)**2
               IF(M7 .EQ. 300) DO
                     M7 = 16
                     RESULT = 1 / M6(2)
                     PARM = -M6(1) / M6(2)
               DOEND
               ELSE IF (M7 .EQ. 400) DO
                     M7 = 18
                     RESULT = 1 / M6(1)
                     PARM = -M6(2) / M6(1)
               DOEND
               ELSE IF(M7 .EQ. 500 .OR. M7 .EQ. 600) DO
                     IF(M7 == 500) RESULT = M6(1) / M6(3)
                     ELSE RESULT = M6(2) / M6(3)
                     IF(M7 == 500) PARM = -M6(2) / M6(3)
                     ELSE PARM = M6(1) / M6(3)
                     IF((M7 == 500 .AND. MA(1) == 1) .OR.
                     (M7 == 600 .AND. MA(1) == 2)) RESULT = -RESULT
                     M7 = 20
               DOEND
                     IF(M6(3) > 0) PARM = FCT * 2 * PARM
                     ELSE DO
                           M6(3) = -M6(3)
                           M7 = M7 + 6
                           PARM = FCT * PARM
                     DOEND
                     IF(M7 .GE. 200) M7 = M7 - 100
                     IF(M7 .EQ. 100 .OR. M7 .EQ. 106) RESULT = PARM
                     ELSE RESULT = RESULT * FCT
                     M7 = M7 + MA(1)
SRIN1:         STATUS(0) = 33
      END
```

PROCEDURE SPEC4

THIS PROCEDURE SET IN TWO EXTERNAL VARIABLES RESULT AND
PARM THE COEFFICIENTS OF THE TWO TERMS OF THE EVENTUAL INTEGRAL
THE COMPLETE INTEGRAL IS GIVEN IN THE FINAL STEP OF THE PACKAGE
WHERE THESE COEFFICIENTS ARE USED TO ATTACH TO THE
APPROPRIATE TERMS. THE SET OF SPECIAL FUNCTIONS THAT ARE TREATED HERE ARE :

FORM	M7 (FORM NUMBER)
F'(X)/(A**2+-B**2*SIN(F(X))**2)	700,701
F'(X)/(A**2+-B**2*COS(F(X))**2)	700,701
F'(X)*SIN(F(X))*SQRT(1+-A**2*SIN(F(X))**2)	800,801
F'(X)*COS(F(X))*SQRT(1+-A**2*SIN(F(X))**2)	800,801
F'(X)*SIN(F(X))/SQRT(1+-A**2*SIN(F(X))**2)	805,806
F'(X)*COS(F(X))/SQRT(1+-A**2*SIN(F(X))**2)	805,806
F'(X)/(A**2*SIN(F(X))**2+-B**2*COS(F(X))**2)	900,901
F'(X)*SIN(LN(F(X)))	1000
F'(X)*COS(LN(F(X)))	1000

STMT NO	DESCRIPTION
7	FORM CHECKING
8 - 59	THE VARIABLE FCT INITIALLY HAS THE COEFFICIENT THAT IS ASSOCIATED WITH THE SPECIAL FUNCTION ADJUST M7 TO THE TYPE NUMBER OF THE INTEGRAND AND IT IS TO BE USED BY THE SNOBOL WRITER RESULT AND PARM RESPECTIVELY ARE SET TO FACTORS OF THE COEFFICIENTS OF THE FIRST AND SECOND TERM OF THE EVENTUAL INTEGRAL THAT DOES NOT INVOLVE ANY FUNCTION NAME AND THE OTHER FACTORS ARE STORED IN THE ARRAY M6 BOTH TERMS OF THE EVENTUAL INTEGRAL ARE STORED IN THE ARRAY IN THE SNOBOL WRITER

```
PROCEDURE SPEC4
   ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1) FCT,PARM,RESULT
   INTEGER I,M3,M6,M7,MA,STATUS
   EXTERNAL M3,M6,M7,MA,FCT,PARM,RESULT,STATUS
   ARRAY M3,M6,MA
   ARRAY (0:2) STATUS
   IF(M7 .GE. 900 .AND. M3(2) .EQ. 2) RETURN
   IF(M7 .GE. 700 .AND. M7 .LE. 701) DO
         RESULT = FCT / M6(1)
         IF(M7 == 700) DO
               M6(3) = M6(1) ** 2 + M6(2) ** 2
               M7 = 12
         DOEND
         ELSE DO
               M6(3) = M6(1) ** 2 - M6(2) ** 2
               M7 = 12
               IF(M6(3) < 0) DO
                     M6(3) = -M6(3)
                     RESULT = RESULT / 2
                     M7 = M7 + 2
               DOEND
         DOEND
   M7 = M7 + MA(1)
   DOEND
   ELSE IF(M7 .GE. 800 .AND. M7 .LT. 807) DO
         IF(M7 .EQ. 805 .OR. M7 .EQ. 806)  RESULT = FCT / M6(2)
         ELSE DO
               RESULT = FCT / 2
               IF(MA(1) .EQ. 1 .AND. M7 .EQ. 800) &
                     PARM = -(1 + M6(2) ** 2) / (2 * M6(2)) * FCT
               ELSE IF(MA(1) .EQ. 1 .AND. M7 .EQ. 801) &
                     PARM = -(1 - M6(2) ** 2)/(2 * M6(2)) * FCT
               ELSE IF(MA(1) .EQ. 2 .AND.
                     M7 < 802) PARM = FCT / (2 * M6(2))
         DOEND
         IF(MA(1) .EQ. 1) RESULT = -RESULT
         M6(2) = M6(2) ** 2
         IF((M7 == 800 .OR. M7 == 805) .AND. MA(1) == 1) M6(3) = 1 + M6(2)
         M7 = M7 + MA(1) * 2
   DOEND
   ELSE IF(M7 .EQ. 900 .OR. M7 .EQ. 901) DO
               M6(3) = M6(2) / M6(1)
               RESULT = FCT / (M6(1) * M6(2))
               IF(M7 .EQ. 901) DO
                     FCT = FCT / 2
                     IF(MA(1) .EQ. 1) DO
                           FCT = -FCT
                           I = M6(2)
                           M6(2) = M6(1)
                           M6(1) = I
                     DOEND
               DOEND
   DOEND
   ELSE DO
         RESULT = FCT / 2
         PARM = FCT / 2
         IF(MA(1) .EQ. 1) PARM = -PARM
   DOEND
   STATUS(0) = 33
END
```

```
                PROCEDURE SQRT0(M,N,I,A,B,C,D3)

THIS PROCEDURE IDENTIFIES ALL THE POSSIBLE CASES OF
INTEGRAND THAT INVOLVES ONLY SQRT,WHEN A FORM F'(X)/SQRT(F(X))
OR F'(X)*SQRT(F(X)) IS DISCOVERED,THE INTEGRAND WILL BE INTEGRATED
DIRECTLY, OTHERWISE THE GLOBAL VARIABLE STATUS WILL BE SET TO
THE VALUE THAT IS ASSOCIATED WITH THE APPROPRIATE PROCEDURE
WHICH CAN HANDLE THE INTEGRATION OF THE INTEGRAND

          STATUS(0)     PROCEDURE     SITUATION
            33                        INTEGRATION COMPLETE
            100             SQRT1     POWER OF SQRT IS 1
                                      ARGUMENT OF SQRT IS A*X+B
            101             SQRT2     ARGUMENT OF SQRT IS
                                      A*X**2+B*X
            102             SQRT3     ARGUMENT OF SQRT IS
                                      A*X**2+B*X+C
            104             RATFCT    POWER OF SQRT IS EVEN
            109             SQRT4     POWER OF SQRT IS NEGATIVE
                                      ARGUMENT OF SQRT IS
                                      (X**2+-B**2) OR (B**2+-X**2)
            110             SQRT5     POWER OF SQRT IS POSITIVE
                                      ARGUMENT OF SQRT IS
                                      (X**2+-B**2) OR (B**2+-X**2)

                    ARGUMENT

     I          THE FORM NUMBER OF THE FORM GIVEN AS BELOWS :
                      I               FORM
                      1               SQRT(X**2+B**2)
                      1               SQRT(B**2+X**2)
                      2               SQRT(X**2-B**2)
                      3               SQRT(B**2-X**2)
     M          THE POWER OF X(SEE THE DESCRIPTION BELOW)
     N          THE POWER OF SQRT(SEE THE DESCRIPTION BELOW)
     D3         SEE THE PROCEDURE MAIN FOR DETAILS
     A,B,C      SEE THE FORMS STATED ABOVE

STMT  NO              DESCRIPTION

 10 - 11        THE FUNCTION NAME SQRT APPEARS MORE THAN ONCE
 12             IDENTIFIES THE FORM F'(X)*SQRT(F(X))**N
 13 - 17        IF IT IS OF THE FORM F'(X)*SQRT(F(X))**N THE INTEGRAL
                IS SQRT(F(X))**(N+2)/(N+2)*2
 18 - 22        DETERMINES IF N IS EVEN IF SO REPLACE
                SQRT(F(X))**N BY F(X)**(N/2)
                SET THE STATOS(0) = 104 TO CALL PROCEDURE RATFCT
 23 - 31        THE FUNCTION HAS THE FORM X**M*SQRT(X**M+-B**2)**N
                OR X**M*SQRT(B**2+-X**2)**N
                OR X**M/SQRT(X**2+-B**2)**N
                OR X**M/SQRT(B**2+-X**2)**N
                SET STATUS(0) = 109 TO CALL SQRT4 IF IT HAS
                ONE OF THE LAST TWO FORMS ELSE SET STATUS(0) = 110
                TO CALL SQRT5
 35 - 37        THE ARGUMENT OF SQRT IS A*X**2+B*X+C
                GET THE VALUES A B AND C
 41 - 53        THE FUNCTION HAS THE FORM 1/(X**2*SQRT(A*X+B))
                OR IT CONTAINS THE FORM SQRT(A*X**2+B*X)*X**M
                OR IT CONTAINS SQRT(A*X**2+B*X+C)**N
                FOR THE FORMER CASE SET STATUS(0) = 100 TO CALL
                PROCEDURE SQRT1 ELSE FOR THE SECOND CASE SET STATUS(0)
                = 101 TO CALL PROCEDURE SQRT2 ELSE SET STATUS(0) = 102 TO
                CALL PROCEDURE SQRT3
 44 - 52        THE ARGUMENT OF SQRT IS A*X**2 SO LET Y = SQRT(A)*X
 59 - 68        IF NOT SOLVED BY THE PREVIOUS STEPS USE SUBSTITUTION
                METHOD
 59 - 65        IF THE DEGREE OF THE VARIABLE IN THE ARGUMENT OF
                SQRT EXCEEDS 1 THAN AFTER THE TRANSFORMATION Y = F(X)
                THE NEW INTEGRAND SHOULD NOT CONTAIN THE VARIABLE
                ELSE RETURN WITH ERROR
 66 - 68        THE ARGUMENT OF SQRT IN THE INTEGRAND IS A*X+B
```

```
      PROCEDURE SQRT0(M,N,I,A,B,C,D3)
          ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1) D1,D2,D3,MB,SER,
                    FCT,PARM,RESULT,M5
          INTEGER A,B,C,M,N,I,M2,LK,LAW,JACK,STATUS
          EXTERNAL M2,M5,MB,LAW,SER,FCT,JACK,PARM,RESULT,STATUS
          ARRAY (6) M2,M5
          ARRAY (0:2) STATUS
          ARRAY MB
          STATUS(1) = 48
          SPLIT(D3,Z(1),D1,D2)
          IF(D2 .NE. 1) GO TO STAR3
          SPLIT(D3/DIFF(MB(1),Z0),Z0,D1,D2)
          IF(D2 .EQ. 1) DO
                STATUS(0) = 33
                SER = MB(1)
                RESULT = RESULT + PARM * D1 * Z(5) ** (N + 2) / (N + 2) * 2
          DOEND
          ELSE IF(IMOD(N,2) == 0) DO
                I = N / 2
                STATUS(0) = 104
                FCT = FCT / Z(1) ** N * MB(1) ** I
          DOEND
          ELSE DO
                     IF(JACK == 14) GO TO STAR1
                     GETPWR(M,0,D3)
                     IF(STATUS(0) .NE. 55) GO TO STAR1
                     B = MB(2)
                     I = JACK - 10
                     IF(N<0) STATUS(0) = 109
                     ELSE STATUS(0) = 110
          DOEND
          RETURN
STAR1:          IF(DEG(ADEN(MB(1)),Z0) .NE. 0 .OR.
                     DEG(MB(1),Z0) .GE. 3) GO TO STAR3
                C = MB(1)(Z0 = 0)
                B = ((MB(1) - C)/Z0)(Z0 = 0)
                A = MB(1)(Z0 = 1) - B - C
                SER = Z0
                GETPWR(M,0,D3)
                IF(STATUS(0) .NE. 55) GO TO STAR3
                IF(M == -2 .AND. N == -1 .AND. A == 0) STATUS(0) = 100
                ELSE IF(C == 0 .AND. A .NE. 0 .AND. B .NE. 0) STATUS(0) = 101
                ELSE IF(A .EQ. 0) GO TO STAR2
                ELSE IF(B == 0 .AND. C == 0) DO
                          STATUS(0) = 33
                          LK = LK + 1
                          M2(LK) = 28
                          M5(LK) = A
                          M = M + N + 1
                          IF(M==0) RESULT=RESULT+PARM*Z(5)*Z(LK+6)**N
                          ELSE RESULT=RESULT+PARM*Z0**M*Z(LK+6)**N/M
                DOEND
                ELSE STATUS(0) = 102
                RETURN
STAR2:    PARM = PARM / D3(Z0 = 1)
STAR3:    SER = MB(1)
          LAW = 4
          STATUS(0) = 104
          IF(DEG(ADEN(MB(1)),Z0) .NE. 0 .OR. DEG(MB(1),Z0) .NE. 1) DO
                FCT = FCT / DIFF(MB(1),Z0)
                SPLIT(FCT,Z0,D1,D2)
                IF(D2 .NE. 1) STATUS(0) = 99
                FCT = FCT(Z(1) = Z0)
                RETURN
          DOEND
          D2 = MB(1)(Z0 = 0)
          D1 = MB(1)(Z0 = 1) - D2
          FCT = FCT(Z0 = (Z0 ** 2 - D2) / D1)(Z(1) = Z0) * 2 / D1 * Z0
      END
```

PROCEDURE SQRT1(A,B)

THIS PROCEDURE INTEGRATES THE FOLLOWING FUNCTION :
1/(X**2*SQRT(A*X+B))

ARGUMENT

A,B	SEE THE FORM STATED ABOVE

STMT NO	DESCRIPTION
8 - 14	THE ARGUMENT OF THE SQRT IS A*X THE INTEGRAL CAN BE OBTAINED DIRECTLY THAT IS , RESULT = -2 / (SQRT(A) * SQRT(X) ** 3 * 3)
9 - 11	SQRT(A) IS STORED IN M2(LK) AND M5(LK)
15	THE FIRST TERM OF THE INTEGRAL
16	PARM CONTAINS THE COEFFICIENT OF THE NEW INTEGRAND
17 - 19	STORE SQRT(A*X+B)
20 - 30	CREATE THE SECOND TERM OF THE INTEGRAL WHEN B < 0
31 - 35	CREATE THE SECOND TERM OF THE INTEGRAL WHEN B > 0

```
  PROCEDURE SQRT1(A,B)
     ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1) M5,SER,PARM,RESULT
     EXTERNAL M2,M5,LK,SER,PARM,RESULT,STATUS
     INTEGER A,B,M,N,M2,LK,STATUS
     ARRAY (6) M2,M5
     ARRAY (0:2) STATUS
     STATUS(0) = 33
     IF(B == 0) DO
          LK = LK + 1
          M2(LK) = 28
          M5(LK) = A
          RESULT = -PARM / (Z(5) ** 3 * 3 * Z(LK + 6)) * 2
          RETURN
     DOEND
     RESULT = RESULT - Z(5) / (B * Z0) * PARM
     PARM = -PARM * A / (2 * B)
     LK = LK + 1
     M2(LK) = 28
     SER = A * Z0 + B
     IF(B < 0) DO
          M5(LK) = -B
          LK = LK + 1
          M2(LK) = 11
          M5(LK) = Z(LK + 7)
          RESULT = RESULT + PARM * 2 * Z(LK + 6) / Z(LK + 5)
          LK = LK + 1
          M2(LK) = 28
          M5(LK) = -SER / B
          RETURN
     DOEND
     M5(LK) = B
     LK = LK + 1
     M2(LK) = 27
     M5(LK) = (Z(5) - Z(LK + 5)) / (Z(5) + Z(LK + 5))
     RESULT = RESULT + PARM * Z(LK + 6) / Z(LK + 5)
  END
```

PROCEDURE SQRT2(M,N,A,B)

THIS PROCEDURE INTEGRATES THE FUNCTION WHICH HAS ANY OF THESE FORMS :

```
1     X**M/SQRT(B*X+A*X**2)
2     X**M*SQRT(B*X+A*X**2)
3     1/(X**M*SQRT(B*X+A*X**2))
4     SQRT(B*X+A*X**2)/X**M
```

ARGUMENT

N	THE POWER OF SQRT(SEE THE FORMS STATED ABOVE)
A,B,M	SEE THE FORMS STATED ABOVE

STMT NO	DESCRIPTION
10	THE POWER OF SQRT(B*X+A*X**2) SHOULD BE 1 OR -1
13 - 21	SIMPLIFY SQRT(B*X+A*X**2) TO SQRT(A)* SQRT(B*X/A-X**2) WHEN A IS NOT -1
26 - 33	INTEGRATION FORMULA FOR M > 0 AND N = 1
34 - 41	INTEGRATION FORMULA FOR M < 0 AND N = 1
42 - 48	INTEGRATION FORMULA FOR M < 0 AND N = -1
49 - 55	INTEGRATION FORMULA FOR M > 0 AND N = -1
56 - 61	THE INTEGRAL OF 1/SQRT(B*X/A-X**2)
62 - 63	THE INTEGRAL OF SQRT(B*X/A-X**2)

```
      PROCEDURE SQRT2(M,N,A,B)
         ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1) M5,SER,PAR,
         PARM,RESULT
         EXTERNAL M2,M5,LK,SER,PARM,RESULT,STATUS
         INTEGER A,B,M,N,I,LK,M2,STATUS
         ARRAY (6) M2,M5
         ARRAY (0:2) STATUS
         STATUS(0) = 99
         STATUS(1) = 38
         IF(N .NE. 1 .AND. N .NE. -1) RETURN
         STATUS(0) = 33
         PAR = 1
         IF(A .NE. -1) DO
               LK = LK + 1
               M2(LK) = 28
               M5(LK) = -A
               PARM = Z(LK + 6) ** M * PARM
               B = -B/ (2 * A)
               SER = -SER / A
               A = 1
         DOEND
         LK = LK + 1
         M2(LK) = 7
         IF(M == 0 .AND. N == -1) GO TO CABL1
         ELSE IF(M == 0 .AND. N == 1) GO TO CABL2
         IF(M > 0 .AND. N == 1) DO
               DO I = M , 1 , -1
                     RESULT = RESULT - Z0 ** (M - 1) / (M + 2) * PAR
                     PAR = PAR * B * (2 * M + 1) / (M + 2)
               DOEND
               RESULT = RESULT * Z(5) ** 3
               GO TO CABL2
         DOEND
         ELSE IF(M < 0 .AND. N == 1) DO
               DO I = M , -1 , 1
                     RESULT = RESULT + PAR / ((3 + M * 2) * B) * Z0 ** M
                     PAR = PAR * (M + 3) / ((2 * M + 3) * B)
               DOEND
               RESULT = RESULT * Z(5) ** 3
               GO TO CABL2
         DOEND
         ELSE IF(M < 0 .AND. N == -1) DO
               DO I = M , -1 , 1
                     RESULT = RESULT + PAR / (B * (1 + 2 * M)) * Z0 ** M
                     PAR = PAR * (M + 1) / ((2* M + 1) * B)
               DOEND
               RESULT = RESULT * Z(5)
         DOEND
         ELSE IF(M > 0 .AND. N == -1) DO
               DO I = M , 1 -1
                     RESULT = RESULT - PAR * Z0 ** (M -1) / M
                     PAR = PAR * B * (2 * M - 1) / N
               DOEND
               RESULT = RESULT * Z(5)
         DOEND
CABL1:   RESULT = (RESULT + 2 * Z(LK + 6)) * PARM
         M5(LK) = Z(LK + 7)
         LK = LK + 1
         M2(LK) = 28
         M5(LK) = Z0 / (2 * B)
         RETURN
CABL2:   M5(LK) = (Z0 - B) / B
         RESULT = (RESULT + (Z0 - B) * Z(5) / 2 + 3 ** 2 * Z(LK + 6) / 2)* PARM
      END
```

```
                PROCEDURE SQRT3(M,N,A,B,C)

THIS PROCEDURE INTEGRATES THE FUNCTION WHICH HAS ANY OF THESE FORMS :

     1.         SQRT(A*X**2+B*X+C)
     2.         1/SQRT(A*X**2+B*X+C)
     3.         X/SQRT(A*X**2+B*X+C)

                     ARGUMENT

     M          THE POWER OF X
     N          THE POWER OF SQRT(N=1 OR N=-1)
     A,B,C      SEE THE FORMS STATED ABOVE

STMT  NO                DESCRIPTION

 13 - 19        WHEN A IS GREATER THAN 0
 20 - 29        WHEN A IS LESS THAN 0
 30 - 33        THE INTEGRAND IS SQRT(A*X**2+B*X+C)
                WHEN A > 0
                THE INTEGRAL IS (2*A*X+B)*SQRT(A*X**2+B*X+C)/(4*A) -
                (B**2-4*A*C)*LN(2*A*X+B+2*SQRT(A)*SQRT(A*X**2+B*X+C)/
                (8*A*SQRT(A))
                WHEN A < 0
                THE INTEGRAL IS (2*A*X-B)*SQRT(A*X**2+B*X+C)/(4*A) +
                (B**2+4*A*C)*ASIN((2*A*X-B)/SQRT(B**2+4*A*C))/
                (8*A*SQRT(A))
 34 - 35        THE INTEGRAND IS 1/SQRT(A**2+B*X+C)
                WHEN A > 0
                THE INTEGRAL IS LN(2*A*X+B+2*SQRT(A)*SQRT(A*X**2+B*X+C))/
                SQRT(A)
                WHEN A < 0
                THE INTEGRAL IS ASIN((2*A*X-B)/SQRT(B**2+4*A*C))/
                SQRT(A)
 36 - 38        THE INTEGRAND IS X/SQRT(A**2+B*X+C)
                WHEN A > 0
                THE INTEGRAL IS SQRT(A*X**2+B*X+C)/A-B*LN(2*A*X+
                B+2*SQRT(A)*SQRT(A*X**2+B*X+C))/(2*A*SQRT(A))
                WHEN A < 0
                THE INTEGRAL IS -SQRT(A**2+B*X+C)/A+B*ASIN((2*A*X-B)/
                SQRT(B**2+4*A*C))/(2*A*SQRT(A))
```

```
PROCEDURE SQRT3(M,N,A,B,C)
   ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1)      &
             MB,M5,SER,PARM,RESULT
   EXTERNAL MB,M2,M5,LK,KEK,SER,PARM,RESULT,STATUS
   ARRAY (0:2) STATUS
   INTEGER A,B,C,M,N,N3,M2,LK,KEK,STATUS
   ARRAY MB
   ARRAY (6) M2,M5
   STATUS(0)=33
   STATUS(1) = 58
   LK=LK+1
   M2(LK)=28
   IF(A>0) DO
         M5(LK)=A
         LK=LK+1
         M2(LK)=27
         M5(LK)=2*A*Z0+B+2*Z(LK+5)*Z(5)
         N3=1
   DOEND
   ELSE DO
         M5(LK)=-A
         LK=LK+1
         M2(LK)=28
         M5(LK)=B**2-4*A*C
         LK=LK+1
         M2(LK)=7
         M5(LK)=-(2*A*Z0+B)/Z(LK+5)
         N3=0
   DOEND
         SER = MB(1)
         IF(N==1.AND.M==0) &
         RESULT=RESULT+PARM*((2*A*Z0+B)*Z(5)/
         (4*A)-(B**2-4*A*C)*Z(LK+6)/(8*A*Z(LK+N3+4)))
         ELSE IF(N==-1.AND.M==0) &
         RESULT=RESULT+PARM*Z(LK+6)/Z(LK+N3+4)
         ELSE IF(N==-1.AND.M==1) &
         RESULT=RESULT+PARM*(Z(5)/A-B*Z(LK+6)/
         (2*A*Z(LK+N3+4)))
         ELSE STATUS(0)=99
END
```

PROCEDURE SQRT4(M,N,A,B)

THIS PROCEDURE INTEGRATES THE FUNCTION WHICH HAS ANY OF THESE FORMS :

FORM	A
X**M*SQRT(B**2+X**2)**N	1
X**M*SQRT(X**2+B**2)**N	1
X**M*SQRT(X**2-B**2)**N	2
X**M*SQRT(B**2-X**2)**N	3

THE VALUE OF A INDICATES THE FORM NUMBER OF THE FORM THE FUNCTION HAS BEFORE THE INTEGRATION BEGINS, THE INTEGRAND IS FIRST TRANSFORMED INTO ONE OF THE FOLLOWING THREE FORMS :

```
F'(X)*SIN(F(X))**M*COS(F(X))**(N+1)
F'(X)*SINH(F(X))**M*COSH(F(X))**(N+1)
F'(X)*COSH(F(X))**M*SINH(F(X))**(N+1)
```

THE INTEGRATION OF EACH OF THESE FORMS IS DONE BY PROCEDURE TRIHY1

ARGUMENT	
M,N,A,B	SEE THE FORMS STATED ABOVE

STMT NO	DESCRIPTION
8 - 13	TRANSFORMS THE INTEGRAND INTO ONE OF THE ABOVE THREE FORMS 3 POSSIBLE TRANSFORMATIONS ARE Y = SIN(X), Y = SINH(X), Y = COSH(X)

```
PROCEDURE SQRT4(M,N,A,B)
   ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1) M5,SER,PARM,RESULT
   INTEGER A,B,M,N,P,LK,M2,KEK,STATUS
   EXTERNAL M2,M5,LK,KEK,SER,PARM,STATUS,RESULT
   ARRAY (0:2) STATUS
   ARRAY (6) M2,M5
   STATUS(0) = 33
   KEK=1
   N=N+1
   IF((M + N) > 0) PARM = PARM * B ** (M + N)
   ELSE IF((M + N) < 0) PARM = PARM / B ** ( - M - N)
   IF(A==3) KEK=0
   IF(A == 2) TRIHY1(N,M)
   ELSE TRIHY1(M,N)
   M=1
   N=2
   P=19
   IF(A==3) P=7
   ELSE IF(A:=2) DO
        M=2
        N=1
        P=20
   DOEND
   SER=SER/B
   LK=LK+1
   M2(LK)=P
   M5(LK)=SER
   RESULT=RESULT(Z(6)=Z(LK+6))
   RESULT=RESULT(Z(M)=SER)
   LK=LK+1
   M2(LK)=N+KEK*12
   M5(LK)=Z(LK+7)
   RESULT=RESULT(Z(N)=Z(LK+6))
   LK=LK+1
   M2(LK)=P
   M5(LK)=SER
END
```

PROCEDURE SQRTS(M,N,A,B)

THIS PROCEDURE INTEGRATES THE FUNCTION WHICH HAS ANY OF THESE FORMS :

FORM	A
X**M/SQRT(X**2+B**2)**N	1
X**M/SQRT(B**2+X**2)**N	1
X**M/SQRT(X**2-B**2)**N	2
X**M/SQRT(B**2-X**2)**N	3

THE VALUE OF A INDICATES THE FORM NUMBER OF THE FORM THE FUNCTION HAS BEFORE THE INTEGRATION BEGINS, THE INTEGRAND IS FIRST TRANSFORMED INTO ONE OF THE FOLLOWING THREE FORMS :

```
F'(X)*SIN(F(X))**M*COS(F(X))**(1-N)
F'(X)*SINH(F(X))**M*COSH(F(X))**(1-N)
F'(X)*COSH(F(X))**M*SINH(F(X))**(1-N)
```

THE INTEGRATION OF EACH OF THESE FORMS IS DONE BY PROCEDURE TRIHY1

ARGUMENT

M,N,A,B - SEE THE FORMS STATED ABOVE

STMT NO	DESCRIPTION
8 - 13	TRANSFORMS THE INTEGRAND INTO ONE OF THE ABOVE THREE FORMS 3 POSSIBLE TRANSFORMATIONS ARE Y = SIN(X), Y = SINH(X), Y = COSH(X) CALL PROCEDURE TRIHY1
14 - 35	SUBSTITUTING BACK INTO THE ORIGINAL VARIABLES

```
PROCEDURE SQRTS(M,N,A,B)
    ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1) M5,SER,PARM,RESULT
    INTEGER A,B,M,N,P,LK,KEK,M2,STATUS
    EXTERNAL RESULT,KEK,LK,M2,M5,PARM,SER,STATUS
    ARRAY (0:2) STATUS
    ARRAY (6) M2,M5
    STATUS(0)=33
    N=N+1
    IF((M + N) > 0) PARM = PARM * B ** (M + N)
    ELSE IF((M + N) < 0) PARM = PARM / B ** ( - M - N)
    KEK=1
    IF(A==3) KEK=0
    IF(A == 2) TRIHY1(N,M)
    ELSE TRIHY1(M,N)
    M=1
    N=2
    P=19
    IF(A==3) P=7
    ELSE IF(A==2) DO
         M=2
         N=1
         P=20
    DOEND
    SER=SER/B
    LK=LK+1
    RESULT=RESULT(Z(6)=Z(LK+6))
    M2(LK)=P
    M5(LK)=SER
    RESULT=RESULT(Z(M)=SER)
    LK=LK+1
    M2(LK)=N+12*KEK
    M5(LK)=Z(LK+7)
    RESULT=RESULT(Z(N)=Z(LK+6))
    LK=LK+1
    M2(LK)=P
    M5(LK)=SER
END
```

PROCEDURE RATFCT

THIS PROCEDURE INTEGRATES THE FUNCTION WHICH IS THE QUOTIENT OF TWO POLYNOMIALS AND IT SATISFIES ONE OF THESE CONDITIONS :

1. THE FUNCTION CAN BE EXPRESSED AS F'(X)/F(X)
2. THE DENOMINATOR OF THE FUNCTION IS A SINGLE TERM
3. THE DENOMINATOR HAS THE DEGREE OF THE VARIABLE LESS THAN 3

IF THE FUNCTION CANNOT BE COMPLETELY INTEGRATED BY RATFCT THEN PROCEDURE RATFUN WILL BE CALLED

STMT NO	DESCRIPTION
12 - 14	WHEN THE DENOMINATOR OF THE INTEGRAND IS INDEPENDENT OF Z0 THE INTEGRAL IS OBTAINED BY USING PROCEDURE PINT(ALTRAN BUILTIN ROUTINE)
16 - 19	IF THE DEGREE OF THE VARIABLE IN THE NUMERATOR IS NOT LESS THAN THAT OF THE DENOMINATOR DIVIDE TO OBTAIN THE QUOTIENT AND REMAINDER THE QUOTIENT IS A POLYNOMIAL AND SO IS INTEGRATED DIRECTLY
21	IF THE REMAINDER IS 0 THE INTEGRATION IS COMPLETE
24 - 30	IF THE INTEGRAND HAS THE FORM F'(X)/F(X) THE INTEGRAL IS LN(F(X))
31 - 35	THE NUMERATOR OF THE UPDATED INTEGRAND SHOULD HAVE A SINGLE TERM OR ITS DEGREE OF Z0 IS LESS THAN 3
37 - 51	WHEN THE NUMERATOR HAS A SINGLE TERM AND THE INTEGRAND IS (A*X**I + B*X**J + ...) / X**K INTEGRATE EACH TERM L * X ** R / X ** K DIRECTLY AND SUM UP ALL THE INTEGRALS
40	GET EACH TERM OF THE INTEGRAND P * X ** R / X**K
42 - 48	NOW INTEGRATE THE CURRENT TERM
42 - 47	IF M EQUAL -1 THE INTEGRAL IS P * LN(Z0)
48	ELSE THE INTEGRAL IS -P / (M - 1) * X ** (M - 1)
50	IF INTEGRATION NOT COMPLETED GO TO 38
52	IF THE INTEGRAND IS NOT IN THE ABOVE CASE PROCEDURE RATFUN WILL BE CALLED

```
      PROCEDURE RATFCT
         ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1) D1,D2,D3,DD,
                                 M5,FCT,PARM,RESULT
         INTEGER M,N,N1,N2,M2,LK,LAW,JACK,STATUS
         EXTERNAL M2,M5,LK,LAW,FCT,JACK,PARM,STATUS,RESULT
         ALTRAN ALGEBRAIC LCPO,LTPO,QUOPO
         ARRAY(0:2) STATUS
         ARRAY(6) M2,M5
         STATUS(1) = 51 + LAW
         STATUS(0) = 33
         JACK = 5
         D1 = ADEN(FCT,D2)
         N1 = DEG(D1,Z0)
         IF(N1 == 0) RESULT = RESULT + PARM * PINT(FCT,Z0)
         ELSE DO
              N2 = DEG(D2,Z0)
              IF(N2 < N1) GO TO WOOD1
              D3 = QUOPO(EXPAND(D2),EXPAND(D1),Z0,D2)
              RESULT = RESULT + PARM * PINT(D3,Z0)
              FCT = D2 / D1
              IF(FCT == 0) RETURN
WOOD1:        D3 = D2 / DIFF(D1,Z0)
              SPLIT(D3,Z0,D3,D2)
              IF(D2 .EQ. 1) DO
                   LK = LK + 1
                   RESULT = RESULT + PARM * D3 * Z(LK + 6)
                   M2(LK) = 27
                   M5(LK) = D1
                   RETURN
              DOEND
              STATUS(0) = 99
              D1 = ADEN(FCT,DD)
              M = DEG(D1,Z0)
              N = NTRM(D1)
              IF(M > 2 .AND. N > 1) RETURN
              STATUS(0) = 33
              IF(N == 1) DO
WOOD2:             D2 = LTPO(EXPAND(DD))
                   DD = DD - D2
                   D2 = D2 / D1
                   M = DEG(D2,Z0)
                   IF(M == -1) DO
                        LK = LK + 1
                        M2(LK) = 27
                        M5(LK) = Z0
                        RESULT = RESULT + PARM * Z(LK + 6) * D2(Z0 = 1)
                   DOEND
                   ELSE RESULT = RESULT + PARM * Z0 ** (M + 1) / (M + 1) *
                   D2(Z0 = 1)
                   IF(DD .NE. 0) GO TO WOOD2
              DOEND
              ELSE STATUS(0) = 55
         DOEND
      END
```

PROCEDURE RATFUN

THIS PROCEDURE INTEGRATES THE FUNCTION WHICH HAS THE FOLLOWING FORM :
(A*X+B)/(C*X**2+D*X+E)
(THE FUNCTION CANNOT BE EXPRESSED AS F'(X)/F(X))

STMT NO	DESCRIPTION
9 - 13	GET THE VALUES OF A B C D AND E IN THE INTEGRAND
14 - 20	IF A NOT EQUAL 0 THE FIRST PART OF THE INTEGRAL IS A*LN(C*X**2+D*X+E)/(2*C) THE REMAINING INTEGRAND IS (B-A*D/(2*C))/C / (X**2+D/C*X+E/C)
23	LET A = (E/C)-(D/(2*C))**2
24	IF A EQUAL 0 THE INTEGRAL OF THE NEW INTEGRAND IS -1/(Z0+D/(2*C))
25 - 43	A NOT EQUAL 0
28 - 31	IF A < 0 THE INTEGRAL IS (B-A*D/(2*C))/(C*SQRT(-A)) * ATAN((Z0+D/(2*C))/SQRT(-A)
32 - 35	A = 1 THE INTEGRAL IS THE SAME AS ABOVE BUT SET SQRT(-A) = 1
36 - 42	A > 0 THE INTEGRAL IS (B-A*D/(2*C))/(C*SQRT(A)) * ATANH((Z0+D/(2*C))/SQRT(A)

```
PROCEDURE RATFUN
   ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1) M5,
                        FCT,TEMP1,TEMP2,PARM,RESULT
   EXTERNAL M2,M5,LK,LAW,KEK,FCT,PARM,RESULT
   INTEGER B,C,D,E,M2,LK,KEK,LAW
   RATIONAL A
   ARRAY (6) M2,M5
   TEMP1=ADEN(FCT,TEMP2)
   B=TEMP2(Z0=0)
   A=TEMP2(Z0=1)-B
   E=TEMP1(Z0=0)
   D=((TEMP1-E)/Z0)(Z0=0)
   C=TEMP1(Z0=1)-E-D
   IF(A.NE.0) DO
         LK=LK+1
         M2(LK)=27
         M5(LK)=TEMP1
         RESULT=RESULT+PARM*A*Z(LK+6)/(2*C)
         PARM=PARM*(B-A*D/(2*C))/C
   DOEND
   ELSE PARM=PARM*B/C
   TEMP1=(Z0+D/(2*C))
   A=(E/C)-(D/(2*C))**2
   IF(A==0) RESULT=RESULT-PARM/TEMP1
   ELSE DO
         LK=LK+1
         M2(LK)=11
         IF(A<0) DO
               A=-A
               M2(LK)=23
         DOEND
         IF(A==1) DO
               M5(LK)=TEMP1
               RESULT=RESULT+PARM*Z(LK+6)
         DOEND
         ELSE DO
               M5(LK)=TEMP1/Z(LK+7)
               LK=LK+1
               M2(LK)=28
               M5(LK)=A
               RESULT=RESULT+PARM*Z(LK+5)/Z(LK+6)
         DOEND
   DOEND
END
```

PROCEDURE ARCFCT(M,N,I,A,B,N1,D3)

THIS PROCEDURE INTEGRATES THE FUNCTION WHICH HAS THE FOLLOWING FORM :

F'(X)*F(X)**M*OP(F(X))
WHERE OP IS ANY INVERSE TRI. OR HYP. FUNCTION

THE INTEGRATION IS FIRST PERFORMED BY INTEGRATION BY PARTS,
THUS OBTAINING THE FIRST TERM OF THE INTEGRAL, THE NEW INTEGRAND
IS THEN TRANSFORMED INTO ONE OF THE FOLLOWING FORMS :

FORM VALUE I	EXPRESSION
1	X**M/SQRT(X**2+B**2)**N
1	X**M/SQRT(B**2+X**2)**N
2	X**M/SQRT(X**2-B**2)**N
3	X**M/SQRT(B**2-X**2)**N
4	X**(M+1)/(1-X**2)
5	X**(M+1)/(1+X**2)

THE FIRST FOUR FORMS ARE HANDLED BY PROCEDURE SQRT4
AND THE REST ARE HANDLED BY PROCEDURE RATFCT

ARGUMENT

A	DUMMY
I	THE INTEGER CORRESPONDING TO OP(SEE ABOVE) ON RETURN I CONTAINS THE FORM VALUE(SEE ABOVE)
N1,D3	SEE THE PROCEDURE MAIN FOR DETAILS
M,N,B	SEE THE NEW FORMS ABOVE THESE VARIABLES ARE PASSED TO PROCEDURE SQRT4

STMT NO	DESCRIPTION
12	THE DEGREE IN THE OP SHOULD BE EXACTLY ONE
13 - 14	OP SHOULD APPEAR ONLY ONCE IN THE INTEGRAND
15 - 16	FORM CHECKING FOR F'(X)*F(X)**M*OP(F(X))
17 - 21	THE FIRST TERM OF THE EVENTUAL INTEGRAL IS OBTAINED USING INTEGRATION BY PARTS METHOD
25 - 29	FOR OP EQUAL ASIN OR ACCS OR ASECH THE INTEGRAND IS TRANSFORMED INTO FORM 3
30 - 36	FOR OP EQUAL ASINH OR ACSCH THE INTEGRAND IS TRANSFORMED INTO FORM 1
37 - 41	FOR OP EQUAL ACSC OR ASEC OR ACOSH THE INTEGRAND IS TRANSFORMED INTO FORM 2
42 - 47	THE REST IS TRANSFORMED INTO FORM 4 OR FORM 5
45	THE INTEGRAND IS TRANSFORMED INTO FORM 5
46	THE INTEGRAND IS TRANSFORMED INTO FORM 4

```
PROCEDURE ARCFCT(M,N,I,A,B,N1,D3)
   ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1) D1,D2,D3,M5,
                          SER,FCT,PARM,RESULT
   INTEGER A,B,M,N,I,N1,M2,MA,LK,KEK,STATUS
   EXTERNAL M2,M5,MA,LK,FCT,KEK,SER,PARM,RESULT,STATUS
   ARRAY (0:2) STATUS
   ARRAY(6) M2,M5
   ARRAY MA
   STATUS(0)=99
   STATUS(1) = 44
   I=MA(1)
        IF(N1.NE.1) RETURN
        SPLIT(D3,Z(1),D1,D2)
        IF(D2.NE.1) RETURN
        GETPWR(M,0,D3)
        IF(STATUS(0).NE.55) RETURN
        PARM = PARM / (M + 1)
        LK = LK + 1
        RESULT = PARM * SER ** (M + 1) * Z(LK + 6)
        M2(LK) = I
        M5(LK) = SER
   N=-1
   B=1
   STATUS(0)=109
   IF(I==7.OR.I==8.OR.I==22) DO
        A=2
        IF(I.NE.22) M=M+1
        IF(I==7) PARM=-PARM
   DOEND
   ELSE IF(I==19.OR.I==21) DO
        A=1
        IF(I==19) DO
             PARM=-PARM
             M=M+1
        DOEND
   DOEND
   ELSE IF(I==9.OR.I==10.OR.I==20) DO
        A=3
        IF(I.NE.9) PARM=-PARM
        IF(I==20) M=M+1
   DOEND
   ELSE DO
        STATUS(0)=104
        IF(I.NE.12) PARM=-PARM
        IF(I==11.OR.I==12) FCT=SER**(M+1)/(1+SER**2)
        ELSE IF(I==23.OR.I==24) FCT=SER**(M+1)/(1-SER**2)
   DOEND
   I=A
END
```

```
                PROCEDURE GETPWR(I,U,D3)

THIS PROCEDURE COMPUTES THE VALUE OF I IN THE "FUNCTION"

                F'(X)*F(X)**I*OP(F(X))
          WHERE OP IS ANY FUNCTION NAME ACCEPTABLE TO THE PACKAGE.

IF THE FORM STATED ABOVE CAN NOT BE OBTAINED IT RETURNS AN ERROR FLAG.

                    ARGUMENTS

     I          SEE THE FORM STATED ABOVE
     U          DUMMY INTEGER
     D3         SEE THE PROCEDURE MAIN FOR DETAILS

STMT  NO             DESCRIPTION

 9              DIVIDE OUT F'(X) FROM THE FUNCTION
 10 - 12        I1 = DEGREE OF X IN NUMERATOR OF D3
                I2 = DEGREE OF X IN DENUMINATOR OF D3
 13 - 15        J1 = DEGREE OF X IN NUMERATOR OF SER
                J2 = DEGREE OF X IN DENUMINATOR OF SER
 16             IF SER IS A CONSTANT RETURN WITH ERROR
 17             IF D3 IS A CONSTANT I IS SET TO 0
 19 - 36        I1 > 0 AND J1 > 0
                TWO POSSIBLE VALUS OF I ARE TESTED
                I = I1 / J1 AND I = -I1 / J2
 22             IF BOTH I1 / J1 AND I1 / J2 NOT INTEGER RETURN
                WITH ERROR
 23 - 33        IF I1 / J2 IS AN INTEGER TEST WITH I = -I1 / J2
 27 - 32        IF I IS FOUND RETURN WITH STATUS(0) = 55
 34             IF NOT FOUND AND I1 / J1 IS NOT AN INTEGER
                RETURN WITH ERROR
 37 - 46        EITHER I1 OR J1 IS ZERO
                IF I1 AND J1 BOTH EQUAL 0 TEST WITH I = I2 / J2
                ELSE IF I1 EQUAL 0 TEST WITH I = -I2 / J1
                ELSE TEST WITH I = -I1 / J2
 49 - 50        THE FORM IS NOT RIGHT RETURN WITH ERROR
 52             THE FORM IS RIGHT SET STATUS(0) = 55
```

```
PROCEDURE GETPWR(I,U,D3)
    ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1) &
              D1,D2,D3,SER,PARM
    EXTERNAL SER,PARM,STATUS
    INTEGER I,J,U,I1,I2,J1,J2,STATUS
    ARRAY (0:2) STATUS
    VALUE D3
    STATUS(1) = 50
    D3 = D3 / DIFF(SER,Z0)
    I1 = DEG(ANUM(D3),Z0)
    I = DEG(D3,Z0)
    I2 = I1 - I
    J1 = DEG(ANUM(SER),Z0)
    J = DEG(SER,Z0)
    J2 = J1 - J
    IF(J1 == 0 .AND. J2 == 0) RETURN
    IF(I1 == 0 .AND. I2 == 0) I = 0
    ELSE DO
         IF(I1 > 0 .AND. J1 > 0) DO
              I = IMOD(I1,J1)
              IF(J2 .NE. 0 .AND. I1 > J2) J=IMOD(I1,J2)
              IF(I .NE. 0 .AND. J .NE. 0) RETURN
              IF(J .EQ. 0) DO
                   J = -I1 / J2
                   D1 = D3 / SER ** J
                   SPLIT(D1,Z0,D1,D2)
                   IF(D2 .EQ. 1) DO
                        I = J
                        STATUS(0) = 55
                        PARM = PARM * D1
                        RETURN
                   DOEND
              DOEND
              ELSE IF(I .NE. 0) RETURN
              I = I1 / J1
         DOEND
         ELSE DO
              IF(I1 == 0) I = I2
              ELSE I = I1
              IF(J1 == 0) J = J2
              ELSE J = J1
              I2 = IMOD(I,J)
              IF(I2 .NE. 0) RETURN
              I = I / J
              IF(I1 .NE. 0 .OR. J1 .NE. 0) I = -I
         DOEND
         D3 = D3 / SER ** I
    DOEND
    SPLIT(D3,Z0,D1,D2)
    IF(D2 .NE. 1) RETURN
    PARM = PARM * D1
    STATUS(0) = 55
END
```

```
                PROCEDURE EXPFCT(M,D3)

THIS PROCEDURE IDENTIFIES THE FUNCTION WHICH ONLY INVOLVES EXP

THERE ARE TWO CASES WHICH ARE ACCEPTABLE TO THE PACKAGE :
     1.         F'(X)*F(X)**M*EXP(F(X))
     2.         THE FUNCTION IS THE QUOTIENT OF TWO EXPRESSIONS
                AT LEAST ONE OF THEM CONTAINS THE FUNCTION NAME EXP

FOR THE LATTER CASE THE ARGUMENTS MUST BE EITHER F(X) OR -F(X)
AND THERE MUST NOT BE ANY VARIABLE Z0 PRESENT IN THE FUNCTION
AS A SINGLE ENTITY
THE FUNCTION OF THE FIRST CASE IS INTEGRATED USING INTEGRATION BY PARTS
THE FUNCTION OF THE SECOND CASE IS FIRST TRANSFORMED BY EXP(F(X)) = Z0
AND THEN INTEGRATED BY PROCEDURE RATFCT

                     ARGUMENT

     M          THE VALUE IN THIS FORM
                F'(X)*F(X)**M*EXP(F(X))
     D3         SEE THE PROCEDURE MAIN FOR DETAILS

STMT  NO             DESCRIPTION


 9               WHEN M7 = -1 THE ARGUMENT OF EXP IS IN FACT LN(A)*F(X)
                 WHERE A IS THE INTEGER IN THE FORM A ** F(X)
 12 - 24         THE FUNCTION HAS FORM 2
 13 - 16         THE FUNCTION SHOULD NOT CONTAIN Z0
 17 - 18         THE ARGUMENTS MUST ALL BE THE SAME OR
                 EACH OF THEM IS EITHER F(X) OR -F(X)
 19 - 23         TRANSFORM THE FUNCTION BY EXP(F(X)) = Z0
 22              SET STATUS(0) = 104 TO CALL PROCEDURE RATFCT
 26 - 34         THE FUNCTION HAS FORM 1


  PROCEDURE EXPFCT(M,D3)
     ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1) D1,D2,D3,D4,
                              MB,FCT,SER,PARM,RESULT
     EXTERNAL MB,M3,M7,FCT,LAW,SER,PARM,STATUS,RESULT
     INTEGER M,N,I,M3,M7,LAW,STATUS
     ARRAY (0:2) STATUS
     ARRAY MB,M3
     STATUS(1) = 39
          IF(M7 == -1) SER = SER * ZZ(1)
          SPLIT(D3,Z(1),D1,D2)
          SPLIT(D3,Z(2),D1,D4)
          IF(D2.NE.1.OR.D4.NE.1) DO
               IF(M3(1)>2) RETURN
               FCT=FCT/DIFF(SER,Z0)
               SPLIT(FCT,Z0,D1,D2)
               IF(D2.NE.1) RETURN
               IF(M3(1)==1) MB(2)=-MB(1)
               IF(MB(1)+MB(2).NE.0) RETURN
               FCT=FCT(Z(1)=Z0)(Z(2)=1/Z0)
               FCT = FCT / Z0
               LAW=2
               STATUS(0) = 104
               RETURN
          DOEND
          IF(M3(1) .LE. 1) GETPWR(M,0,D3)
          IF(STATUS(0) == 55) DO
               I = -1
               DO N = M , 1 , -1
                    RESULT = RESULT + PARM * Z(6) ** N
                    PARM = PARM * N * I
               DOEND
               RESULT = (RESULT + PARM) * Z(3)
               STATUS(0) = 33
          DOEND
  END
```

```
                PROCEDURE EXP1(I,N1,N2,D3)

THIS PROCEDURE IDENTIFIES THE FUNCTION WHICH HAS ANY OF THESE FORMS :

     1.         F'(X)*OP(F(X))**N1*EXP(F(X))**N2
     2.         E(X)*OP1(F(X))**N1*OP2(G(X))**N2*EXP(H(X))**N3
                (WHERE OP,OP1,OP2 ARE TRI. OR HYP. FUNCTIONS)

                     ARGUMENT

     I          THE INTEGER CORRESPONDING TO OP(SEE THE FORMS STATED ABOVE
                SEE ALSO "DELAY" IN PROCEDURE MAIN)
     D3         SEE THE PROCEDURE MAIN FOR DETAILS
     N1,N2      SEE THE FORMS STATED ABOVE


STMT  NO              DESCRIPTION

 11               N3 IS THE POWER OF EXP
 13 - 16          THE INTEGRAND THAT INVOLVES EXP AND AT LEAST A
                  TRIGONOMETRIC OR HYPERBOLIC FUNCTION SHOULD
                  HAVE THE FORM AS STATED ABOVE
 18 - 20          THE POWER OF EXP IN THE INTEGRAND IS EXPECTED
                  TO BE 1
 21 - 31          THE INTEGRAND CONTAINS SIN(F(X))**N*COS(F(X))**N
                  OR SINH(F(X))**N*COSH(F(X))**N AND CAN BE SIMPLIFIED
                  BY DOUBLE ANGLE RULES
 35               EXP HAS THE VALUE A IN EXP(A*F(X)) WHERE F(X) IS
                  THE ARGUMENT OF THE TRIGONOMETRIC OR HYPERBOLIC FUNCTION
 36 - 56          THE INTEGRAND HAS THE SECCND FORM AS STATED ABOVE
                  THE INTEGRAND IS THUS SPLIT INTO TWO PARTS
                  THE INFORMATION REGARDING THESE PARTS ARE STORED HERE
 40               THE ARGUMENT OF THE TRIGONOMETRIC OR HYPERBOLIC FUNCTION
                  THAT APPEARS IN THE SECOND PART OF THE INTEGRAND
 41 - 42          AFTER THE TRANSFORMATION Y = F(X) NO SIMPLE VARIABLE
                  IS IN ANY PART OF THE INTEGRAND
 43 - 55          ADJUST COEFFICIENTS
 51               IF M7 = -1 THE FUNCTION CONTAINS A ** F(X)
 58 - 59          AFTER THE TRANSFORMATION Y = F(X) THE SIMPLE VARIABLE IS NOT
                  IN THE FUNCTION ANY LCNGER
                  CTHERWISE TERMINATE THE INTEGRATION
 62               ADJUSTMENT
```

```
PROCEDURE EXP1(I,N1,N2,D3)
   ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1) D1,D2,D3,
   MB,SER,EXP,PARM,OSER,OEXP,OPARM,FCT
   EXTERNAL MA,MB,M3,M7,SER,EXP,PARM,OSER,OEXP,OPARM,FCT,KEK,
        JACK,STATUS,DELAY
   INTEGER A,I,N1,N2,N3,MA,M3,M7,KEK,JACK,STATUS,DELAY
   ARRAY (0:2) STATUS
   ARRAY MA,MB,M3
   JACK = 6
   STATUS(1) = 40
   N3 = DEG(D3,Z(3))
   D3 = D3 / Z(3) ** N3
   DO A = 1 , 3
        SPLIT(D3,Z(A),D1,D2)
        IF(D2 .NE. 1) RETURN
   DOEND
   STATUS(1) = 41
   A = M3(1) + M3(2) + M3(3)
   IF(((A==3 .OR. A==102) .AND. N3 .NE. 1) .OR.
   (A==101 .AND. N2 .NE. 1)) RETURN
   IF (A==3 .AND. (MB(1)-MB(2) .EQ. 0)) DO
        STATUS(1) = 57
        IF(N1 .NE. N2) RETURN
        M3(2) = 99
        MA(1) = 1 + KEK * 12
        MB(2) = MB(3)
        SER = SER * 2
        MB(1) = MB(1) * 2
        IF(N1 > 0) PARM = PARM / 2 ** N1
        ELSE IF(N1 .NE. 0) PARM = PARM * 2 ** N1
   DOEND
   IF(A .NE. 3) M3(2) = 99
   FCT = FCT(Z(2) = 1)(Z(3) = 1)
   I = MA(1)
   EXP = MB(2) / MB(1)
   IF(M3(1) + M3(2) .NE. 100) DO
        STATUS(1) = 42
        IF(N1 .NE. 1 .OR. N2 .NE. 1) RETURN
        STATUS(1) = 43
        OSEP = MB(1) - MB(2)
        SPLIT(D3/DIFF(OSER,Z0),Z0,D1,D2)
        IF(D2 .NE. 1) RETURN
        OPARM = D1 / 2
        I = KEK * 12 + 2
        IF(A == 3) I = 1 + KEK * 12
        IF(MA(1) == 11 .AND. M3(1) == 2) OPARM = -OPARM
        IF((MA(1) - KEK * 12) == 2 .AND. (MA(2) - KEK * 12) ==
        1) OPARM = -OPARM
        DELAY = 1
        OEXP = MB(3) / OSER
        IF(M7 == -1) OEXP = OEXP * ZZ(1)
        PARM = 1 / 2
        SER = MB(1) + MB(2)
        EXP = MB(3) / SER
        IF(MA(1) == 1 .AND. M3(1) == 2) PARM = -PARM
   DOEND
   STATUS(1) = 43
   SPLIT(D3 / DIFF(SER,Z0),Z0,D1,D2)
   IF(D2 .NE. 1) RETURN
   STATUS(0) = 55
   PARM = PARM * D1
   IF(M7 == -1) EXP = EXP * ZZ(1)
END
```

PROCEDURE EXP2(I,M)

THIS PROCEDURE INTEGRATES THE FUNCTION WHICH HAS ANY OF THESE FORMS :

1. F'(X)*OP(F(X))**M*EXP(A*F(X))
2. E(X)*OP1(F(X))*OP2(G(X))*EXP(H(X))
(WHERE OP,OP1,OP2 ARE TRI. OR HYP. FUNCTIONS)

ARGUMENT

M — SEE THE FORMS STATED ABOVE(SEE ALSO "DELAY" IN PROCEDURE MAIN)

I — THE INTEGER CORRESPONDING TO OP(SEE THE FORMS STATED ABOVE)

STMT NO	DESCRIPTION
14	IF DELAY EQUAL 1 GO TO 32
15 - 25	DELAY EQUAL 0 THE FORM IS F'(X)*OP(F(X))**M*EXP(A*F(X)) INITIALIZATION OF VARIABLES
26	IF THE DENOMINATOR OF THE CURRENT TERM OF THE INTEGRAL IS 0 RETURNS
27 - 29	ADD THE CURRENT TERM OF THE INTEGRAL TO RESULT
31	IF M > 1 CALCULATE THE NEXT TERM OF THE INTEGRAL
32	IF M = 0 THE LAST TERM OF THE INTEGRAL IS EXP(A*X)/A
33 - 40	FOR M = 1 THE INTEGRAL OF THE INTEGRAND IS OBTAINED DIRECTLY
42 - 48	WHEN DELAY = 1 CALCULATE THE SECOND INTEGRAL DIRECTLY
49 - 52	THE FINAL RESULT IS COMPLETED BY MULTIPLYING WITH EXP(A*X)

```
      PROCEDURE EXP2(I,M)
         ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1) &
                              SER,OSER,PARM,OPARM,MS,PAR,EXP,OEXP,TEMP,RESULT
         INTEGER I,J,K,L,M,P,II,LK,M2,KEK,DELAY,STATUS
         EXTERNAL M2,MS,LK,KEK,EXP,SER,PARM,OEXP,OSER,OPARM,DELAY,STATUS,RESULT
         ARRAY (0:2) STATUS
         ARRAY (6) M2,MS
         STATUS(0) = 99
         STATUS(1) = 19
         P = I
         I = I - KEK * 12
         IF(I == 1) J = I + 1
         ELSE J = I - 1
         IF(DELAY == 1) GO TO EXON2
         OEXP = EXP
         K = 1
         IF(M < 2) GO TO EXON2
         II = 2 * (P - 1) + KEK + 1
         L=-1
         K=-1
         IF(II.EQ.1) L=1
         ELSE IF(II.EQ.3) DO
              L=1
              K=1
         DOEND
EXON1:   IF((EXP ** 2 + L * M ** 2) == 0) RETURN
         TEMP=Z(I)**(M-1)*(EXP*Z(I)+K*M*Z(J))/(EXP**2+L*M**2)
         RESULT = RESULT + PARM * TEMP
         PARM = PARM * (M * (M - 1)) / (EXP ** 2 + L * M ** 2)
         M = M - 2
         IF(M > 1) GO TO EXON1
EXON2:   IF(M < 1) RESULT = RESULT + PARM / EXP
      ELSE DO
         II = -1
         L = 1
         IF(KEK == 1) L = -1
         IF(P == 2 .AND. KEK == 0) II = 1
         IF((EXP ** 2 + L) == 0 .OR. (OEXP ** 2 + L) == 0) RETURN
         RESULT=RESULT+PARM*(EXP*Z(I)+(II*Z(J)))/(EXP**2+L)
      DOEND
         IF(DELAY == 0) GO TO EXON3
              LK = LK + 2
              M2(LK-1) = I
              M2(LK) = J
              MS(LK-1) = OSER
              MS(LK) = OSER
              RESULT = RESULT + OPARM * (OEXP * Z(LK + 5) +
                       (II * Z(LK + 6))) / (OEXP ** 2 + L)
EXON3:   LK = LK + 1
         RESULT = RESULT * Z(6 + LK)
         M2(LK) = 26
         MS(LK) = EXP * SER
         STATUS(0) = 33
      END
```

```
                PROCEDURE HYPTRI(I,M,OM,N1,N2,D3)

THIS PROCEDURE IDENTIFIES THE FUNCTION WHICH INVOLVES
EXP,SIN,COS,SINH,COSH,WITH AT LEAST ONE FUNCTION NAME BEING
SIN,COS,SINH,OR COSH
THE SET OF FORMS THAT IS IDENTIFIED HERE INCLUDES :

     1.         F'(X)*F(X)**M*OP(F(X))**N
     2.         E(X)**M*OP1(F(X))*OP2(G(X))
     3.         F'(X)*OP1(F(X))**N1*OP2(F(X))**N2
                (WHERE OP,OP1,OP2 ARE TRI. OR HYP. FUNCTIONS)

                     ARGUMENT

     I          THE INTEGER CORRESPONDING TO OP(SEE THE FORMS STATED
                ABOVE AND SEE ALSO "DELAY" IN THE PROCEDURE MAIN)
     D3         SEE THE PROCEDURE MAIN FOR DETAILS
     M,OM       SEE THE FORMS STATED ABOVE
     N1,N2      SEE THE FORMS STATED ABOVE

STMT  NO             DESCRIPTION

 11              IF NOT THE ABOVE FORM GO TO 33
 12              IF ARGUMENTS NOT ALL EQUAL GO TO 33
 13 - 26         THE INTEGRAND HAS THE ABOVE FORM PROCEDURE
                 TRIHY1 WILL BE CALLED
 27 - 32         THE INTEGRAND CONTAINS  SIN(F(X))**N*COS(F(X))**N
                 OR SINH(F(X))**N*COSH(F(X))**N AND CAN BE SIMPLIED
                 BY THE DOUBLE ANGLE RULES
                 FOR EXAMPLE SIN(2*X)=2*SIN(X)*COS(X)
 33              THE INTEGRAND IS EXPECTED TO HAVE THE FORM
                 F'(X)*F(X)**M*OP(F(X))**N
 34 - 54         THE INTEGRAND IS EXPECTED TO HAVE THE FORM
                 E(X)*OP(F(X))*OP(G(X))
 36              FOR THESE FORMS THE POWERS OF THE OP'S MUST BE 1
```

```
      PROCEDURE HYPTRI(I,M,OM,N1,N2,D3)
         ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1) D1,D2,D3,
            MB,SER,PARM,OSER,OPARM,FCT,RESULT
         EXTERNAL MA,MB,M1,M3,
         KEK,FCT,SER,PARM,OSER,OPARM,JACK,STATUS,RESULT,DELAY
         INTEGER I,M,OM,N1,N2,MA,M1,M3,KEK,JACK,STATUS,DELAY
         ARRAY (0:2) STATUS
         ARRAY MA,MB,M1,M3
         JACK = 2
         IF(M1(1) + M3(1) .EQ. 2 .AND. N1 < 0) GO TO WORH1
         IF(M3(1) .NE. 1 .OR. M3(2) .NE. 1) GO TO WORH2
         IF(MB(2) - MB(1) .NE. 0) GO TO WORH2
WORH1:   IF(N1 .NE. N2) DO
               STATUS(1) = 43
               SPLIT(D3 / DIFF(SER,Z0),Z0,D1,D2)
               IF(D2 .NE. 1) RETURN
               IF(MA(1) == 2) DO
                     JACK = N1
                     N1 = N2
                     N2 = JACK
               DOEND
               JACK = 1
               PARM = PARM * D1
               STATUS(0) = 55
               RETURN
         DOEND
         MA(1) = KEK * 12 + 1
         SER = SER * 2
         M3(2) = 99
         IF(N1 > 0) PARM = PARM / 2 ** N1
         ELSE IF(N1 .NE. 0) PARM = PARM * 2 ** N1
         FCT = FCT(Z(2) = 1)
WORH2:   IF(M3(1) + M3(2) .EQ. 100) I = MA(1)
         ELSE DO
               STATUS(1) = 42
               IF(N1 .NE. 1 .OR. N2 .NE. 1) RETURN
               SER = MB(1) - MB(2)
               DELAY = 1
               PARM = 1 / 2
               STATUS(0) = 99
               GETPWR(M,0,D3)
               IF(STATUS(0) .NE. 55) RETURN
               OSER = SER
               OPARM = PARM
               OM = M
               I = KEK * 12 + 2
               IF(M3(1) .EQ. 1 .AND. M3(2) .EQ. 1) I = KEK * 12 + 1
               IF(MA(1) == 11 .AND. M3(1) == 2) OPARM = -OPARM
               IF((MA(1) - KEK * 12) == 2 .AND. (MA(2) - KEK * 12) ==
               1) OPARM = -OPARM
               SER = MB(1) + MB(2)
               PARM = 1 / 2
               IF(MA(1) == 1 .AND. M3(1) == 2) PARM = -PARM
         DOEND
         STATUS(0) = 99
         GETPWR(M,0,D3)
      END
```

```
                PROCEDURE LOGFCT(A,B,M,N,I,D3)

THIS PROCEDURE IDENTIFIES THE FUNCTION THAT INVOVLES EITHER LN
ALONE OR LN AND SQRT WITH SQRT BEING IN THE ARGUMENT OF LN(SEE THE
FORMS STATED BELOWS) :

          FORMS                    I
                                (FORM NUMBER)
     LN(X-SQRT(X**2-B**2))        6
     LN(X-SQRT(X**2+B**2))        7
     LN(X-SQRT(B**2+X**2))        7
     LN(X+SQRT(X**2-B**2))        8
     LN(X+SQRT(X**2+B**2))        9
     LN(X+SQRT(B**2+X**2))        9

THE FUNCTION IS FIRST INTEGRATED USING INTEGRATION BY PARTS
THE REMAINING INTEGRAND IS THEN TRANSFORMED INTO ONE OF THESE FORMS :

                FORM                      I
                                   (NEW FORM VALUE)
          X**(M+1)/SQRT(X**2+B**2)        1
          X**(M+1)/SQRT(X**2-B**2)        2

                ARGUMENT

     A          DUMMY
     B          SEE THE FORMS STATED ABOVE
     I          WHEN CALLED I CONTAINS THE FORM NUMBER OF THE FUNCTION
                ON RETURN I CONTAINS THE NEW FORM NUMBER(SEE ABOVE)
     D3         SEE THE PROCEDURE MAIN FOR DETAILS
     M,N        THE VALUES IN ANY OF THESE FORMS :

                     1.    X**M*LN(X.SQRT(X**2%B**2))**N
                           (WHERE . AND % ARE '+' OR '-')
                     2.    X**M*LN(X)**N

STMT  NO              DESCRIPTION

 10 - 11          IN ANY CASE LN SHOULD APPEAR IN THE FUNCTION
                  ONLY ONCE
12 - 13           THE FUNCTION SHOULD BE ABLE TO BE
                  WRITTEN EITHER AS F'(X)*F(X)**M*OP
                  WHERE OP IS ANY OF THE SIX FORMS STATED ABOVE
                  OR F'(X)*F(X)**M*LN(F(X))**N
 15 - 51          THE INTEGRAND HAS ONE OF THE ABOVE SIX FORMS
 16               FORM CHECKING TO SEE FOR EXAMPLE WHETHER IN
                  LN(U-SQRT(V**2-B**2)) U AND V ARE EQUAL
 21               THE DEGREE OF LN IN THE INTEGRAND IN ANY CASE
                  SHOULD BE ONE
 24 - 44          STORE INFORMATION
                  INITIALISE VARIABLES AND
                  SET STATUS(0) = 109 TO CALL PROCEDURE SQRTS
                  AND SET I TO THE FORM NUMBER OF THE FORM INTO WHICH
                  THE FUNCTION IS TO BE TRANSFORMED
 40               THE NEW FORM NUMBER IS 1
 41               THE NEW FORM NUMBER IS 2
 48 - 50          IF THE ARGUMENT OF SQRT IS X**2 REPLACE IT BY X
                  IF THIS RESULTS IN THE ARGUMENT OF LN TO BE 0
                  RETURN WITH ERROR
                  AND THE INTEGRAND WILL BE INTEGRATED AS IF IT
                  INVOLVES ONLY LN ALONE
 53               THE INTEGRAND IS LN(X)**N/X (N NOT EQUAL -1)
                  THE INTEGRAL OF IT IS LN(X)**(N+1)/(N+1)
 54 - 59          THE INTEGRAND IS 1/(X*LN(X)) THE INTEGRAL OF IT IS
                  LN(LN(X))
 60               THE VALUE OF M IN F'(X)*F(X)**M*LN(F(X))**N MUST BE POSITIVE
 61 - 69          ELSE APPLY INTEGRATION FORMULA INVOLVING INTEGRATION
                  BY PARTS
```

```
PROCEDURE LOGFCT(A,B,M,N,I,D3)
    ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1) D1,D2,D3,MB,
          M5,SER,PARM,RESULT
    INTEGER A,B,M,N,I,M2,LK,JACK,STATUS
    EXTERNAL MB,M2,M5,LK,SER,PARM,JACK,STATUS,RESULT
    ARRAY (0:2) STATUS
    ARRAY (6) M2,M5
    ARRAY MB
    STATUS(1) = 56
    SPLIT(D3,Z(1),D1,D2)
    IF(D2 .NE. 1) RETURN
    GETPWR(M,0,D3)
    IF(STATUS(0) .NE. 55) RETURN
    STATUS(0) = 99
    IF(JACK .NE. 10) DO
          IF(MB(2) .NE. MB(3)) RETURN
          B = MB(1)
    STATUS(1) = 59
          IF(B .NE. 0) DO
                STATUS(1) = 56
                IF(N .NE. 1) RETURN
                ELSE DO
                      I = 10 - JACK
                      IF(I == 1 .OR. I == 2) A = 1
                      ELSE A = -1
                      LK = LK + 1
                      M2(LK) = 27
                      M5(LK) = SER + A * Z(LK + 7)
                      IF(I == 1 .OR. I == 3) A = 1
                      ELSE A = -1
                      LK = LK + 1
                      M2(LK) = 28
                      M5(LK) = SER ** 2 + A * B ** 2
                      PARM = PARM / (M + 1)
                      LK = LK + 1
                      M2(LK) = 30
                      M5(LK) = SER
                      RESULT = RESULT + PARM * Z(LK + 6) ** (M+1) * Z(LK+4)
                      IF(I == 1 .OR. I == 2) PARM = -PARM
                      IF(I == 1 .OR. I == 3) I = 1
                      ELSE I = 2
                      STATUS(0) = 109
                      M = M + 1
                      N = -1
                      RETURN
                DOEND
          DOEND
          ELSE IF(JACK == 6 .OR. JACK == 7) RETURN
          SER = SER * 2
          PARM = PARM / 2 ** (M + 1)
    DOEND
    STATUS(1) = 60
    IF(M == -1 .AND. N .NE. -1) RESULT = PARM * Z(4) ** (N + 1) / (N + 1)
    ELSE IF(M == -1 .AND. N == -1) DO
          LK = LK + 1
          RESULT = PARM * Z(LK + 6)
          M2(LK) = 27
          M5(LK) = Z(4)
    DOEND
    ELSE IF(N < 0) RETURN
    ELSE DO
          A = -1
          PARM = PARM / (M + 1)
          DO I = N , 1 , -1
                RESULT = RESULT + Z(4) ** I * PARM
                PARM = PARM * A * I / (M + 1)
          DOEND
          RESULT = (RESULT + PARM) * Z(6) ** (M + 1)
    DOEND
    STATUS(0) = 33
END
```

PROCEDURE TRIHY1(M,N)

THIS PROCEDURE INTEGRATES THE FUNCTION WHICH HAS ANY OF THESE FORMS :

1. F'(X)*SIN(F(X))**M*COS(F(X))**N
2. F'(X)*SINH(F(X))**M*COSH(F(X))**N

ARGUMENT

M,N SEE THE FORMS STATED ABOVE

STMT NO	DESCRIPTION
5 - 6	INITIALIZATION OF VARIABLES
7 - 15	IF M > -2 AND M < 2 AND N > -2 AND N < 2 THE INTEGRAL IS OBTAINED IN THE SNOBOL WRITER
16 - 21	REDUCTION FORMULA WHEN M < -1
22 - 27	REDUCTION FORMULA WHEN N < -1
28 - 33	REDUCTION FORMULA WHEN M > 1
34 - 39	REDUCTION FORMULA WHEN N >
38	GO TO LOOP

```
      PROCEDURE TRIHY1(M,N)
         ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1) PARM,RESULT
         INTEGER M,N,I,I1,M7,KEK
         EXTERNAL M7,KEK,PARM,RESULT
         IF(KEK == 1) I1 = -1
         ELSE I1 = 1
ACAN1:   IF(M.LE.1 .AND. M.GE.-1 .AND. N.LE.1 .AND. N.GE.-1) DO
               I = M * 2 + N * 3 + 5
               IF(I .EQ. 5) RESULT = RESULT + PARM * Z(6)
               ELSE IF(PARM .NE. 0) M7 = 4000 + 1000 * KEK + I
               IF(I == 10) PARM = PARM / 4
               IF((I == 0 .OR. I == 3 .OR. I == 4 .OR. I == 7 .OR. I == 10) .AND.
               KEK == 0)  PARM = -PARM
               RETURN
         DOEND
         IF(M < -1) DO
ACAN2:         RESULT = RESULT + Z(1) ** (M+1) * Z(2) ** (N+1) * PARM / (M+1)
               PARM = PARM * (M + N + 2) / (M + 1) * I1
               M = M + 2
               IF(M < -1) GO TO ACAN2
         DOEND
         IF(N < -1) DO
ACAN3:         RESULT = RESULT - Z(2) ** (N+1) * Z(1) ** (M+1) * PARM / (N+1)
               PARM = PARM * (M + N + 2) / (N + 1)
               N = N + 2
               IF(N < -1) GO TO ACAN3
         DOEND
         IF(M > 1) DO
ACAN4:         RESULT = RESULT - Z(1) ** (M-1) * Z(2) ** (N+1) * PARM / (M+N) *I1
               PARM = PARM * (M - 1) / (M + N) * I1
               M = M - 2
               IF(M > 1) GO TO ACAN4
         DOEND
         IF(N > 1) DO
ACAN5:         RESULT = RESULT + Z(2) ** (N-1) * Z(1) ** (M+1) * PARM / (M+N)
               PARM = PARM * (N - 1) / (M + N)
               N = N - 2
               IF(N > 1) GO TO ACAN5
         DOEND
         GO TO ACAN1
      END
```

```
                PROCEDURE TRIHY2(M,N,I,OM)

THIS PROCEDURE INTEGRATES THE FUNCTION WHICH HAS ANY OF THESE FORMS :

     1.         F'(X)*F(X)**M*OP(F(X))**N
     2.         E(X)**M*OP1(F(X))*OP2(G(X))
                (WHERE OP,OP1,OP2 ARE TRI. OR HYP. FUNCTIONS)

                     ARGUMENT

     I          THE INTEGER CORRESPONDING TO OP(SEE THE FORMS STATED ABOVE
                SEE ALSO "DELAY" IN PROCEDURE MAIN)
     M,N,OM     SEE THE FORMS STATED ABOVE

STMT  NO                DESCRIPTION

 10 - 14        INITIALIZATION OF VARIABLES
 15 - 22        THE INTEGRAND IS F'(X)*F(X)**M*OP(F(X))**2 AND
                IS TRANSFORMED INTO TWO PARTS USING DOUBLE
                ANGLE RULES
                FOR EXAMPLE
                X**3*SIN(X)**2 = X**3 * (1 - COS(2*X))/2
 16 - 18        THE FIRST PART OF THE INTEGRAL IS OBTAINED
                DIRECTLY
 23 - 29        THE INTEGRAND IS F'(X)*F(X)**M*OP(F(X))**3
                AND IS TRANSFORMED INTO TWO PARTS USING TRIPLE ANGLE RULE
 28             TRIPLE THE ARGUMENT FOR THE SECOND PART
 30 - 34        IF N < 1 OR N > 3 RETURN WITH ERROR
 35 - 41        INITIALISATION OF VARIABLES
 42 - 52        INTEGRATION
 54 - 63        WHEN DELAY EQUAL 1 THE SECOND PART IS TO BE
                INTEGRATED WHEN OM NOT EQUAL M
 64 - 66        WHEN N = 3 OR DELAY = 1 WITH OM EQUAL TO M
                THE SECOND INTEGRAL IS OBTAINED FROM THE FIRST
                BY SUBSTITUTION
 67 - 69        THE SECOND INTEGRAL (M NOT EQUAL OM)
 71 - 77        STORE INFORMATION FOR SECOND INTEGRAL FOR
                USE BY THE SNOBOL WRITER TO TRANSLATE THE INTEGRAL
                CALL PROCEDURE TRIHY1
 14 - 35        AFTER THE INTEGRAL IS OBTAINED PERFORM BACK SUBSTITUTION
```

```
      PROCEDURE TRIHY2(M,N,I,OM)
         ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1) &
                   M5,PAR,SER,OSER,PARM,OPARM,RESULT
         INTEGER I,J,K,L,M,N,P,OM,II,M2,LK,KEK,DELAY,STATUS
         EXTERNAL M2,M5,LK,KEK,SER,PARM,OSER,OPARM,DELAY,RESULT,STATUS
         ARRAY (6) M2,M5
         ARRAY (0:2) STATUS
         STATUS(1) = 45
         IF(DELAY .NE. 1) OM = M
         P = I
SHEL1:   PAR = 1
         K = 4 * N - 2 * I + 25 * KEK + 1
         IF(M .LT. 0) RETURN
         IF(N .EQ. 2) DO
               PARM = PARM / (2 ** (M + 2) * (M + 1))
               RESULT = Z(6) ** (M + 1)
               IF(K .EQ. 32) RESULT = -RESULT
               IF(K .EQ. 7) PAR = -1
               SER = SER * 2
               I = 2
         DOEND
         ELSE IF(N .EQ. 3) DO
               PARM = PARM * 3 / 4
               IF(K == 12) PARM = -PARM
               OPARM = PARM / (3 ** (M + 2))
               IF(I .EQ. 1 .OR. I .EQ. 13) OPARM = -OPARM
               OSER = 3 * SER
         DOEND
         ELSE IF(N .NE. 1) RETURN
         I = I - KEK * 12
         IF(I .EQ. 1) I = 2
         ELSE I = 1
         IF(I .EQ. 2 .AND. KEK .EQ. 0) J = -1
         ELSE J = 1
         IF(I .EQ. 1) K = 0
         ELSE K = 1
         DO L = M , 1 , -1
               RESULT = RESULT + Z(I) * Z(6) ** L * PAR * J
               PAR = PAR * L
               IF(I .EQ. 1) I = 2
               ELSE I = 1
               K = K + 1
               IF(K .EQ. 2 .OR. KEK .EQ. 1) DO
                     K = 0
                     J = -J
               DOEND
         DOEND
         RESULT = RESULT + PAR * Z(I) * J
         IF(OM .NE. M) DO
               DELAY = 2
               M = OM
               PAR = OPARM
               OPARM = RESULT * PARM
               PARM = PAR
               RESULT = 0
               I = P
               GO TO SHEL1
         DOEND
         ELSE IF(DELAY ==1 .OR. N == 3) RESULT = PARM * RESULT +
                           OPARM * RESULT(Z(1) = Z(LK + 7))(Z(2) = Z(LK + 8))   &
                           (Z(6) = OSER)
         ELSE IF(DELAY == 2) RESULT = OPARM +
                                 RESULT(Z(1) = Z(LK + 7))(Z(2) = Z(LK + 8))     &
                                 (Z(6) = OSER) * PARM
         ELSE RESULT = RESULT * PARM
         IF(DELAY .NE. 0 .OR. N == 3) DO
               LK = LK + 2
               M2(LK - 1) = KEK * 12 + 1
               M2(LK) = KEK * 12 + 2
               M5(LK - 1) = OSER
               M5(LK) = OSER
         DOEND
      END
```

PROCEDURE TRIHYP

THIS PROCEDURE INTEGRATES THE FUNCTION WHICH ONLY INVOLVES SIN,COS,
SINH OR COSH BUT IT IS NOT ANY OF THE SPECIAL FUNCTIONS LISTED IN THE
APPENDIX G
THE FUNCTION MUST ALSO CONTAIN MORE THAN ONE TERM IN EITHER ITS
NUMERATOR OR DENOMINATOR AND FOR THE SIN AND COS CASE THE DENOMINATOR
OF THE FUNCTION TO BE INTEGRATED MUST INVOLVE SIN OR COS
IN ANY CASE THE FUNCTION SHOULD NOT CONTAIN THE VARIABLE AS A SINGLE
ENTITY.

STMT NO	DESCRIPTION
9 - 10	ALL ARGUMENTS OF FUNCTIONS IN THE INTEGRAND SHOULD BE EQUAL
13 - 14	Z0 MAY NOT APPEAR IN THE INTEGRAND
15 - 27	THE INTEGRAND IS IN THE TRIGONOMETRIC CASE SUBSTITUTE SIN(X) = 2 * T / (1 + T ** 2) SUBSTITUTE COS(X) = (1 - T ** 2) / (1 + T ** 2)
28 - 37	THE INTEGRAND IS IN THE HYPERBOLIC CASE SUBSTITUTE SINH(X) = (EXP(X) - EXP(-X)) / 2 SUBSTITUTE COSH(X) = (EXP(X) + EXP(-X)) / 2
38	PROCEDURE RATFCT WILL BE CALLED

```
PROCEDURE TRIHYP
   ALGEBRAIC (Z(12):8,Z0:8,Z2:8,Z3:8,Z1(6):8,ZZ(12):1) D1,D2,D3,MB,FCT
   EXTERNAL MA,MB,KEK,LAW,FCT,STATUS
   INTEGER MA,KEK,LAW,STATUS
   ARRAY (0:2) STATUS
   ARRAY MA,MB
   STATUS(0)=99
   STATUS(1) = 46
   IF(MB(2)==99) MB(2) = MB(1)
   IF(MB(3) .NE. 99 .OR. MB(1) - MB(2) .NE. 0) RETURN
   STATUS(1) = 43
   FCT = FCT / DIFF(MB(1),Z0)
   SPLIT(FCT,Z0,D1,D2)
   IF(D2 .NE. 1) RETURN
   IF(KEK .EQ. 0) DO
        D1 = ADEN(FCT,D2)
        STATUS(1) = 47
        IF(DEG(D1,Z(1)) .EQ. 0 .AND. DEG(D1,Z(2)) .EQ. 0) RETURN
        LAW = 3
        IF(MA(2) .EQ. 99) DO
             IF(MA(1) .EQ. 1) FCT = FCT(Z(1) = 2 * Z0 / (1 + Z0 ** 2))
             IF(MA(1)==2) FCT = FCT(Z(1) = (1 - Z0 **2)/(1 + Z0 ** 2))
   DOEND
        ELSE FCT=FCT(Z(MA(2))=(1-Z0**2)/(1+Z0**2))(Z(MA(1))=Z0*2/
             (1+Z0**2))
        FCT = FCT * 2 / (1 + Z0**2)
   DOEND
   ELSE DO
        LAW = 2
        IF(MA(2) .EQ. 99) DO
             IF(MA(1) .EQ. 13) FCT = FCT(Z(1) = (Z0 - 1/Z0)/2)
             IF(MA(2) .EQ. 14) FCT = FCT(Z(1) = (Z0 + 1/Z0)/2)
        DOEND
        ELSE FCT = FCT(Z(MA(2) - 12) = (Z0 + 1/Z0)/2) &
                       (Z(MA(1) -12) = (Z0 - 1/Z0)/2)
        FCT = FCT / Z0
   DOEND
   STATUS(0) = 104
END
```

```
-UNLIST
*
*
*
*
*                                  **********************
*                                  *       WRITER       *
*                                  *                    *
*                                  **********************
*
*
*
*
*
*
*
*                 STRINGS PATTERNS DEFINITIONS

        INPUT('INPUT',3,80)
        OUTPUT('OUTPUT',9,'(1X,132A1)')
        &TRIM = 1
        &ANCHOR = 0
        LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
        DIGITS = '0123456789'
        OPERATORS = '+-*/'
        CONTENT = SPAN(DIGITS LETTERS OPERATORS '()')
        TEXT1 = ARRAY(30,'')
        TEXT2 = ARRAY(6,'')
        TECH = ARRAY(30,'')
        KIND = SPAN(DIGITS '.')
*
*
*                 THE SET OF DERIVATIVES OF ELEMENTARY FUNCTIONS
*
*
        TECH<1> = 'COS#'
        TECH<2> = '(-SIN#)'
        TECH<3> = '(-CSC#*COT#)'
        TECH<4> = 'TAN#*SEC#'
        TECH<5> = '(SEC#)**2'
        TECH<6> = '(-CSC#)**2'
        TECH<7> = '(1/SQRT(1-#**2))'
        TECH<8> = '(-1/SQRT(1-#**2))'
        TECH<9> = '(-1/(#*SQRT(#**2-1)))'
        TECH<10> = '(1/(#*SQRT(#**2-1)))'
        TECH<11> = '(1/(1+#**2))'
        TECH<12> = '(-1/(1+#**2))'
        TECH<13> = 'COSH#'
        TECH<14> = 'SINH#'
        TECH<15> = 'CSCH#*COTH#'
        TECH<16> = 'TANH#*SECH#'
        TECH<17> = 'SECH#**2'
        TECH<18> = 'CSCH#**2'
        TECH<19> = '(1/SQRT(#**2+1))'
        TECH<20> = '(-1/SQRT(#**2+1))'
        TECH<21> = '(-1/SQRT(#**2*(1+#**2)))'
        TECH<22> = '(-1/(#*SQRT(1-#**2)))'
        TECH<23> = '(1/(1-#**2))'
        TECH<24> = '(1/(1-#**2))'
        TECH<25> = '(1/(#LOG10))'
```

```
TECH<26> = 'EXP#'
TECH<27> = '(1/#)'
TECH<28> = '(1/(2*SQRT#))'
TEXT1<29> = 'LOG'
*
*
*                THE ARRAY OF ELEMENTARY FUNCTIONS
*                INDEXED BY THE INTEGERS CORRESPONDING TO
*                THE FUNCTION NAMES
*
*

TEXT1<1> = 'SIN'
TEXT1<2> = 'COS'
TEXT1<3> = 'CSC'
TEXT1<4> = 'SEC'
TEXT1<5> = 'TAN'
TEXT1<6> = 'COT'
TEXT1<7> = 'ASIN'
TEXT1<8> = 'ACOS'
TEXT1<9> = 'ACSC'
TEXT1<10> = 'ASEC'
TEXT1<11> = 'ATAN'
TEXT1<12> = 'ACOT'
TEXT1<13> = 'SINH'
TEXT1<14> = 'COSH'
TEXT1<15> = 'CSCH'
TEXT1<16> = 'SECH'
TEXT1<17> = 'TANH'
TEXT1<18> = 'COTH'
TEXT1<19> = 'ASINH'
TEXT1<20> = 'ACOSH'
TEXT1<21> = 'ACSCH'
TEXT1<22> = 'ASECH'
TEXT1<23> = 'ATANH'
TEXT1<24> = 'ACOTH'
TEXT1<26> = 'EXP'
TEXT1<27> = 'LN'
TEXT1<28> = 'SQRT'
*
*
*                THE ARRAY USED FOR DIRECT SUBSTITUTION OF
*                Z(I) BY A FUNCTION NAME
*
*

TEXT2<1> = 'SIN'
TEXT2<2> = 'COS'
TEXT2<3> = 'EXP'
TEXT2<4> = 'LN'
TEXT2<5> = 'SQRT'
*
*
*                THE INTEGRATION TABLE USED FOR THE SPECIAL FUNCTIONS
*
*

TEXT3 = ARRAY(105,'')
TEXT3<1> = 'LOG|TAN(((&)+ASIN(#/SQRT($)))/2)|/SQRT($)'
TEXT3<2> = 'ATAN(SQRT($)*TAN(&)/#)/SQRT($)'
```

```
TEXT3<3> = 'ATAN((#)*TAN(Ɛ)/SQRT($))/SQRT($)'
TEXT3<4> = 'LOG|(SQRT($)*TAN(Ɛ)+#)/(SQRT($)*TAN(Ɛ)-#)|'
TEXT3<5> = 'LOG|((#)*TAN(Ɛ)-SQRT($))/((#)*TAN(Ɛ)+SQRT($))|'
TEXT3<6> = '(Ɛ)'
TEXT3<7> = '(Ɛ)'
TEXT3<8> = 'LOG|TAN((Ɛ)/2)|'
TEXT3<9> = 'LOG|TAN((Ɛ)/2+PHI/4)|'
TEXT3<10> = 'COS(Ɛ)/(#+?*SIN(Ɛ))'
TEXT3<11> = 'SIN(Ɛ)/(#+?*COS(Ɛ))'
TEXT3<12> = 'LOG(1+COS(Ɛ))'
TEXT3<13> = 'LOG(1-COS(Ɛ))'
TEXT3<14> = 'LOG(1+SIN(Ɛ))'
TEXT3<15> = 'LOG(1-SIN(Ɛ))'
TEXT3<16> = 'LOG|(1+COS(Ɛ))/COS(Ɛ)|'
TEXT3<17> = 'LOG|(1-COS(Ɛ))/COS(Ɛ)|'
TEXT3<18> = 'LOG|(1+SIN(Ɛ))/SIN(Ɛ)|'
TEXT3<19> = 'LOG|(1-SIN(Ɛ))/SIN(Ɛ)|'
TEXT3<20> = 'LOG|TAN((Ɛ)/2+PHI/8)|/SQRT(2)'
TEXT3<21> = 'LOG|TAN((Ɛ)/2-PHI/8)|/SQRT(2)'
TEXT3<24> = 'TAN((Ɛ)-PHI/4)'
TEXT3<25> = 'TAN((Ɛ)+PHI/4)'
TEXT3<28> = '/(1+COS(Ɛ))_LOG|TAN((Ɛ)/2)|'
TEXT3<29> = '/(1-COS(Ɛ))_LOG|TAN((Ɛ)/2)|'
TEXT3<30> = '/(1+SIN(Ɛ))_LOG|TAN(PHI/4+(Ɛ)/2)|'
TEXT3<31> = '/(1-SIN(Ɛ))_LOG|TAN(PHI/4+(Ɛ)/2)|'
TEXT3<32> = '(Ɛ)_LOG|SIN(Ɛ)+COS(Ɛ)|'
TEXT3<33> = '(Ɛ)_LOG|SIN(Ɛ)-COS(Ɛ)|'
TEXT3<34> = '(Ɛ)_LOG|SIN(Ɛ)+COS(Ɛ)|'
TEXT3<35> = '(Ɛ)_LOG|SIN(Ɛ)-COS(Ɛ)|'
TEXT3<36> = '/(1+SIN(Ɛ))_LOG|TAN(PHI/4+(Ɛ)/2)|'
TEXT3<37> = '/(1-SIN(Ɛ))_LOG|TAN(PHI/4+(Ɛ)/2)|'
TEXT3<38> = '/(1+COS(Ɛ))_LOG|TAN((Ɛ)/2)|'
TEXT3<39> = '/(1-COS(Ɛ))_LOG|TAN((Ɛ)/2)|'
TEXT3<40> = 'TAN(PHI/4-(Ɛ)/2)|'
TEXT3<41> = 'TAN((Ɛ)/2)'
TEXT3<42> = 'TAN(PHI/4+(Ɛ)/2)'
TEXT3<43> = 'COT((Ɛ)/2)'
TEXT3<44> = '(Ɛ)_TAN(PHI/4-(Ɛ)/2)'
TEXT3<45> = '(Ɛ)_TAN((Ɛ)/2)'
TEXT3<46> = '(Ɛ)_TAN(PHI/4+(Ɛ)/2)'
TEXT3<47> = '(Ɛ)_COT((Ɛ)/2)'
TEXT3<48> = 'TAN(PHI/4-(Ɛ)/2)_LOG|TAN((Ɛ)/2)|'
TEXT3<49> = 'LOG|TAN(PHI/4+(Ɛ)/2)|_TAN((Ɛ)/2)'
TEXT3<50> = 'TAN(PHI/4+(Ɛ)/2)_LOG|TAN((Ɛ)/2)|'
TEXT3<51> = 'LOG|TAN(PHI/4+(Ɛ)/2)|_COT((Ɛ)/2)'
TEXT3<52> = 'TAN(PHI/4-(Ɛ)/2)_TAN(PHI/4-(Ɛ)/2)**3'
TEXT3<53> = 'TAN((Ɛ)/2)_TAN((Ɛ)/2)**3'
TEXT3<54> = 'COT(PHI/4-(Ɛ)/2)_COT(PHI/4-(Ɛ)/2)**3'
TEXT3<55> = 'COT((Ɛ)/2)_COT((Ɛ)/2)**3'
TEXT3<56> = 'TAN(PHI/4-(Ɛ)/2)_TAN(PHI/4-(Ɛ)/2)**3'
TEXT3<57> = 'TAN((Ɛ)/2)_TAN((Ɛ)/2)**3'
TEXT3<58> = 'COT(PHI/4-(Ɛ)/2)_COT(PHI/4-(Ɛ)/2)**3'
TEXT3<59> = 'COT((Ɛ)/2)_COT((Ɛ)/2)**3'
TEXT3<60> = 'ASIN((3*SIN(Ɛ)**2-1)/(SIN(Ɛ)**2+1))/SQRT(2)'
TEXT3<61> = 'TAN(Ɛ)'
TEXT3<62> = 'ASIN((1-3*COS(Ɛ)**2)/(1-COS(Ɛ)**2))/SQRT(2)'
TEXT3<63> = 'COT(Ɛ)'
```

```
      TEXT3<64> = '*COS(&)*SQRT(1+(?)*SIN(&)**2)'
     +'_ASIN((#)*COS(&)/SQRT($)))'
      TEXT3<65> = '*COS(&)*SQRT(1-(?)*SIN(&)**2)_LOG((#)*'
     +            '*COS(&)+SQRT(1-(?)*SIN(&)**2))'
      TEXT3<66> = '*SIN(&)*SQRT(1+(?)*SIN(&)**2)_'
     +'LOG((#)*SIN(&)+SQRT(1+(?)*SIN(&)**2))'
      TEXT3<67> = '*SIN(&)*SQRT(1-(?)*SIN(&)**2)_'
     +'ASIN((#)*SIN(&))'
      TEXT3<69> = '*ASIN((#)*COS(&)/SQRT($))'
      TEXT3<70> = '*LOG((#)*COS(&)+SQRT(1-(?)*SIN(&)**2))'
      TEXT3<71> = '*LOG((#)*SIN(&)+SQRT(1+(?)*SIN(&)**2))'
      TEXT3<72> = '*ASIN((#)*SIN(&))'
      TEXT3<73> = '*ATAN((#)*TAN(&))'
      TEXT3<74> = '*LOG|((?)*TAN(&)+(#))/((?)*TAN(&)-(#))|'
      TEXT3<75> = '*(&)*SIN(LOG(&))_(&)*COS(LOG(&))'
      TEXT3<77> = '*ATAN(((#)*TAN((&)/2)+?)/SQRT($))/SQRT($)'
      TEXT3<78> = '*ATAN(((:)*TAN((&)/2))/SQRT($))/SQRT($)'
      TEXT3<79> = '*LOG|((#)*TAN((&)/2)+?-SQRT($))/((#)*TAN'
     +            '*((&)/2)+?-SQRT($))|/SQRT($)'
      TEXT3<80> = '*LOG|((.)*TAN((&)/2)+SQRT($))/((.)*TAN'
     +'*((&)/2)-SQRT($))|/SQRT($)'
      TEXT3<81> = '*LN(CSC(2*(#)) + COT(2*(#)))'
      TEXT3<83> = '*LN(SEC(#) + TAN(#))'
      TEXT3<84> = '*LN(CSC(#) + COT(#))'
      TEXT3<85> = '*LN(COS(#))'
      TEXT3<87> = '*LN(SIN(#))'
      TEXT3<88> = '*COS(#)'
      TEXT3<89> = '*SIN(#)'
      TEXT3<91> = '*COS(2*(#))'
      TEXT3<92> = '*(&)'
      TEXT3<93> = '*LOG|(TAN(&)+1)/(TAN(&)-1)|'
      TEXT3<94> = '*LN(CSCH(2*(#))+COTH(2*(#)))'
      TEXT3<96> = '*ASEC(#)'
      TEXT3<97> = '*LN(CSCH(#)+COTH(#))'
      TEXT3<98> = '*LN(COSH(#))'
      TEXT3<100> = '*LN(SINH(#))'
      TEXT3<101> = '*COSH(#)'
      TEXT3<102> = '*SINH(#)'
      TEXT3<104> = '*COSH(2*(#))'
*
*
*           THE ARRAY USED FOR INDIRECT SUBSTITUTION OF
*           Z(I) BY A FUNCTION NAME
*
*
      TEXT5 = '1SIN2COS13SINH14COSH26EXP27LN28SQRT'
*
*
*           THE SET OF ERROR MESSAGES
*
*
      ERROR = ARRAY(65,'')
      ERROR<1> = 'THE FUNCTION IS NULL'
      ERROR<2> = 'THE ARGUMENT OF LN IS NOT IN A CORRECT FORM'
      ERROR<3> = 'THE ARGUMENT OF A FUNCTION IS NOT ACCEPTABLE'
      ERROR<4> = 'THE FUNCTION NAMES IN THE ARGUMENT OF '
     +           'LN SHOULD BE THE SAME BUT IS NOT'
```

```
      ERROR<5> = 'A FUNCTION NAME IS UNRECOGNIZABLE'
      ERROR<6> = 'THE LEFT AND RIGHT PARENTHESIS ARE NOT '
     +           'MATCHED IN THE FUNCTION'
      ERROR<7> = 'THE DERIVATIVE OF THE NESTED FUNCTION '
     +           'IS NOT FOUND IN THE FUNCTION'
      ERROR<8> = 'IN THE FUNCTION THE OPERATORS ARE MISPLACED '
     +           'OR () OR )( IS FOUND'
      ERROR<9> = 'THE FUNCTION CONTAINS AN UNACCEPATBLE CHARACTER'
      ERROR<10> = 'THE INTEGRAND SHOULD NOT INVOLVE ANY '
     +           'TWO FUNCTIONS FROM LN, SQRT AND INVERSE FUNCTIONS'
      ERROR<11> = 'THE INTEGRAND SHOULD NOT INVOLVE ANY '
     +           'TWO FUNCTIONS IN WHICH ONE IS LN OR SQRT '
     +           'OR AN INVERSE FUNCTION'
      ERROR<12> = 'THE INTEGRAND SHOULD NOT INVOLVE BOTH '
     +           'TRIGONOMETRIC AND HYPERBOLIC FUNCTIONS'
      ERROR<13> = 'THE INTEGRAND INVOLVING NESTING OF FUNCTIONS '
     +           'CONTAINS AN UNEXPECTED FUNCTION NAME'
      ERROR<14> = 'THE NESTED FUNCTION NAME IS UNACCEPTABLE'
      ERROR<15> = 'THE NESTED FUNCTION NAMES ARE NOT IDENTICAL'
      ERROR<16> = 'THE ARGUMENTS OF ALL THE NESTED FUNCTIOS '
     +           'SHOULD BE THE SAME BUT IS NOT'
      ERROR<17> = 'THE DERIVATIVE OF THE NESTED FUNCTICN '
     +           'IS NOT PRESENT IN THE FUNCTION'
      ERROR<18> = 'THE INTEGRAND INVOLVING ONLY SIN AND COS IS '
     +           'IN AN UNACCEPTABLE FORM'
      ERROR<19> = 'ATTEMPTED TO DIVIDE AN EXPRESSION BY 0 DURING'
     +           'THE INTEGRATION OF THE FUNCTION INVOLVING EXP'
      ERROR<20> = 'AN ALPHABETIC CONSTANT IS NOT A '
     +           'SINGLE CHARACTER'
      ERROR<21> = 'THE INTEGRAND CONTAINS A**F(X) AND B**E(X) '
     +           'BUT A AND B ARE NOT THE SAME INTEGER'
      ERROR<22> = 'THE ARGUMENT OF ANY FUNCTION WHICH '
     +           'IS NOT AN INTEGER SHOULD CONTAIN THE VARIABLE'
      ERROR<23> = 'THE FORM A ** F(X) APPEARS IN THE ARGUMENT '
     +           'OF A FUNCTION NAME'
      ERROR<24> = 'IN THE FUNCTION DIGITS AND LETTERS '
     +           'SHOULD BE SEPERATED BY AN OPERATOR BUT IS NOT'
      ERROR<25> = 'IN THE FUNCTION A SINGLE LETTER IS FOLLOWED '
     +           ' BY ('
      ERROR<26> = 'IN THE FUNCTION AN INTEGER IS FOLLOWED BY ('
      ERROR<27> = 'NUMERIC OVERFLOW OR UNDERFLOW'
      ERROR<28> = 'RECIPROCAL OF 0 ATTEMPTED'
      ERROR<29> = 'ATTEMPTED TO COMPUTED 0 ** 0'
      ERROR<30> = 'EXPONENT OVERFLOW DURING COMPUTATION'
      ERROR<31> = 'A COEFFICIENT IN COMPUTATION BECAME TOO LARGE'
      ERROR<32> = 'ILLEGAL CHARACTERS IN THE INPUT'
      ERROR<33> = 'ILLEGAL SYNTAX IN THE INPUT'
      ERROR<34> = 'ILLEGAL SEMANTICS IN THE INPUT'
      ERROR<35> = 'THE WORKSPACE WAS NOT ENOUGH FOR COMPUTING '
     +           'YOUR PROBLEM'
      ERROR<36> = 'THE INTEGRAND HAS MORE THAN EXPECTED NUMBER'
     +           ' OF DISTINCT ARGUMENTS'
      ERROR<37> = 'THE INTEGRAND HAS MORE THAN EXPECTED NUMBER OF '
     +           'DISTINCT ARGUMENTS OR EXP APPEARS MORE '
     +           'THAN ONCE'
      ERROR<38> = 'THE POWER OF SQRT IS OUT OF THE PACKAGE RANGE '
     +           '(IT SHOULD BE 1 OR -1)'
```

```
      ERROR<39> = 'THE ARGUMENTS OF EXP SHOULD BE EITHER '
     +            'F(X) OR -F(X)'
      ERROR<40> = 'THE FUNCTION CONTAINING EXP AND TRI. FUNCTION '
     +            'OR HYP. FUNCTION IS NOT IN AN ACCEPTABLE FORM'
      ERROR<41> = 'THE POWER OF EXP IS OUT OF THE PACKAGE RANGE '
     +            '(IT SHOULD BE 1)'
      ERROR<42> = 'THE POWERS OF TRI. FUNCTIONS OR HYP. FUNCTIONS '
     +            'IS OUT OF THE PACKAGE RANGE(IT SHOULD BE 1)'
      ERROR<43> = 'THE DERIVATIVE OF A FUNCTION IS NOT FOUND '
     +            'IN THE INTEGRAND'
      ERROR<44> = 'THE FUNCTION IS EXPECTED TO HAVE THE FORM '
     +            'F,(X) * F(X) ** M * OP(F(X)) WHERE OP IS ANY '
     +            'INVERSE TRI. FUNCTION OR HYP. FUNCTION '
     +            'BUT IS NOT'
      ERROR<45> = 'THE POWER OF THE TRI. FUNCTION OR HYP. '
     +            'IS OUT OF THE PACKAGE RANGE(IT SHOULD BE '
     +            'BETWEEN 1 AND 3'
      ERROR<46> = 'THE ARGUMENTS OF SIN AND COS ARE EXPECTED '
     +            'TO BE THE SAME BUT IS NOT'
      ERROR<47> = 'THE DENOMINATOR OF THE INTEGRAND SHOULD '
     +            'INVOLVE SIN OR COS BUT IS NOT'
      ERROR<48> = 'THE INTEGRAND INVOLVING SQRT DOES NOT '
     +            'BELONG TO ANY ACCEPTABLE FORM'
      ERROR<49> = 'THE POWER OF THE TRI. FUNCTION OR HYP. '
     +            'FUNCTION IS OUT OF THE PACKAGE RANGE '
     +            '(IT SHOULD BE 1)'
      ERROR<50> = 'THE INTEGRAND CAN NOT BE ARRANGED TO THE FORM '
     +            'F,(X) * F(X) ** M * OP(F(X)) '
      ERROR<51> = 'THE DENOMINATOR OF THE INTEGRAND IS OUT OF '
     +            'THE PACKAGE RANGE'
      ERROR<52> = 'THE ARGUMENTS OF ALL THE FUNCTIONS '
     +            'SHOULD BE THE SAME BUT IS NOT'
      ERROR<53> = 'THE INTEGRAND HAS BEEN TRANSFORMED BY '
     +            'Y = EXP(F(X)) BUT THE INTEGRAND THUS FORMED '
     +            'IS NOT ACCEPTABLE'
      ERROR<54> = 'THE INTEGRAND HAS BEEN TRANSFORMED BY '
     +            'Y = TAN(X)/2 BUT THE INTEGRAND THUS FORMED '
     +            'IS NOT ACCEPTABLE'
      ERROR<55> = 'THE INTEGRAND HAS BEEN TRANSFORMED BY '
     +            'Y = SQRT(F(X)) BUT THE INTEGRAND THUS FORMED '
     +            'IS NOT ACCEPTABLE'
      ERROR<56> = 'THE INTEGRAND WHICH CONTAINS '
     +            'LN(X+SQRT(X**2+A**2)) SHOULD HAVE THE FORM '
     +            'X**M*LN(X+SQRT(X**2+A**2)) BUT IS NOT'
      ERROR<57> = 'THE POWERS OF THE TRI. FUNCTIONS OR THE '
     +            'HYP. FUNCTIONS SHOULD BE EQUAL BUT IS NOT'
      ERROR<58> = 'THE INTEGRAND SHOULD BE EITHER '
     +            'X/SQRT(A*X**2+B*X+C) OR SQRT(A*X**2+B*X+C) '
     +            'OR 1/SQRT(A*X**2+B*X+C) BUT IS NOT'
      ERROR<59> = 'THE FUNCTION SHOULD NOT CONTAIN '
     +            'LN(X-SQRT(X**2+-0**2))'
      ERROR<60> = 'IN THE INTEGRAND X**M*LN(X)**N N SHOULD '
     +            'BE POSITIVE WHEN M NOT EQUAL -1 BUT IS NOT'
      ERROR<61> = 'THE FUNCTION CONTAINS MORE THAN MAXIMUN '
     +            'ALLOWED NUMBER(12) OF FUNCTION NAMES'
      ERROR<62> = 'THE FUNCTION CONTAINS MORE THAN MAXIMUN '
     +            'ALLOWED NUMBER(5) OF ALPHABETIC CONSTANTS'
```

```
          ERROR<63> = 'THE PROBLEM IS UNRECOGNIZABLE'
*
*
*                    THE WRITER STARTS HERE
*
*
          M0 = INPUT
STRT1     M0 = INPUT
          M0 = INPUT
          M0 SPAN(DIGITS) . FLAG0 =
          EQ(FLAG0,999)                                          :S(EXIT)
          M0 SPAN(DIGITS) . FLAG1 =
          M0 SPAN(DIGITS) . FLAG2
          SAVE1 = ''
          SAVE2 = ''
          M7 = 0
          NEST = 95
          M0 = INPUT                                             :F(FAIL)
          M0 = INPUT                                             :F(FAIL)
          M0 = INPUT                                             :F(FAIL)
          KN = INPUT
          KN SPAN(DIGITS) . IN =
          LETTERS LEN(IN) LEN(1) . SAVE
          INPUT('INPUT',10,400)
          FUNCTION = INPUT
          GT(FLAG2,1)                                            :F(STRT2)
          TERM = INPUT
          TERMNO = 0
STRT2     EQ(FLAG2,0)                                            :S(STRT3)
          COUNT = COUNT + 1
          OUTPUT = 'PROBLEM ' COUNT
          OUTPUT =
          FUNCTION 'INTE' NOTANY(LETTERS) . C0 = 'INTEGRATE' C0
          FUNCTION 'DIFF' NOTANY(LETTERS) . C0 = 'DIFFERENTIATE' C0
          OUTPUT = FUNCTION
          EQ(FLAG1,63)                                           :S(STRT5)
          OUTPUT =
          OUTPUT = ' WITH RESPECT TO ' SAVE
          MESSAGE = EQ(FLAG2,1) 'THE RESULT IS :'
          OUTPUT = NE(FLAG2,1)
          OUTPUT = NE(FLAG2,1) 'THE INTEGRAL YOU HAVE PROVIDED '
+                 'HAS BEEN SEPERATED INTO ' FLAG2 ' TERMS'
          OUTPUT = NE(FLAG2,1) 'WHICH HAVE BEEN INTEGRATED SEPERATELY'
          EQ(FLAG2,1)                                            :S(STRT4)
STRT3     TERM = EQ(FLAG2,0) FUNCTION
          TERMNO = TERMNO + 1
          OUTPUT =
          OUTPUT = 'TERM ' TERMNO
          OUTPUT =
          OUTPUT = TERM
          MESSAGE = 'THE INTEGRAL IS :'
STRT4     INPUT('INPUT',3,80)
          EQ(FLAG0,33)                                           :S(STRT6)
          M0 = INPUT
          M0 = INPUT
          OUTPUT =
          LT(FLAG1,16)                                           :S(STRT5)
```

```
        GE(FLAG1,20) LT(FLAG1,26)                                   :F(STRT5G)
STRT5   OUTPUT = '*** INPUT ERROR *** ATTEMPT AT PROBLEM '
+               'SOLUTION CANCELLED ***'
        OUTPUT = '*** ' ERROR<FLAG1> ' ***'
        IN = FLAG1
        NE(IN,3) NE(IN,5) NE(IN,7) NE(IN,9) NE(IN,13) NE(IN,14)
+       NE(IN,20) NE(IN,25)                                         :S(PRNT7)
        FCTNAME =
STRT5A  KN SPAN(DIGITS) . IN =
        EQ(IN,100)                                                  :F(STRT5A)
STRT5B  KN SPAN(DIGITS) . IN =
        EQ(IN,99)                                                   :S(STRT5C)
        LETTERS LEN(IN) LEN(1) . CHAR
        FCTNAME = FCTNAME CHAR                                      :(STRT5B)
STRT5C  EQ(FLAG1,9)                                                 :S(PRNT7)
STRT5D  EQ(FLAG1,20)                                                :F(STRT5E)
        OUTPUT = '*** THE CONSTANT IS ' FCTNAME ' ***'              :(PRNT7)
STRT5E  EQ(FLAG1,25)                                                :F(STRT5F)
        OUTPUT = '*** THE SINGLE LETTER IS ' FCTNAME ' ***'
+                                                                   :(PRNT7)
STRT5F  OUTPUT = '*** THE FUNCTION NAME IS ' FCTNAME ' ***'
+                                                                   :(PRNT7)
STRT5G  OUTPUT = '*** RUN-TIME ERROR *** INTEGRATION ON THE '
+               'FUNCTION CANCELLED ***'
        OUTPUT = '*** THE FUNCTION IS BEYOND PACKAGE CAPABILITY ***'
        OUTPUT = '*** ' ERROR<FLAG1> ' ***'                         :(PRNT7)
STRT6   M1 = INPUT                                                  :F(FAIL)
        M1 'M1'                                                     :F(STRT6)
        M1 = INPUT                                                  :F(FAIL)
        M1 = INPUT                                                  :F(FAIL)
        M0 = INPUT                                                  :F(FAIL)
        M0 = INPUT                                                  :F(FAIL)
        M1 KIND =
        M1 KIND . INDEX
        I = 1
        P = 5
        P = EQ(INDEX,1) 12
        M = ARRAY(15,'')
        M<1> = M1
STRT7   I = LT(I,P) I + 1                                           :F(STRT11)
        C0 =
        C0 = EQ(I,4) '¬'
STRT8   COND = INPUT                                                :F(FAIL)
        IDENT(COND,'')                                              :S(STRT8)
        COND '#'                                                    :S(STRT7)
        COND SPAN(' ') =
        EQ(I,4)                                                     :S(STRT10)
STRT9   COND SPAN(' ') =                                            :S(STRT9)
STRT10  M<I> = M<I> COND C0                                         :(STRT8)
STRT11  M<3> SPAN(DIGITS) . CHAR ':' BREAK(',') . SAVE
+       = SAVE                                                      :S(STRT12)
        M<3> SPAN(DIGITS) . CHAR ':' BREAK(')') . SAVE
+       = SAVE                                                      :S(STRT12)
        M<11> SPAN(DIGITS) . CHAR ':' BREAK(',') . SAVE
+       = SAVE                                                      :S(STRT12)
        M<11> SPAN(DIGITS) . CHAR ':' BREAK(')') . SAVE
+       = SAVE                                                      :F(HEAD1)
```

```
STRT12   M<3> '(=' CHAR ')' = SAVE                                  :S(STRT12)
STRT13   M<11> '(=' CHAR ')' = SAVE                                 :S(STRT13)
+                                                                    F(STRT11)
*
*
*                    THE INTEGRAL OF A SPECIAL FUNCTION
*                    OR REMAINING PART OF THE INTEGRAL
*
*
HEAD1    RESULT = M<4>
         M7 = EQ(INDEX,1) M<9>                                      :F(BODY1)
         RESULT '¬' RPOS(0) =
         RESULT = LT(M7,1) RESULT ' + CONSTANT¬'
         GT(M7,0)                                                   :F(BODY1)
         RESULT POS(0) SPAN(' ') =
         C3 = '+'
         RESULT = LT(M7,4000) '(' RESULT ')'
         M<6> = '(' M<6> ')'
         RESULT POS(0) '(-' = '- ('
         M<6> POS(0) '(' '-' . C3 = '('
         M6 = M<7>
         M6 '(' BREAK(',') . C0 ',' BREAK(',') . C1 ','
+        BREAK(',') . C2 ','
         LT(M7,29)                                                  :F(HEAD3)
         RESULT = LT(M7,17) RESULT '*' TEXT3<M7 - 11>               :S(BACK1)
         J = 76
         J = GT(M7,22) 78
         M7 = GT(M7,22) M7 - 6
         K = M7 - 11
HEAD2    M7 = GT(M7,2) M7 - 2                                       :S(HEAD2)
         RESULT = RESULT '*' TEXT3<K> ' ' C3 ' ' M<6> '*'
+        TEXT3<M7 + J> '¬'                                          :(BACK2)
HEAD3    I = LT(M7,88) 48                                           :S(BACK1)
         I = LT(M7,105) 24                                          :S(BACK1)
         I = LT(M7,110) 28                                          :S(BACK1)
         I = LT(M7,800)                                             :F(HEAD4)
         SAVE = ' ' M7
         SAVE ' ' LEN(1) . I LEN(1) LEN(1) . J
         I = I * 96 - 31                                            :(BACK1)
HEAD4    I = LT(M7,900) 738                                         :S(BACK1)
         I = LT(M7,910) 828                                         :S(BACK1)
         I = LT(M7,960) 858                                         :S(BACK1)
         I = LT(M7,3000) 925                                        :S(BACK1)
         I = LT(M7,5000) 3919                                       :S(HEAD5)
         I = 4906
HEAD5    RESULT = DIFFER(RESULT,'0') RESULT ' ' C3 ' ' M<6> '*'
+        TEXT3<M7 - I> '¬'                                          :S(HEAD6)
         C3 = IDENT(C3,'+')
         C3 = IDENT(C3,'-') C3 ' '
         RESULT = C3 M<6> '*' TEXT3<M7 - I> '¬'
HEAD6    RESULT '#' = M<10>                                         :S(HEAD6)
         RESULT = '$' RESULT                                        :(BACK8)
*
*
*                    SUBSTITUTIONS
*
*
```

```
BACK1     RESULT SPAN(' ') =                                          :S(BACK1)
          RESULT = RESULT '*' TEXT3<M7 - I> '¬'
          RESULT '_' DIFFER(M<6>,'(0)') = ' ' C3 ' ' M<6> '*'
+                                                                     :S(BACK2)
          RESULT '_' BREAK('¬') =
BACK2     RESULT '#' = C0                                             :S(BACK2)
BACK3     RESULT '?' = C1                                             :S(BACK3)
BACK4     RESULT '$' = C2                                             :S(BACK4)
BACK5     RESULT '&' = M<10>                                          :S(BACK5)
          RESULT '*/' = '/'
BACK6     RESULT ':' = C0 - C1                                        :S(BACK6)
BACK7     RESULT '.' = C1 - C0                                        :S(BACK7)
          RESULT = '$' RESULT
BACK8     RESULT NOTANY(DIGITS LETTERS) . C0 '(' SPAN(DIGITS)
+         . C1 ')' = C0 C1                                            :S(BACK8)
BACK9     RESULT '*1' NOTANY(DIGITS) . C0 = C0                        :S(BACK9)
BACK10    RESULT NOTANY(DIGITS) . C0 '1*' = C0                        :S(BACK10)
          RESULT '¬' RPOS(0) =
          RESULT = RESULT ' + CONSTANT¬'
          RESULT '$' =
          GT(M7,3999)                                                 :F(NEST1)
*
*
*                    RETRIEVING DERIVATIVES
*
*
BODY1     M2 = ARRAY(12,0)
          M3 = ARRAY(12,0)
          LK = 1
          M<2> '(' =
          M<3> '(' =
          M<3> ')' RPOS(0) = ','
BODY2     M<2> SPAN(DIGITS) . SAVE =                                  :F(BODY3)
          M2<LK> = SAVE
          M<3> BREAK(',') . SAVE ',' =
          M3<LK> = SAVE
          LK = NE(M2<LK>,0) LT(LK,12) LK + 1                          :S(BODY2)
BODY3     LK = 1
          EQ(INDEX,1)                                                 :S(TAIL1)
          SAVE = M<5>
          M<5> '(' =
          SAVE '(' =
BODY4     EQ(M2<LK>,0)                                                :S(BODY9)
          EQ(M2<LK>,100)                                              :F(BODY6)
          M<5> BREAK(',') . C0 ',' =
BODY5     RESULT 'ZZ(' LK ')' = C0 '**(' M3<LK> ')*LN(' C0 ')'
+                                                                     :S(BODY5)
+                                                                      F(BODY8)
BODY6     RESULT 'ZZ(' LK ')' = TECH<M2<LK>>                          :S(BODY6)
BODY7     RESULT '#' = '(' M3<LK> ')'                                 :S(BODY7)
BODY8     LK = LT(LK,12) LK + 1                                       :S(BODY4)
BODY9     LK = 1
BODY10    EQ(M2<LK>,0)                                                :S(LAST1)
          EQ(M2<LK>,100)                                              :F(BODY12)
          SAVE BREAK(',') . C0 ',' =
BODY11    RESULT 'Z(' LK ')' = C0 '**(' M3<LK> ')'                    :S(BODY11)
+                                                                      F(BODY13)
```

```
:
BODY12   RESULT 'Z(' LK ')' = TEXT1<M2<LK>> '(' M3<LK> ')'   :S(BODY12)
BODY13   LK = LT(LK,12) LK + 1                               :S(BODY10)
+                                                             F(LAST1)
*
*
*                  RETRIEVING INTEGRAL OTHER THAN SPECIAL
*                  FUNCTIONS
*
*
TAIL1    SAVE = 0
TAIL2    M<12> SPAN(DIGITS) . C0 =
         M<12> SPAN(DIGITS) . C1 =
         EQ(C0,99)                                           :S(TAIL4)
         SAVE = SAVE + 1
TAIL3    RESULT 'ZZ(' SAVE ')' = TEXT1<C0> '(' C1 ')'        :S(TAIL3)
TAIL10   M<10> 'ZZ(' SAVE ')' = TEXT1<C0> '(' C1 ')'         :S(TAIL10)
+                                                            F(TAIL2)
TAIL4    RESULT 'Z2' = 'LOG(10)'                             :S(TAIL4)
         RESULT 'Z(6)' = '(' M<10> ')'                       :S(TAIL4)
TAIL5    RESULT 'Z(' SPAN(DIGITS) . M0 ')' = '#'             :F(NEST1)
TAIL6    RESULT 'Z(' M0 ')' = '#'                            :S(TAIL6)
         LT(M0,6)                                            :F(TAIL8)
         COND =
         COND = EQ(M<8>,1) LT(M0,3) 'H'
TAIL7    RESULT '#' = TEXT2<M0> COND '(' M<10> ')'           :S(TAIL7)
+                                                             F(TAIL5)
TAIL8    M0 = M0 - 6
TAIL9    RESULT '#' = TEXT1<M2<M0>> '(' M3<M0> ')'           :S(TAIL9)
+                                                             F(TAIL5)
*
*
*                  SUBSTITUTION PROCESS WHEN THE INTEGRAL
*                  INVOLVES NESTING OF FUNCTIONS
*
*
NEST1    RESULT 'Z3' = 'TAN(' M<10> ')'                      :S(NEST1)
         EQ(M<5>,0)                                          :S(LAST1)
         NEST = 85
NEST2    M<11> 'Z0' = '?'                                    :S(NEST2)
         TEXT5 M<5> SPAN(LETTERS) . M<5>                     :F(LAST1)
NEST3    RESULT 'Z0' = M<5> '(' M<11> ')'                    :S(NEST3)
NEST4    RESULT '?' = 'Z0'                                   :S(NEST4)
*
*
*                  RETRIEVING ORIGINAL VARIABLES FROM
*                  INTERNAL REPRESENTATIONS
*
*
LAST1    LETTERS LEN(IN) LEN(1) . SAVE
LAST2    RESULT 'Z0' = SAVE                                  :S(LAST2)
         LK = 1
LAST3    KN KIND . IN =                                      :F(PRNT1)
         EQ(IN,99)                                           :S(PRNT1)
         LETTERS LEN(IN) LEN(1) . SAVE
LAST4    RESULT 'Z1(' LK ')' = SAVE                          :S(LAST4)
         LK = LK + 1                                         :(LAST3)
*
```

```
*
*                    PRINT THE RESULT IN READABLE FORM
*
*
PRNT1     OUTPUT =
          OUTPUT = MESSAGE
PRNT2     RESULT BREAK('¬') . SAVE '¬' =                              :F(PRNT6)
          SAVE POS(0) SPAN(' ') =
          OUTPUT =
          OUTPUT = ' ' LT(SIZE(SAVE),125) SAVE                        :S(PRNT2)
          SAVE POS(0) (LEN(NEST) BREAK(' ') ' ' SPAN(OPERATORS)
+         ' ') . SAVE1 =                                              :F(PRNT3)
          LT(SIZE(SAVE1),110)                                         :S(PRNT5)
+                                                                      F(PRNT4)
PRNT3     SAVE1 = SAVE
          SAVE =
PRNT4     SAVE1 POS(0) (LEN(100) BREAK('+-')) . SAVE2
+         ('+' | '-') . C0 =
          OUTPUT = ' ' SAVE2 ' ' C0
          RESULT = SAVE1 SAVE RESULT                                  :(PRNT2)
PRNT5     RESULT = SAVE RESULT
          OUTPUT = ' ' SAVE1                                          :(PRNT2)
PRNT6     OUTPUT = DIFFER(RESULT,'')                                  :F(PRNT7)
          OUTPUT = ' ' RESULT
PRNT7     OUTPUT =
          OUTPUT =
          OUTPUT =
          OUTPUT =
          OUTPUT =                                                    :(STRT1)
FAIL      OUTPUT =
          OUTPUT =
          OUTPUT =
          OUTPUT =
          OUTPUT = '*** THE RUN WAS ABNORMALLY TERMINATED ***'
          OUTPUT = '*** DUE TO A SYSTEM ERROR WHICH OCCURED '
+                  'WHEN THE ABOVE PROBLEM OR THE ONE '
+                  'AFTER IT WAS BEING PROCESSED ***'
TRAC1     INPUT('INPUT',8,132)
TRAC2     MESSAGE = INPUT                                             :F(END)
          MESSAGE 'ERROR' SPAN(' ') SPAN(DIGITS) . ERRORNO            :F(TRAC2)
          OUTPUT =
          ERRORNO = LT(ERRORNO,30) GT(ERRORNO,19) 27                  :S(TRAC3)
          ERRORNO = LT(ERRORNO,50) GT(ERRORNO,39) 28                  :S(TRAC3)
          ERRORNO = LT(ERRORNO,60) GT(ERRORNO,49) 29                  :S(TRAC3)
          ERRORNO = LT(ERRORNO,90) GT(ERRORNO,69) 30                  :S(TRAC3)
          ERRORNO = LT(ERRORNO,150) GT(ERRORNO,139) 31                :S(TRAC3)
          ERRORNO = EQ(ERRORNO,727) 32                                :S(TRAC3)
          ERRORNO = EQ(ERRORNO,737) 33                                :S(TRAC3)
          ERRORNO = EQ(ERRORNO,747) 34                                :S(TRAC3)
          ERRORNO = LT(ERRORNO,2000) GT(ERRORNO,999) 35               :S(TRAC3)
+                                                                      F(END)
TRAC3     OUTPUT =
          OUTPUT = '*** ' ERROR<ERRORNO> ' ***'                       :(END)
EXIT      OUTPUT =
          OUTPUT =
          OUTPUT =
          OUTPUT =
```

```
      OUTPUT = '***** ALL PROBLEMS ATTEMPTED *****'
END
```

APPENDIX D

PROCEDURE DIFF, PINT, SPLIT(ALTRAN ROUTINES)

```
                    PROCEDURE DIFF(FCT,X)
```

THIS PROCEDURE COMPUTES THE PARTIAL DERIVATIVE OF THE FUNCTION "FCT" WITH RESPECT TO THE VARIABLE "X". THE FUNCTION "FCT" SHOULD BE A FUNCTION OF SIMPLE VARIABLES WHICH ARE INDETERMINATES DECLARED IN A DECLARATION STATEMENT. "FCT" SHOULD NOT CONTAIN ANY REAL-VALUED QUANTITY. IF "FCT" DOES NOT SATISFY ANY OF THESE CONDITIONS TTHEN AN ALTRAN SYSTEM ERROR WILL OCCUR.
GIVEN THE FOLLOWING FUNCTION

```
                FCT = (4-X) / (X*C)
```

THEN THE CALL DIFF(FCT,X) YIELDS

```
                    - 4 / (X**2*C)
```

```
                    PROCEDURE PINT(FCT,X)
```

THIS PROCEDURE COMPUTES THE INTEGRAL OF THE FUNCTION "FCT" WITH RESPECT TO "X". THE FUNCTION "FCT" SHOULD BE A POLYNOMIAL WITH RATIONAL COEFFICIENTS AND IN SIMPLE VARIABLES WHICH ARE INDETERMINATES DECLARED IN A DECLARATION STATEMENT. IF THESE CONDITIONS ARE NOT SATIFIED THEN AN ALTRAN SYSTEM ERROR WILL OCCUR.
GIVEN THE FOLLOWING FUNCTION

```
                FCT = (1-X**2+W*X) / 5
```

THEN THE CALL PINT(FCT,X) YIELDS

```
                    (30*X-2*X**3+3*W*X**2) / 30
```

```
                    PROCEDURE SPLIT(D1,X,D2,D3)
```

THIS PROCEDURE PLACES THE FACTOR OF THE ALGEBRAIC EXPRESSION D1 WHICH IS INDEPENDENT OF X IN D2 AND PLACES THE REST IN D3
GIVEN THE FOLLOWING FUNCTION

```
                    D1 = 4*(X-1)**3*(C*B-12**3*Z)
```

THEN THE CALL SPLIT(D1,X,D2,D3) YIELDS

```
                    D2 = 4*(C*B-12**3*Z)
```

AND

```
                    D3 = (X-1)**3
```

APPENDIX E

FUNCTION NAMES AND THEIR REPRESENTATIONS

Function name	Representation
SIN	1
COS	2
CSC	3
SEC	4
TAN	5
COT	6
ASIN	7
ACOS	8
ACSC	9
ASEC	10
ATAN	11
ACOT	12
SINH	13
COSH	14
CSCH	15
SECH	16
TANH	17
COTH	18
ASINH	19
ACOSH	20
ACSCH	21
ASECH	22
ATANH	23
ACOTH	24
EXP	26
LN	27
SQRT	28
LOG	29

APPENDIX F

SPECIAL FUNCTIONS (FORM 15)

In the following integrals A and B are non-zero integers; C and D are functions of the variable of integration. G can be the identity function, LN, EXP, SQRT, any trigonometric function or any hyperbolic function. OP, OP1, OP2 denote both SIN and COS, and OP1 and OP2 are always distinct when they appear in the same function. Finally, if the function OP,OP1,OP2, G or U occurs more than once in any of the following expressions, each occurrence denotes the same function.

1. $$\int \frac{G'(U)}{(A + B\ OP(G'(U))\ )}$$

2. $$\int \frac{G'(U)\ OP(G(U))}{(A + B\ OP(G(U))\ )}$$

3. $$\int \frac{G'(U)}{(OP(G(U))\ (A + B\ OP(G(U)))\ )}$$

4. $$\int \frac{G'(U)}{(A + B\ OP(G(U))\ )^2}$$

5. $$\int \frac{G'(U)\ OP(G(U))}{(A + B\ OP(G(U))\ )^2}$$

6. $$\int \frac{G'(U)}{(C^2 \pm D^2\ OP^2(G(U))\ )}$$

7. $$\int \frac{G'(U)\ \ OP(G(U))}{SQRT(1 \pm C^2 SIN^2(G(U)))}$$

8. $$\int G'(U)\ \ OP(G(U))\ \ SQRT(1 \pm C^2 SIN^2(G(U)))$$

9. $$\int \frac{G'(U)\ \ OP1(G(U))}{(\ 1 \pm OP2(G(U))\ )}$$

10. $$\int \frac{G'(U)}{(\ OP1(G(U))\ \ (1 \pm OP2(G(U))))}$$

11. $$\int \frac{G'(U)\ \ OP1(G(U))}{(\ OP2(G(U))\ \ (1 \pm OP2(G(U))))}$$

12. $$\int \frac{G'(U)\ \ OP1(G(U))}{(\ OP2(G(U))\ \ (1 \pm OP1(G(U))))}$$

13. $$\int \frac{G'(U)}{(\ OP1(G(U))\ \pm OP2(G(U))\ )}$$

14. $$\int \frac{G'(U)\ \ OP1(G(U))}{(\ OP1(G(U))\ \pm OP2(G(U))\ )}$$

15. $$\int \frac{G'(U)\ \ OP1(G(U))}{(\ OP2(G(U))\ \pm OP1(G(U))\ )}$$

16. $$\int \frac{G'(U)}{(\ OP1(G(U))\ \pm OP2(G(U))\ )^2}$$

17. $$\int \frac{G'(U)}{(\ A\ \ OP1(G(U))\ + B\ \ OP2(G(U))\ )}$$

18. $$\int \frac{G'(U)}{(\ C^2 + OP1^2(G(U))\ \pm D^2\ \ OP2^2(G(U))\ )}$$

19. $$\int G'(U)\ \ OP(LN(G(U)))$$

APPENDIX G

THE SOLUTIONS OF THE SET OF 30 TEST PROBEEMS

```
PROBLEM 1
 DIFFERENTIATE SQRT(X+SQRT(X**2+3))
 WITH RESPECT TO X
THE RESULT IS :
 (1/(2*SQRT(SQRT(X**2+3)+X))) * ( 2*X*(1/(2*SQRT(X**2+3))) + 1 )


PROBLEM 2
 DIFFERENTIATE LN(LN(X))
 WITH RESPECT TO X
THE RESULT IS :
 (1/(LN(X)))*(1/(X))


PROBLEM 3
 DIFFERENTIATE ATAN(SQRT(X))
 WITH RESPECT TO X
THE RESULT IS :
 (1/(1+(SQRT(X))**2))*(1/(2*SQRT(X)))


PROBLEM 4
 DIFFERENTIATE ASIN(X)/LN(X)
 WITH RESPECT TO X
THE RESULT IS :
 - ( ASIN(X)*(1/(X)) - LN(X)*(1/SQRT(1-(X)**2)) ) / ( LN(X)**2 )


PROBLEM 5
 DIFFERENTIATE (X**3-3*X**2+1)/(X**2+1)
 WITH RESPECT TO X
THE RESULT IS :
 X * ( X**3 + 3*X - 8 ) / ( X**2 + 1 )**2
```

```
PROBLEM 6
 DIFFERENTIATE SQRT(X+2)**5*SQRT(X-1)
 WITH RESPECT TO X
THE RESULT IS :
 SQRT(X+2)**4 * ( SQRT(X+2)*(1/(2*SQRT(X-1))) + 5*SQRT(X-1)*(1/(2

PROBLEM 7
 DIFFERENTIATE ((X**2-X+1)/(X+1))**2
 WITH RESPECT TO X
THE RESULT IS :
 2 * ( X**2 - X + 1 ) * ( X**2 + 2*X - 2 ) / ( X + 1 )**3

PROBLEM 8
 DIFFERENTIATE X**4+3*X**2-27
 WITH RESPECT TO X
THE RESULT IS :
 2*X * ( 2*X**2 + 3 )

PROBLEM 9
 DIFFERENTIATE SIN(2**X)
 WITH RESPECT TO X
THE RESULT IS :
 COS(2**(X))*2**(X)*LN(2)

PROBLEM 10
 INTEGRATE 1/(X*SQRT(X**2-1))
 WITH RESPECT TO X
THE RESULT IS :
 ASEC(X) + CONSTANT
```

```
PROBLEM 11
 INTEGRATE X*EXP(-X)
 WITH RESPECT TO X
THE RESULT IS :
 EXP(-X) * ( (-X) - 1 ) + CONSTANT


PROBLEM 12
 INTEGRATE 2*X*SQRT(X**2+1)
 WITH RESPECT TO X
THE RESULT IS :
 2*SQRT(X**2+1)**3 / 3 + CONSTANT


PROBLEM 13
 INTEGRATE (SEC(X))**2
 WITH RESPECT TO X
THE RESULT IS :
 SIN(X) / ( COS(X) ) + CONSTANT


PROBLEM 14
 INTEGRATE CSC(X)*COT(X)
 WITH RESPECT TO X
THE RESULT IS :
 - COS(X)**2 / ( SIN(X) ) - SIN(X) + CONSTANT


PROBLEM 15
 INTEGRATE SEC(X)
 WITH RESPECT TO X
THE RESULT IS :
 LN(SEC(X) + TAN(X)) + CONSTANT
```

```
PROBLEM 16
 INTEGRATE CSC(X)
 WITH RESPECT TO X
THE RESULT IS :
 - LN(CSC(X) + COT(X)) + CONSTANT


PROBLEM 17
 INTEGRATE SEC(X)*TAN(X)
 WITH RESPECT TO X
THE RESULT IS :
 SIN(X)**2 / ( COS(X) ) + COS(X) + CONSTANT


PROBLEM 18
 INTEGRATE X*ATAN(X)
 WITH RESPECT TO X
THE RESULT IS :
 ( ATAN(X)*X**2 + ATAN(X) - X ) / 2 + CONSTANT


PROBLEM 19
 INTEGRATE 1/(X-1)**2
 WITH RESPECT TO X
THE RESULT IS :
 - 1 / ( X - 1 ) + CONSTANT


PROBLEM 20
 INTEGRATE 1/SQRT(1-X**2)
 WITH RESPECT TO X
THE RESULT IS :
 ASIN(X) + CONSTANT
```

```
PROBLEM 21
 INTEGRATE 1/(X*LN(X)**2)
 WITH RESPECT TO X
THE RESULT IS :
 - 1 / ( LN(X) ) + CONSTANT


PROBLEM 22
 INTEGRATE SQRT(1-X**2)
 WITH RESPECT TO X
THE RESULT IS :
 ( ASIN(X) + COS(ASIN(X))*X ) / 2 + CONSTANT


PROBLEM 23
 INTEGRATE 4*X**3+7
 WITH RESPECT TO X
THE RESULT IS :
 X * ( X**3 + 7 ) + CONSTANT


PROBLEM 24
 INTEGRATE TAN(X)
 WITH RESPECT TO X
THE RESULT IS :
 - LN(COS(X)) + CONSTANT


PROBLEM 25
 INTEGRATE 2/(5*X**2)
 WITH RESPECT TO X
THE RESULT IS :
 - 2 / ( 5*X ) + CONSTANT
```

```
PROBLEM 26

 INTEGRATE 1/((X-1)*(X**2-3))

 WITH RESPECT TO X

*** THE INTEGRAND IS BEYOND PACKAGE CAPABILITY ***
*** THE PROBLEM WAS CANCELLED ***
*** THE DENOMINATOR IS OUT OF THE PACKAGE RANGE ***


PROBLEM 27

 INTEGRATE 1/(1+X**2)

 WITH RESPECT TO X

THE RESULT IS :

 ATAN(X) + CONSTANT


PROBLEM 28

 INTEGRATE X/SQRT(X+2)**3

 WITH RESPECT TO X

THE RESULT IS :

 2 * ( SQRT(X+2)**2 + 2 ) / ( SQRT(X+2) ) + CONSTANT


PROBLEM 29

 INTEGRATE X*SIN(X**2)**3*COS(X**2)**3

 WITH RESPECT TO X

THE RESULT IS :

 - ( 9*COS(2*X**2) - COS(6*X**2) ) / 384 + CONSTANT


PROBLEM 30

 INTEGRATE 10**X

 WITH RESPECT TO X

THE RESULT IS :

 EXP(X*LN(10)) / ( LN(10) ) + CONSTANT


***** ALL PROBLEMS ATTEMPTED *****
```

```
PROBLEM 1
 DIFFERENTIATE SIN(COS(X-20*X**3+45*X+CSC(X)))
 WITH RESPECT TO X
THE RESULT IS :
 - COS(COS(CSC(X)-20*X**3+46*X))*(-SIN(CSC(X)-20*X**3+46*X)) * ( 60*X**2 - (-CSC(X)*COT(X)) - 46 )


PROBLEM 2
 DIFFERENTIATE 1/SQRT(X**2-X+3)
 WITH RESPECT TO X
THE RESULT IS :
 - (1/(2*SQRT(X**2-X+3))) * ( 2*X - 1 ) / ( SQRT(X**2-X+3)**2 )


PROBLEM 3
 DIFFERENTIATE SIN(4-X**2+A*X)+COS(1-SIN(X))
 WITH RESPECT TO X
THE RESULT IS :
 - ( 2*X*COS(-(X**2-X*A-4)) - A*COS(-(X**2-X*A-4)) + (-SIN(-(SIN(X)-1)))*COS(X) )


PROBLEM 4
 DIFFERENTIATE SIN(X)*COS(X**2-3+SIN(X))
 WITH RESPECT TO X
THE RESULT IS :
 2*SIN(X)*X*(-SIN(SIN(X)+X**2-3)) + SIN(X)*(-SIN(SIN(X)+X**2-3))*COS(X) + COS(SIN(X)+X**2-3)*COS(X)
```

```
PROBLEM 5
 DIFFERENTIATE 10**(SIN(X-COS(1-X**2)+X*A**2)+10*A*X*R)
 WITH RESPECT TO X
THE RESULT IS :
 10**(SIN(-(COS(-(X**2-1))-X*A**2-X))+10*X*A*R)*LN(10) * ( 2*X*COS(-(COS(-(X**2-1))-X*A**2-X))*(-SIN( -
 (X**2-1))) + A**2*COS(-(COS(-(X**2-1))-X*A**2-X)) + 10*A*R + COS(-(COS(-(X**2-1))-X*A**2-X)) )


PROBLEM 6
 INTEGRATE X**2*SIN(X**3-1)*SQRT(1-2**2*SIN(X**3-1)**2)
 WITH RESPECT TO X
THE RESULT IS :
 -(1/6)*COS(X**3-1)*SQRT(1-4*SIN(X**3-1)**2) + (1/4)*LOG(2*COS(X**3-1)+SQRT(1-4*SIN(X**3-1)**2)) + CONSTANT


PROBLEM 7
 INTEGRATE SIN(X)*SINH(COS(X))**3*COSH(COS(X))**3
 WITH RESPECT TO X
THE RESULT IS :
 ( 9*COSH(2*COS(X)) - COSH(6*COS(X)) ) / 192 + CONSTANT


PROBLEM 8
 INTEGRATE (X**6-X+45*X**5+9)/(X**2-7)
 WITH RESPECT TO X
THE RESULT IS :
 ( 66120*LN((X**2-7))*SQRT(7) + 21120*ATANH(X/(SQRT(7))) + 12*SQRT(7)*X**5 + 675*SQRT(7)*X**4 +
 140*SQRT(7)*X**3 + 9450*SQRT(7)*X**2 + 2940*SQRT(7)*X ) / ( 60*SQRT(7) ) + CONSTANT
```

```
PROBLEM 9
 INTEGRATE (4-X**3+X**6+X**5)/X**7
 WITH RESPECT TO X
THE RESULT IS :
 ( 3*LN(X)*X**6 - 3*X**5 + X**3 - 2 ) / ( 3*X**6 ) + CONSTANT


PROBLEM 10
 INTEGRATE SIN(X**2-X+1)*COS(X**2-X+1)*COSH(COS(X**2-X+1))*(2*X-1)
 WITH RESPECT TO X
THE RESULT IS :
 - ( SINH(COS(X**2-X+1))*(COS(X**2-X+1)) - COSH(COS(X**2-X+1)) ) + CONSTANT


PROBLEM 11
 INTEGRATE (X-1)**5*LN(X-1)**3
 WITH RESPECT TO X
THE RESULT IS :
 (X-1)**6 * ( 36*LN(X-1)**3 - 18*LN(X-1)**2 + 6*LN(X-1) - 1 ) / 216 + CONSTANT


PROBLEM 12
 INTEGRATE EXP(X-U)**5*EXP(EXP(X-U))
 WITH RESPECT TO X
THE RESULT IS :
 EXP(EXP(X-U)) * ( (EXP(X-U))**4 - 4*(EXP(X-U))**3 + 12*(EXP(X-U))**2 - 24*(EXP(X-U)) + 24 ) + CONSTANT
```

```
PROBLEM 13
 INTEGRATE COS(X-1)*SIN(X-1)**2*LN(SIN(X-1)-SQRT(5**2+SIN(X-1)**2))
 WITH RESPECT TO X
THE RESULT IS :
 ( 3*LN(-(SQRT(SIN(X-1)**2+25)-SIN(X-1)))*(SIN(X-1))**3 + 5*COSH(ASINH(SIN(X-1)/5))*SIN(X-1)**2 ) / 9  -
  (250/9)*COSH(SIN(X-1)/5) + CONSTANT


PROBLEM 14
 INTEGRATE SIN(X**2-X+1)**3*(X**2-X+1)**2*(2*X-1)
 WITH RESPECT TO X
THE RESULT IS :
 ( 162*SIN(X**2-X+1)*(X**2-X+1) - 81*COS(X**2-X+1)*(X**2-X+1)**2 + 162*COS(X**2-X+1) - 6*SIN(3*(X**2-X+1))*X**2 +
 6*SIN(3*(X**2-X+1))*X - 6*SIN(3*(X**2-X+1)) + 9*COS(3*(X**2-X+1))*X**4 - 18*COS(3*(X**2-X+1))*X**3 +
 27*COS(3*(X**2-X+1))*X**2 -18*COS(3*(X**2-X+1))*X + 7*COS(3*(X**2-X+1)) ) / 108 + CONSTANT


PROBLEM 15
 INTEGRATE SIN(X-1)*COS(X-1)**2*SQRT(5**2-COS(X-1)**2)**3
 WITH RESPECT TO X
THE RESULT IS :
 - 3125 * ( 15*ASIN(COS(X-1)/5) - 8*COS(ASIN(COS(X-1)/5))**5*COS(X-1) + 2*COS(ASIN(COS(X-1)/5))**3*COS(X-1) +
 3*COS(ASIN(COS(X-1)/5))*COS(X-1) ) / 48 + CONSTANT


PROBLEM 16
 INTEGRATE SIN(X-1)**5*EXP(X-1)
 WITH RESPECT TO X
THE RESULT IS :
 EXP((X-1)) * ( SIN(X-1)**5 - 5*SIN(X-1)**4*COS(X-1) + 2*SIN(X-1)**3 - 6*SIN(X-1)**2*COS(X-1) +
 6*SIN(X-1) - 6*COS(X-1) ) / 26 + CONSTANT
```

```
PROBLEM 17
 INTEGRATE 10/(SIN(X)+COS(X)+5)
 WITH RESPECT TO X
THE RESULT IS :
 5*ATAN((4*TAN(X/2)+1)/(4*SQRT(23/16))) / ( SQRT(23/16) ) + CONSTANT


PROBLEM 18
 INTEGRATE SINH(X)*COSH(4*X)*EXP(10*X)
 WITH RESPECT TO X
THE RESULT IS :
 EXP(10*X) * ( 182*SINH(5*X) - 91*COSH(5*X) + 150*SIN(-3*X) + 45*COS(-3*X) ) / 2730 + CONSTANT



***** ALL PROBLEMS ATTEMPTED *****
```

APPENDIX I

JCL REQUESTED OF THE SYMDIP USER

The package SYMDIP can be made available with following JCL setup :

```
//JOBNAME JOB (ACC. INFORMATION),CLASS=C,TIME=5
// EXEC SYMDIP
//INPUT DD *
      .
      .
      .      Your input problems
      .
      .
      .
//
```

APPENDIX J

THE COMPLETE JCL FOR SYMDIP

```
//INTEDIFF PROC
//GO EXEC PGM=SNOBOL4
//STEPLIB DD DSN=SYS1.USERLIB,DISP=SHR
//FT02F001 DD DSN=&&CODE,DISP=(,PASS),UNIT=SYSDA,
// SPACE=(400,(100,10)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)
//FT03F001 DD DDNAME=INPUT
//FT05F001 DD DSN=ANALIZER.SNOBOL4,DISP=OLD
//FT06F001 DD DUMMY
//FT07F001 DD DUMMY
//FT10F001 DD DSN=&&INPUT,DISP=(,PASS),UNIT=SYSDA,
// SPACE=(400,(100,10)),DCB=(RECFM=FB,LRECL=400,BLKSIZE=400)
//STEP2 EXEC PGM=SYMDIP
//STEPLIB DD DSN=PACKAGE,DISP=SHR
//FT02F001 DD DSN=&&CODE,DISP=(OLD,DELETE)
//FT03F001 DD DSN=&&RESULT,DISP=(,PASS),UNIT=SYSDA,
// SPACE=(400,(100,10)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)
//FT06F001 DD DUMMY
//FT08F001 DD DSN=&&ERROR,DISP=(,PASS),UNIT=SYSDA,
// SPACE=(400,(100,10)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=400)
//FT20F001 DD DUMMY
//STEP3 EXEC PGM=SNOBOL4
//STEPLIB DD DSN=SYS1.USERLIB,DISP=SHR
//FT03F001 DD DSN=&&RESULT,DISP=(OLD,DELETE)
//FT05F001 DD DSN=WRITER.SNOBOL4,DISP=OLD
//FT06F001 DD DUMMY
//FT07F001 DD DUMMY
//FT08F001 DD DSN=&&ERROR,DISP=(OLD,DELETE)
//FT09F001 DD SYSOUT=A
//FT10F001 DD DSN=&&INPUT,DISP=(OLD,DELETE)
```

APPENDIX K

THE PROGRAM FOR CREATING OVERLAY LEVELS

```
C
C   THIS PROGARM RETURNS LEVELS OF ROUTINES
C   EACH OF THESE LEVELS CONTAINS ROUTINES THAT ARE MUTUALLY INDEPENDENT
C   THE LEVELS ARE CREATED IN SUCH A WAY THAT A ROUTINE IN THE HIGHER LEV 3 C N
C   ONLY CALL ROUTINES IN THE LOWER LEVELS WITH AT LEAST ONE OF THE CALLED
C   ROUTINES BEING IN THE LEVEL IMMEDIATELY BELOW IT
C   TWO SUBROUTINES ARE USED TO ASSIST CREATING SUCH LEVELS
C   SUBPROCEDURE CROSS WILL CROSS OFF A ROW IN THE MATRIX
C   WHEN THE SUBROUTINE IT REPRESENTS IS ADDED TO THE CURRENT LEVEL
C   SUBROUTINE ADJUST COMPRESSES THE ELEMENTS ON A ROW WHERE A 0 APPEARS
C
      DIMENSION A(230,19),C(19),LEVEL(60),STRING(6),MOM(4),LENS(5)
      INTEGER A,B,C
      WRITE(6,90)
      DO 2 I = 1 , 228
            READ (5 , 10) C
            WRITE(6,15) I,C
            DO 1 J = 1 , 19
                  A(I,J) = C(J)
    1       CONTINUE
    2 CONTINUE
      READ(5,7) (LEVEL(J),J=30,50)
      DO 4 I = 1 , 24
            K = 0
            IF(I .EQ. 1) GO TO 1111
            DO 999 LOL = 1 , 60
                  LEVEL(LOL) = 0
  999       CONTINUE
            IF(I .EQ. 5) GO TO 888
 1111       DO 3 J = 1 , 228
                  IF(A(J,1).NE.0) GO TO 3
                  K = K + 1
                  LEVEL(K) = J
                  A(J,1) = -1
    3       CONTINUE
            GO TO 666
  888       READ(5,30) (LEVEL(J),J=1,14)
            DO 777 KK = 1 , 14
                  JJ = LEVEL(KK)
                  A(JJ,1) = -1
  777       CONTINUE
            K = 14
  666       WRITE(6,5) I
            IF(I .EQ. 1) K = K + 21
            WRITE(6,6) K
            WRITE(6,60)
            DO 555 J = 1 , K
                  L = LEVEL(J)
                  READ(5,80) STRING
                  READ(5,85) MOM
                  WRITE(6,50) L,STRING,MOM
  555       CONTINUE
            CALL CROSS (A,LEVEL,K)
            READ(5,65) LENS
            WRITE(6,70) LENS
    4 CONTINUE
    5 FORMAT('1','THIS IS LEVEL NO',I5,'  THE TOTAL SUBROUTINES  ')
    6 FORMAT('0','IN IT IS',I5)
    7 FORMAT('0',21I3)
   10 FORMAT('0',4X,18I4,I2)
   15 FORMAT('0',I5,4X,19I5)
   30 FORMAT('0',14I3)
   50 FORMAT('0',1X,I5,20X,6A1,15X,4A1)
   60 FORMAT('0','SUBROUTINES',15X,'NAMES',15X,'LENGTHS')
   65 FORMAT('0',5A1)
   70 FORMAT('0','THE TOTAL LENGTH OF THIS LEVEL IS   ',5A1)
   80 FORMAT('0',15X,6A1)
   85 FORMAT('0',4A1)
   90 FORMAT('1','THIS FOLLOWING IS THE MATRIX OF CALLS')
      END
```

```
      SUBROUTINE CROSS (A,LEVEL,LVELNO)
      DIMENSION A(230,19),LEVEL(60)
      INTEGER A
      DO 4 I = 1 , 228
           IF (A(I,1) .EQ. -1) GO TO 4
           LAST = 1
           DO 3 K = 1 , LVELNO
                DO 1 L = LAST , 19
                     IF(A(I,L) .EQ. 0) GO TO 4
                     IF(A(I,L) .GT. LEVEL(K)) GO TO 2
                     IF(A(I,L) .NE. LEVEL(K)) GO TO 1
                     A(I,L) = 0
                     CALL ADJUST(A,I,L)
                     L = L - 1
                     GO TO 2
1               CONTINUE
2          LAST = L
3          CONTINUE
4     CONTINUE
      RETURN
      END



      SUBROUTINE ADJUST(A,I,L)
      DIMENSION A(230,19)
      INTEGER A
      IF(L .EQ. 19) RETURN
      L = L + 1
      DO 1 J = L , 19
           K = J - 1
           A(I,K) = A(I,J)
           IF (A(I,J) .EQ. 0) RETURN
1     CONTINUE
      RETURN
      END
```

VITA

Surname: WANG Given names: KEK-WAN

Place of Birth: SINGAPORE Date of Birth: NOV. 10, 1946

Educational Institutions Attended, with Dates of Entering and Leaving:

NANYANG UNIVERSITY (SINGAPORE) 1965 to 1970

TEACHERS' TRAINING COLLEDGE (SINGAPORE) 1971 to 1972

UNIVERSITY OF VICTORIA, B.C. 1972 to 1976

Degrees, Diplomas, Etc., Awarded, with Dates and Names of Institutions:

B.Sc. (Hons.)	1970	NANYANG UNIVERSITY
Diploma in Education	1972	TEACHERS' TRAINING COLLEDGE

Honors and Awards:

The Singapore Government University Scholarship, 1965-1970

University of Victoria Fellowship, 1974-1976

Appointments with Dates and Names of Employers

Ministry of Education (Singapore) , 1971-1974

Publications:

PARTIAL COPYRIGHT LICENSE

I hereby grant to the University of Victoria the right to lend my thesis or dissertation (the title of which is shown below) to users in the University of Victoria Library, and to make single copies only in response to a written request from the library of any other university, or similar institution, on its own behalf or for one of its users. For this service, a fee may be collected by the University of Victoria to cover the bare costs of reproduction. It is expressly understood that there will be no multiple copying, nor will any copies be sold at a profit. This license will continue in effect until further notice from me.

Title of Thesis/Dissertation:

SYMDIP: A computer Program for Symbolic Differentiation and Integration

Author: [redacted]
(signature)

WANG KEK WAN
(name)

28/9/76
(date)