Title

GENERATION OP GENERALIZED RUNGE-KUTTA INTERGRATION METHODS FOR n-th ORDER SYSTEMS OP p-th ORDER ORDINARY DIFFERENTIAL EQUATIONS

Author

Close, Elon Ryder.

Publication Date

1968-12-11

UCRL-18650

cy. 2

GENERATION OF GENERALIZED RUNGE-KUTTA INTEGRATION METHODS FOR n-th ORDER SYSTEMS OF p-th ORDER ORDINARY DIFFERENTIAL EQUATIONS

Elon Ryder Close
(Ph. D. Thesis)

December 11, 1968

LAWRENCE RADIATION LABORATORY
UNIVERSITY of CALIFORNIA BERKELEY

Cy. 2

UCRL-18650

DISCLAIMER

UCRL-18650

UNIVERSITY OF CALIFORNIA

Lawrence Radiation Laboratory
Berkeley, California

AEC Contract No. W-7405-eng-48

GENERATION OF GENERALIZED RUNGE-KUTTA INTEGRATION METHODS FOR n-th ORDER SYSTEMS OF p-th ORDER ORDINARY DIFFERENTIAL EQUATIONS

Elon Ryder Close
(Ph. D. Thesis)

December 11, 1968

TABLE OF CONTENTS

GENERATION OF GENERALIZED RUNGE-KUTTA INTEGRATION METHODS FOR n-th ORDER SYSTEMS OF p-th ORDER ORDINARY DIFFERENTIAL EQUATIONS

Elon Ryder Close

Lawrence Radiation Laboratory
University of California
Berkeley, California

December 11, 1968

ABSTRACT

We treat the generation of schemes for the numerical integration of the initial value problem $D^p x = X \circ \xi$, $\xi(a) = b$ where $x \in R \to R^n$, $\xi = (D^p x, \ldots, x) \in R \to R^{n \times p}$, $X \in R^{r \times p} \to R^n$, $b \in R^{r \times p}$, $a \in R$, R is the real line, $R^{n \times p}$ and R^n are real $n \times p$ and n dimensional vector spaces. Scheme definitions are provided that include the classical Runge-Kutta (RK) and finite difference schemes and, at the same time, are generalizations which provide schemes outside this class. Use is made of the first derivative DX. A global view of the scheme generation problem is provided by developing a formalism that makes use of the concept of differentials. It is shown that all basic results can be obtained from one recursive definition of a generic operator Y and the generic z and y which lie in its domain and range. A constructive means for obtaining specific schemes is developed and an ALGOL 60 program which performs this work is presented. The use of these results is illustrated on a classical example (fourth order RK) and on the simplest class of generalized RK schemes. The generalized RK schemes presented are believed to be new.

I. INTRODUCTION

The work presented in the following chapters and their associated appendices has as its objective the obtaining of schemes for the numerical integration of n-th order systems of p-th order ordinary differential equations written as an initial value problem in the form*

$$\begin{aligned} D^p x &= X \circ \xi \\ \xi(a) &= b \end{aligned} \tag{1}$$

where $x \in R \to R^n$, $\xi = (D^{p-1}x, \ldots, x) \in R \to R^{n\times p}$, $X \in R^{n\times p} \to R^n$, $b \in R^{n\times p}$, and $a \in R$. We take R to be the real line and $R^{n\times p}$ and R^n to be, respectively, $n \times p$ and n-dimensional vector spaces. Since our principal tool used in the analysis is the classical Taylor's series, we shall always implicitly assume that the functions discussed possess enough derivatives to carry out the analysis.

The schemes considered are given precise definition as they are introduced. However, they can loosely be described as follows. Given an interval of integration, we have, or will construct in the manner described below, a set of approximations ξ_i to the true solution $\xi(t_i)$. We also have, or can obtain, the function values $X_i = X(\xi_i)$ and if we wish, we can also obtain derivatives of X; for example, here we consider $DX(\xi_i)$. We define our schemes by saying that any approximation is a linear combination of these quantities or of quantities derivable from them. Thus,

$$\xi_i = \Sigma\, \alpha_j\, \xi_j + \Sigma\, \beta_j\, X_j \tag{2}$$

or

$$\xi_i = \Sigma\, \alpha_j\, \xi_j + \Sigma\, \beta_j\, X_j + \Sigma\, \gamma_j\, \eta_j \tag{3}$$

*See Appendix V.

where the coefficient matrices are, of course, assumed to be suitably defined. In the second case, the η_j are, essentially, derived by multiplying the derivative matrix $DX(\xi_j)$ times some previously constructed quantity.

The first of these we call generalized Runge-Kutta (RK) schemes since they contain those classical schemes and the second of these we call generalized Runge-Kutta-Frey (RKF) schemes since they make use of the first derivative DX of X as introduced by Frey. Both of these schemes lead to parameter defining equations that are similar in form to those associated with the classical Runge-Kutta schemes.

In our work, we say nothing about the accuracy of any approximation. We simply assume that to the set of approximations there is associated a lower bound such that all approximations are at least that accurate and it is implicitly assumed that there is at least one approximation for which we shall try and minimize the error $\xi_o - \xi(t_o) = \epsilon_o$. This unrestricted approach to the specification of the accuracy of an approximation allows the inclusion of finite difference schemes (all ξ_i have an equal order of accuracy) and classical RK schemes (we build from the lowest order 1 to the highest order desired at each step) in our definition.

Since the obtaining of any one integration scheme of any complexity can become a formidable task when working with systems of equations, we have devoted a considerable amount of our effort to formalizing and systematizing this task to such a degree that in using our results we can easily obtain the parameter defining equations for any given scheme in the class of schemes considered and, further, for generalized RK schemes, we can obtain an overall view of what these equations will look like without defining a specific scheme.

We thus, as a result of this work, have at our disposal a global view of the parameter equations, a view that allows us to anticipate the character of the equations that we must solve for any particular scheme. We also have a local, direct constructive approach to these equations. Using this latter approach, we have built procedures that will actually perform the algebraic work involved and present the investigator with the equations associated with a specific scheme.

Since the formal structure that we derive and the associated procedures which preserve this structure are valid for a whole class of schemes, we speak of the generation of integration schemes. We mean this in a sense quite analogous to what one means when speaking, say, of the generation of a value of the sine function. In the context of a machine-oriented calculation, one does not look up the value of the sine, nor does one derive the expansions and calculate the value of the function (one doesn't derive the result); instead, the value is generated by a well-defined algorithm that is preserved in the form of a system procedure.

One of our basic results is that, to a large degree, our work has this character; it is preserved in the form of suitably defined ALGOL procedures that are presented here and the degree to which this is true can be readily increased.

Our work, as presented here, can be considered as an extension of the work presented by Butcher[1], Ceschino-Kuntzmann[2], and R. DeVogelaere[3]. It will be obvious to anyone familiar with Butcher's work that much of our theoretical development is influenced by what he has done. His work has been the principal inspiration in the presentation of our results in terms of recursive definitions of quantities which are

then proved to be the coefficients of the expansions that we wish to have. The work of Ceschino-Kuntzmann has proved invaluable in suggesting the form and direction that our scheme definitions and results should take. In fact, in their work and the way in which it is presented, they have come extremely close to presenting the generalized schemes that we present here. I am quite indebted to Professor DeVogelaere for the assistance his work has given us. The notation* that is used here for the expansions of the vector-valued functions is principally due to him and without the help of such a concise, precise notation, it would not have been possible to arrive at our results. Also, the basic list procedures along with a good suitable list structure which he made available have been invaluable in constructing in the program RKMI.

Our results are thus built upon the work of others. We would characterize our contribution here as follows:

1) Scheme definitions are provided that include the classical RK schemes and finite difference schemes and, at the same time, are generalizations in that they provide us naturally with schemes outside this class. Ceschino-Kuntzmann almost did this, but for the class of schemes considered, we have a more general definition than they give.

2) An extension has been made to the work of Butcher which deals only with $Dx = X \circ x$ and classical RK schemes. We, thus, are able to define quantities analogous to his, but for generalized RK schemes. This leads to an overall global view of the scheme generation problem.

3) It is shown that the differentials and the various harmonics that we define after the fashion of Butcher are, in reality,

*See Appendix V.

all obtainable from one recursive definition of a generic operator Y and the generic z and y which lie in the domain and range of Y.

4) The basic algorithms of R. DeVogelaere have been used to build procedures that actually carry out the substitutions needed to generate generalized RKF schemes and, thus, extend his basic work.

5) The application of these results has been illustrated by applying them to some recently obtained generalized schemes presented by Butcher[4] and it is shown that if one is willing to remember a function value rather than perform an intermediate substitution, one can obtain the same orders of accuracy with one less substitution.

6) From a formal viewpoint, we have tried, and we hope to some degree succeeded, to show how one can develop a formalism for the expression of these schemes that leads naturally to their development and, at the same time, is expressed in such a manner that it can be reflected and preserved in procedures that will then free the investigator from having to reperform the derivation for each specific example. In short, we have attempted to generate schemes.

There is a rather logical ordering to the chapters that follow. Chapter II deals with the simplest schemes, the generalized RK schemes for $Dx = X \circ x$ and, thus, serves as an introduction to the remaining work. Chapter III develops a different, constructive approach to obtaining schemes and does so starting with the slightly more complicated case of generalized RKF schemes for $D^p x = X \circ x$. Although it is, in a sense,

a separate work and can stand alone, the ideas used in the proofs presented there are quite similar in character to Chapter II. Chapter IV generalizes the work of Chapter II to generalized RK schemes for $D^p x = X \circ x$ and relies, to some extent, on the work and ideas presented in Chapter III. Chapter V re-expresses our results using a new basis and the harmonics associated with this basis. Surprisingly, the results of Chapters III and IV simplify tremendously in this basis. Chapter VI explains how we represent the work of Chapter III in such a manner that we can build procedures to carry out the successive substitutions. Chapter VII presents illustrative examples of how these results can be obtained. Chapter VIII indicates some omissions and some possible directions that future work might take. The appendices are fairly well summarized by their titles. The appendices form an integral part of the work; however, they can be consulted only as needed and then only to the depth required.

In each chapter and each appendix of this work, the first few paragraphs contain a short introduction to the work. It seems rather pointless to repeat those remarks here; we simply indicate that an overall view should be obtainable by reading only these introductions. We shall, therefore, in the remainder of this Introduction make some rather general remarks about the work presented in the various chapters.

The original idea that this investigation of integration schemes containing data from the past be carried out was made by Professor R. DeVogelaere shortly before taking his sabbatical leave in Europe. At that time, very little progress was made. Upon his return, the project was again resumed and he very kindly made available the basic ALGOL list procedures that appear here. Following the ideas of his

work in this field, a large portion of Chapter III was built into a procedure. However, while it was obvious what needed to be done (that is, it was necessary to express the operations of substitution $X(\xi_j)$ and and multiplication $DX(\xi_j) \cdot S$ in the coefficient space of the derivatives of X), the author had no suitable means of obtaining these quantities other than actually laboriously carrying out these expansions and extracting the terms.

The author finally decided, almost out of desperation, that there must be some other way to obtain the needed quantities, and Butcher's work with differentials seems to hold the key to the desired results. However, in view of the work presented by Ceschino-Kuntzmann and that of Professor DeVogelaere with which the author was familiar, it was equally obvious that in order to arrive at usable results, it was necessary to follow their lead and split the solution $\xi = u + v$ where u is a finite number of terms of the Taylor expansion of ξ and v is the remaining infinite set of terms. This, together with the fact that we wished to treat $D^p x$ rather than Dx, along with our insistence in using points from the past, caused much difficulty in obtaining a definition for the differentials. However, with the aid of a very systematic, concise expression for the Taylor's expansion of a vector valued function and by restricting ourselves to $Dx = X \circ x$, an appropriate definition was arrived at and out of this grew the work of Chapter II. This, however, turned out to present a global view to the parameter defining equations. Once having seen how the pattern of generation was established and how the proofs were to be constructed, it became an easy task to develop the appropriate differentials and the corresponding harmonics for Chapter III.

We, thus, arrived at a position where instead of carrying out a laborious expansion for a few terms and extracting the coefficients, we could now define a basis, define the harmonics of interest (for all the basis elements), and then prove that these expansions were those that we wanted. Thus, the work of Chapter III has as its basis the later work of Chapter II although, in a sense, it is more fundamental and is more easily generalized than that of the other chapters.

Chapter IV arose because having once done the work of Chapter II and the work of Chapter III dealing with $D^p x = X \circ x$, it seemed possible to extend the results of Chapter II to p-th order differential equations. The two Chapters, II and IV, are redundant; one need only present Chapter IV. However, the work of Chapter II was already completely done and it serves as a good introduction to the quantities which we have defined, the definitions of which are not obvious as given in Chapter IV.

The development of the results of Chapter IV, along with the application of the results of Chapters II, III, and IV to known examples lead to the restatement of the results in terms of error differentials and error harmonics as presented in Chapter V. The simplicity of the results when expressed in this form was an unexpected bonus; we were simply seeking to equate our work to the examples of Butcher[4]. In fact, the results are so simple that we would never have found a basis had we started in this fashion.

The direct connection that exists between the work of Chapters III and IV, though rather obvious once noted, went unnoticed for a long time. During this time, all the work of the various harmonics of Chapter IV was developed and these quantities were tabulated and used on various

examples. We present these results even though they may seem somewhat redundant. They have a significant place in our work and the relations of these harmonics one to another can help serve as a check on the validity of the presented results.

Since one of the principal aims of our work has been to present a means by which schemes can be generated, we present in this work the ALGOL 60 procedure, RKMI, which is a direct reflection of the constructive approach that has been developed in Chapter III. This procedure, as built, should prove capable of obtaining the parameter defining equations of generalized Runge-Kutta schemes with memory including the first derivative DX. The decision to present this algorithm has lead to the inclusion of Chapter VI and Appendices II and IV, all of which are devoted to various apects of RKMI.

The original intent, with respect to examples, was to investigate schemes pertaining to second order differential equations for which the first derivative did not appear explicitly. There seems to be good promise of obtaining more efficient or more accurate schemes for this class of equations provided one uses some data from the past. The works of R. DeVogelaere[5] and R. E. Scraton[6] give some indication of this. However, this has not been done, principally because of the scope of the results obtained. These schemes are certainly still of high interest. However, we now have at our disposal the means of carrying out a systematic investigation of a large class of schemes, but we do not presently have the time for such a task.

Therefore, a different set of illustrative examples has been presented. We noticed that Butcher's work[4], with a slight modification, furnishes examples of schemes that are truely generalized RK schemes

with memory. In Chapter VII, we present one classical example (the fourth order RK process) that will serve as a familiar introduction to how our results are applied. We then present four examples derived from a class of schemes that are the very simplest generalized RK processes with memory. These examples can be contrasted with those of Butcher and serve to illustrate what can be done with our results. These schemes are, we believe, new and their characteristics and worth have yet to be evaluated.

II. AN ANALYTIC DERIVATION OF GENERALIZED RUNGE-KUTTA METHODS FOR SYSTEMS OF FIRST ORDER DIFFERENTIAL EQUATIONS

In Chapter II, we treat the case of a system of first order, ordinary differential equations written as $Dx = X \circ x$, $x(a) = b$, where $x \in R \to R^n$ and $X \quad R^n \to R^n$. Our principal aim in this chapter will be to define a scheme, called a generalized Runge-Kutta scheme, for the numerical solution of this problem and to derive the nonlinear equations that define the parameters appearing in that scheme. This scheme is such that it will contain the Runge-Kutta and finite difference methods as special cases and, more generally, any mixture of these methods.

In order to effect this derivation, we define a number of functions. These are the approximations $\zeta \in R \to R^n$, the weighted differentials $W \in R \to R^n$, and the elementary differentials $A \in R \to R^n$. Along with these functions, we also introduce the weighted polynomials Φ, the elementary polynomials Γ, the polynomial weights γ, and the derivative harmonics α.

Since the approximations ζ and the solution x are functions, we can carry out a Taylor's expansion using a common origin. Using Theorems 1 and 4, we show that the derivatives of ζ are expandable into a series $\sum_i \alpha_i W$ where α are derivative harmonics. Using Theorems 2 and 5, we show that the derivatives of the solution x can be written as $\sum_i \alpha_i A_i$ where α_i are a suitable subset of the derivative harmonics. The connection between these series is obtained from Theorem 3 which shows that $W = \Phi A$.. At this stage, it is possible to directly compare the series for ζ and x and thus obtain the parameter defining equations, since Φ is a function of the scheme parameters. It is, however, convenient to factor the polynomials $\Phi = \gamma\Gamma$, using Theorem 6, where the polynomial weights γ are

numbers and the elementary polynomials Γ consist entirely of scheme parameters.

In the course of our analysis, it is assumed that there is a lower bound ℓ to the order of accuracy of the approximations ξ_i obtained from a generalized Runge-Kutta scheme and this leads to further conditions (Condition A) that the parameters must satisfy.

At the end of Chapter II, we collect together our results and present the problem, the solution scheme, and the nonlinear equations that define the parameters appearing in the solution scheme. These nonlinear equations consist of two sets; the set associated with Condition A which arises from a lower bound on the accuracy of the approximations, and the set associated with Condition B which arises when we equate the components of the Taylor's series development of x and the Taylor's series development of the approximation ζ after the series have been re-expressed in a common basis consisting of the elementary differentials A.

The results derived here are an extension of the work of Butcher[1]. In his paper, Butcher treats the case of a system of first order differential equations $Dx = X \circ x$, $x \in R \to R^n$, $X \in R^n \to R^n$ and using the derivatives of X, defines functions which he calls elementary differentials which enable him to compare the Taylor's series of the true solution $x(t_i)$ and the approximate solution x_i. The coefficients of the approximate solution lead to polynomials in terms of the parameters of the method under consideration. In his work, he is able to define these polynomials and their numerical coefficients in such a way that they can be generated from the definitions without recourse to carrying out the expansions. In effect, he has defined a basis of a space in which both the solution and the approximation are contained and given a means

of generating their harmonies in this space. This he has done for Runge-Kutta methods. It is our intent to extend these results to methods that use information from previous integration steps. These methods shall be called generalized Runge-Kutta type integration methods. The derivation of the results will be a rather straightforward application of the Taylor's series expansion and it is, of course, assumed that the function X has a suitable number of derivatives. The notation used will be explained as it is introduced.

The problem to be solved is stated as follows: Let $X \in R^n \to R^n$ be sufficiently differentiable and let it be desired to find the solution $x \in R \to R^n$ of the system of first order ordinary differential equations

$$\begin{aligned} Dx &= X \circ x \\ x(a) &= b. \end{aligned} \tag{1}$$

It is, or course, well known that for such an initial value problem a solution does exist locally and can be extended throughout an interval. It is assumed here that the problem is well posed, that a solution does exist and that it is desired to find an approximation x_i to the solution $x(t_i)$. This approximation is to be a constructive one and shall exist as a series of points in R^n. To effect the construction of such an approximation, it is, however, convenient to stay in a function space in order that the usual tools of analysis may be applied.

With this in mind, the following set of functions $\zeta \in R \to R^n$ is defined.

Definition 1:

$\zeta \in R \to R^n$ is an approximation if, and only if

$\zeta = u + \eta$ where

$$u = \sum_{r=o}^{\ell} \frac{I^r}{r!} D^r x(0)$$

$$\eta = \sum_{r=\ell}^{\infty} \frac{I^{r+1}}{(r+1)!} (r+1)(D^r \delta^r)(0) \qquad (2)$$

$$\delta^r = \sum_j g_{ij} \eta + \sum_j f^r_{ij} X \circ \zeta .$$

Note that u, η, and δ^r are all functions that map $R \to R^n$, that I is the identity function, $I(t) = t$ for $t \in R$. Also, while g_{ij} and f^r_{ij} are, in general, undetermined parameters in R, no more is specified other than that f may (or may not) depend on the index r. Definition 1 is recursive in nature and does, indeed, define a set of functions, approximations, since for each choice of g_{ij} or f^r_{ij}, there is defined a different function. More will be said about these parameters shortly.

Now let $\phi_i = \theta_i \cdot I$ and denote for any function y

$$y_i = y \circ \phi_i .$$

There is then, associated to (2), the set of equations

$$\zeta_i = u_i + \eta_i$$

$$u_i = \sum_{r=o}^{\ell} \frac{I^r}{r!} D^r x_i(0)$$

$$\eta_i = \sum_{r=\ell}^{\infty} \frac{I^{r+1}}{(r+1)!} (r+1)(D^r \delta^r_i)(0) \qquad (3)$$

$$\delta^r_i = \sum_j g_{ij} \eta_j + \sum_j f^r_{ij} X \circ \zeta_j .$$

This essentially amounts to evaluating each of the approximations at a

separate point t_i and then requiring that there be a dependence in the form $t_i = \theta_i t$. Thus, (3) are also referred to as approximations. Actually, to be completely precise, more notation is needed for the functions defined by (2) and then new symbols should be used in (3). But this would only confuse the presentation and since it is not needed to keep track of the particular function of the set, this has not been done.

Writing out ζ_i explicitly gives

$$\zeta_i = u_i + \sum_{r=\ell}^{\infty} \frac{I^{r+1}}{(r+1)!} (r+1) \{\sum_j g_{ij} D^r \eta_j + \sum_j f_{ij}^r D^r(X \circ \zeta_j)\}(0). \quad (4)$$

Before proceeding further, the special case of $\zeta_i \equiv u_i$ should be considered. This implies that the quantity in the braces in (4) is identically zero. Since, however, there is no guarantee that

$$D^r \eta_i(0) = D^r(X \circ \zeta_i)(0) = 0$$

it must be true that $g_{ij} = f_{ij}^r = 0$ in this instance. Thus, if it is determined that $\zeta_i = u_i$, then g_{ij} and f_{ij}^r are to be set identically to zero.

If in (4) the following choice of parameters is made

$$g_{ij} \equiv 0,\ f_{ij}^r = 0 \text{ for } i \neq j \text{ and } f_{ii}^r = \theta_i/(r+1),$$

then

$$\zeta_i = u_i + \sum_{r=\ell}^{\infty} \frac{I^{r+1}}{(r+1)!} \theta_i D^r(X \circ \zeta_i)(0)$$

which can be written as

$$\zeta(\theta_i I) = u(\theta_i I) + \sum_{r=\ell}^{\infty} \frac{(\theta_i I)^{r+1}}{(r+1)!} D^r(X \circ \zeta)(0). \tag{5}$$

On the other hand, if $Dx = X \circ x$, then

$$x(\theta_i) = u(\theta_i) + \sum_{r=\ell}^{\infty} \frac{\theta_i^{r+1}}{(r+1)!} D^r(X \circ x)(0). \tag{6}$$

A comparison of (5) and (6) shows that

$$\zeta_i(1) = x(\theta_i) \tag{7}$$

in this particular case. That (5) and (6) do, indeed, produce (7) can be proved inductively by noting that ζ and x have the same derivatives at zero provided the order is less than $\ell + 1$. Their higher order derivatives can be expressed in terms of those of lower order. Thus, the derivatives are the same for all orders at 0 and, hence, they have the same Taylor's series at zero. We state these results as

Property 1: If ζ_i is an approximation defined by (3) and if x is a solution of $Dx = X \circ x$, then

$\zeta_i(1) = \zeta(\theta_i) = x(\theta_i)$ provided $g_{ij} \equiv 0$ and $f_{ij}^r = 0$, $i \neq j$, $f_{ii}^r = \theta_i/(r+1)$.

That is, with a suitable choice of parameters, the approximation reduces to the solution. Or to put it another way, the solution exists in the set of approximations; that is, the solution is an approximation.

If, on the other hand, the parameters are chosen with $g_{ij} \in R$ and $f_{ij}^r = a_{ij} \in R$ independent of r, then ζ_i can be written as

$$\zeta_i = u_i + \sum_{r=\ell}^{\infty} \frac{I^{r+1}}{r!} \{\sum_j g_{ij} D^r \eta_j + \sum_j a_{ij} D^r(X \circ \zeta_j)\}(0). \tag{8}$$

However,

$$X \circ \zeta_j - R_j = \sum_{r=\ell}^{\infty} \frac{I^r}{r!} D^r(X \circ \zeta_j)(0)$$

where
$$R_j = \sum_{r=0}^{\ell-1} \frac{I^r}{r!} D^r(X \circ \zeta_j)(0) \tag{9}$$

and
$$\eta_j = \sum_{r=\ell}^{\infty} \frac{I^r}{r!} D^r \eta_j(0).$$

This last relation is obtained from (3). Substituting these into (8) gives

$$\zeta_i = u_i + I\{\sum_j g_{ij}(\zeta_j - u_j) + \sum_j a_{ij}(X \circ \zeta_j - R_j)\}. \tag{10}$$

Up to now, nothing has been said about the type of schemes that are to be used in the solution of (1). As was previously mentioned, it is desired to find points in R^n that are constructive approximations to the solution $x(t_i)$ at the point t_i in the interval of interest. There are a large number of methods for step-by-step solution of ordinary differential equations and no attempt is made here to cover all of them. A convenient reference to many such methods and a guide to literature can be found in Ceschino-Kuntzmann[2]. We shall here consider a class of methods that is sufficiently large to contain Runge-Kutta methods, finite difference methods, and methods that can be considered as a mixture or generalization of such methods.

In the following work, ξ_i will be considered to be an element of the vector space R^n and since it is to be an approximation to $x(t_i)$ where t_i is in the interval of interest, it too will be referred to as an approximation. There should be no confusion with the previously defined approximations since the former are functions and the latter are points in a vector space. A generalized Runge-Kutta type scheme is now

defined as follows:

Definition 2: The approximation ξ_i is said to be obtained by means of a generalized Runge-Kutta type scheme if, and only if

$$\xi_i = \sum_j g_{ij}\, \xi_j + \sum_j a_{ij}\, X(\xi_j), \tag{11}$$

where $X \in R^n \to R^n$ is that of (1); g_{ij}, $a_{ij} \in R$, and ξ_j is an approximation obtained in the same fashion.

In short, any scheme which creates approximations using a linear combination of function values and other approximations is called a generalized R. K. scheme. Obviously, any Runge-Kutta method or finite difference method can be written in this fashion; equally obvious is the fact that there are methods that use higher order derivatives that are not included here. However, it is not our intent to try and cover all known methods and the class defined by (11) is sufficient large to be of interest.

A few words are in order concerning the interpretation of (11). As is usually the case, it is assumed that the solution is desired in some interval and that this interval has been subdivided into discrete points t_i. It is also implied that ξ_i will be an approximation to $x(t_i)$. At the moment, nothing further is said about the ordering of the points or the starting of the method. These details are discussed in Chapter III where schemes of a more general nature are defined. They are not needed for the present work and, thus, are omitted. The schemes defined here are contained in the class of schemes discussed in Chapter III and it is implicitly assumed that the details presented there concerning the actual ordering of the approximations are to also be used

here. The present aim is to show how the parameters g_{ij}, a_{ij} are to be obtained so that $\xi_i \approx x(t_i)$.

Equation (11) can be rewritten as

$$\xi_i = u(\theta_i) + \sum_j g_{ij} [\xi_j - u(\theta_j)] + \sum_j a_{ij} [X(\xi_j) - \bar{R}_j] + \sum_j g_{ij} u(\theta_j) + \sum_j a_{ij} \bar{R}_j - u(\theta_i). \tag{12}$$

In most schemes, it is possible to state that for all approximations ξ_i there is a lowest order of accuracy. That is, $\xi_i = x(t_i) + O(h^{\ell+1})$ where ℓ has some lower bound. This means that the value ξ_i must equal the Taylor's expansion $u(\theta_i) + \cdots$. Thus, it is reasonable to require that this approximation can be written as

$$\xi_i = u(\theta_i) + \sum_j g_{ij} [\xi_j - u(\theta_j)] + \sum_j a_{ij} [X(\xi_j) - \bar{R}_j] \tag{13}$$

with

$$\sum_j g_{ij} u(\theta_j) - \sum_j a_{ij} \bar{R}_j - u(\theta_i) = 0. \tag{13'}$$

Equation (12) is an identity. The requirement that (13') be satisfied forces $\xi_i = x(t_i) + O(h^{\ell+1})$ provided $X(\xi_j) - \bar{R}_j$ is of high enough order. This will be the case if $\bar{R}_j = R_j(1)$ where R_j is defined by (9). It is now easily seen that if g_{ij}, a_{ij} satisfy (13'), then from (10) and (13) comes the fact that $\zeta_i(1) = \xi_i$.

We state this as

Property 2: If $\zeta_i \in R \to R^n$ is an approximation defined by (3) and if ξ_i is an approximation obtained by means of a generalized R. K. scheme, then $\zeta_i(1) = \xi_i$ provided the parameters g_{ij}, $f^r_{ij} = a_{ij}$ common to both approximations satisfy (13') which are referred to as the conditions on the parameters.

Properties 1 and 2 show that, in the set of approximations ζ_i, there

exist approximations that are the solution x and there exist approximations that have the value of approximations obtained constructively. It is thus possible to work with these functions in determining the parameters of the scheme.

The problem of finding a correct scheme can be stated as follows: Let $x \in R \to R^n$ be the solution of the differential equation $Dx = X \circ x$. Let $\zeta_i(1)$ be the desired approximation. Choose an origin and write the Taylor's expansion of x and ζ_i as

$$\begin{aligned} x(\theta_i) &= u(\theta_i) + \sum_{r=\ell}^{\infty} \frac{\theta_i^{r+1}}{(r+1)!} D^r(X \circ x)(0) \\ \zeta_i(1) &= u_i(1) + \sum_{r=\ell}^{\infty} \frac{(1)^{r+1}}{(r+1)!} D^{r+1}\zeta_i(0) \quad . \end{aligned} \tag{14}$$

Choose the parameters that appear in ζ_i so that these two series match to a given order.

It is thus seen that the principal problem is the calculation of the derivatives of ζ_i and of $X \circ x$ in such a fashion that it is possible to compare the series. Therefore, a common basis must be found for these two series. Equation (5) shows that it is only necessary to calculate the derivatives of $X \circ \zeta_i$ since a proper choice of g_{ij} and f_{ij}^r will give the expansion of $x(\theta_i) = x_i(1)$. We now proceed to construct an expansion formula that will allow the derivation of the necessary equations.

The derivatives $D^r\zeta_i$ are to be calculated for $r \geq \ell + 1$. Equation (3) shows that this is equivalent to calculating $D^r \eta_i$ for those values of r. That is, $D^{r+1}\zeta_i(0) = D^{r+1} \eta_i(0) = (r + 1) D^r \delta_i^r(0)$. Thus, the derivatives of δ_i^r will first be calculated. In the following, the superscrip on δ_i and f_{ij} will be dropped; however, it is necessary to remember their dependence, particularly if the summation indices are changed.

By definition $\delta_i = \sum_j g_{ij}\,\eta_j + \sum_j f_{ij}\, X \circ \zeta_j$

which leads to*

$$\delta_i = \sum_j g_{ij}\,\eta_j + \sum_j f_{ij}\, X \circ (u_j + \eta_j) \tag{15}$$

$$\delta_i = \sum_j g_{ij}\,\eta_j + \sum_j f_{ij}\left[X \circ u_j + \sum_{s=1}^{\infty} \frac{1}{s!}(D_{N_1 \ldots N_s}\, X \circ u_j)\eta_{jN_1} \cdots \eta_{jN_s}\right].$$

Differentiate (15) to obtain

$$D^r\delta_i = \sum_j g_{ij}\, D^r\eta_j + \sum_j f_{ij}\left[D^r(X \circ u_j) + \sum_{s=1}^{\infty} \frac{1}{s!} \sum_{i_1=0}^{r} \cdots \sum_{i_s=0}^{i_{s-1}} \alpha_{ri_1\cdots i_s}\, D^{r-i_1}(D_{N_1\ldots N_s}\, X \circ u_j)\, D^{i_1-i_2}\eta_{jN_1} \cdots D^{i_s}\eta_{jN_s}\right] \tag{16}$$

where $\alpha_{ri_1\cdots i_s} = \binom{r}{i_1}\binom{i_1}{i_2}\cdots\binom{i_{r-1}}{i_s}$.

However, $D^{r+1}\eta_i(0) = (r+1)\, D^r\delta_i(0)$ so that (16) can be written as

$$D^{r+1}\eta_i(0) = \sum_j (r+1)\, f_{ij}^r\, D^r(X \circ u_j)(0) + \sum_{s=1}^{\infty}\left[\frac{1}{s!} \sum_{i_1} \cdots \sum_{i_s} \alpha_{ri_1\cdots i_s} \sum_j (r+1)\, f_{ij}^r\, D^{r-i_1}(D_{N_1\ldots N_s}\, X \circ u_j)(0) \cdot D^{i_1-i_2}\eta_{jN_1}(0) \cdots D^{i_s}\eta_{jN_s}(0)\right] + \sum_j (r+1)\, g_{ij}\, D^r\eta_j(0). \tag{16'}$$

Before proceeding further, it is convenient to note that the range of the summation indices can be materially reduced. From (3) $D^r\eta_i(0) = 0$ for $r < \ell + 1$. Thus, one can write that

$$\begin{aligned} i_1 - i_2 &\geq \ell + 1 \\ i_2 - i_3 &\geq \ell + 1 \\ &\vdots \\ i_s &\geq \ell + 1 \end{aligned}$$

which enables the following limits to be set

*See Appendix V

$$0 \leqq s \leqq r \div (\ell + 1)$$

$$i_1 \geqq s(\ell + 1),\ i_2 \geqq (s - 1)(\ell + 1),\ \ldots,\ i_s \geqq (\ell + 1) \qquad (17)$$

$$i_1 \leqq r,\ i_2 \leqq i_1 - (\ell + 1),\ i_3 \leqq i_2 - (\ell + 1),\ \ldots,\ i_s \leqq i_{s-1} - (\ell + 1)\ .$$

The limits defined in (17) will be considered as the normal range of summation for (16') when evaluating $D^r\eta_i(0)$. We now proceed to define functions $W \in R \to R^n$, $A \in R \to R^n$ and polynomials Φ in a fashion similar to Butcher[1]. While the work of Butcher was inspirational in suggesting this approach, it is the expansion represented by (16') that has suggested how to actually define these quantities and (16') will be used repeatedly to derive the required results.

We now proceed to define functions $W \in R \to R^n$ of given rank R, order r, degree s where $R \geqq r \geqq \ell + 1$ are integers and the integer s lies in the range $0 \leqq s \leqq (r - 1) \div (\ell + 1)$. The division symbol $\div$ is used to indicate an integer divide. The functions are called weighted differentials and it is important to note that for a given rank R, order r, degree s there may correspond many of these functions.

Definition 3: The weighted differentials of rank R, order r, degree s = 0 with R = r are defined to be

$$W_i = \sum_j R\, f_{ij}^{R-1}\ D^{r_o}\ (X \circ u_j),\ r = r_o + 1. \qquad (18)$$

W_i is a weighted differential of rank R, order r, degree s if

$$W_i = \sum_j R\, g_{ij}\, W_j \qquad (19)$$

where W_j is a weighted differential of rank R - 1, order r, degree s, or if

$$W_i = \sum_j R\, f_{ij}^{R-1}\; D^{r_o}(D_{N_1 \cdots N_s} X \circ u_j)\; W_{jN_1}^{(1)} \cdots W_{jN_s}^{(s)} \tag{20}$$

where

$$r = r_o + r_1 + \ldots + r_s + 1$$

$$R = r_o + R_1 + \ldots + R_s + 1$$

and r_i = order of $W_{jN_i}^{(i)}$, R_i = rank of $W_{jN_i}^{(i)}$.

The degree s is bounded by $0 \leq s \leq (r - 1) \div (\ell + 1)$, f and g are elements of the real line R^1.

Since the parameters f_{ij} and g_{ij} are in no way restricted in the definition, there will correspond for each choice of these parameters and their associated summation ranges a set of weighted differentials. However, in practice, the set that will be of interest will be those weighted differentials for which the parameters f and g are those of the generalized R. K. scheme under consideration.

The actual generation of the set of weighted differentials can be carried out in many ways. We shall indicate one such pattern. Before doing this, it is helpful to introduce some notation that will help shorten the task of writing down the differentials. Define

$$E = \sum_j R\, g_{ij}$$

$$C_j^o = \sum_{j_1} R\, f_{ij_1}^{R-1}\; D^{j+\ell-1}(X \circ u_{j_1}) \tag{21}$$

$$C_j^s = \sum_{j_1} R\, f_{ij_1}^{R-1}\; D^{j-1}(D_{N_1 \cdots N_s} X \circ u_{j_1}) \quad .$$

Using these operators (18), (19), (20) can be written as

$$W_i = C_j^o \quad , \quad r = j + \ell \tag{18'}$$

$$W_i = E\, W_j \tag{19'}$$

$$W_i = C_j^s W_{j_1 N_1}^{(1)} \cdots W_{j_s N_s}^{(s)} , \; r_o = j - 1 \qquad (20')$$

It is possible to utilize this notation quite effectively in the generation of the weighted differentials and their related quantities. However, it is necessary to be careful since the associative law does not necessarily apply to these quantities. Their use is described much more fully in Appendix I. At the present time, we only indicate the pattern that is used in generating the differentials.

The first weighted differential is taken to be that of lowest order and rank, C_1^o. There is now a choice of how to proceed. The order can be kept fixed and the function of higher rank can be generated or the rank can be set equal to the order and functions of higher order, with the rank equal to the order, can be generated, or the process can be carried out as a mixture of the above two patterns. In any case, the differentials are considered as arranged in a matrix fashion ordered with respect to rank R and order r. This can be indicated as Γ_{rR} where the entry Γ_{rR} has order r, rank R. We note that E raises the rank while leaving the order along with the degree unchanged. The operator C_j^o creates a differential of degree 0 of given rank and order, these latter being equal. While the operator C_j^s raises the rank, order, and degree. The ordering of the sets of functions with like rank, order, and degree is not unique. In fact, to do this in general requires some careful thought to be sure all functions are obtained. The order that is established by construction in the tables in Appendix I is to hold for all our work. However, it will be evident that we have not given a unique means for continuing this ordering to larger sets of functions than have been used here. If these quantities are generated by an algorithm then such a pattern must be established. For the present derivation

the actual ordering of the function is of no consequence; it is simply assumed that there is some sequential positioning that is well established.

The reader should refer to Table I of Appendix I and the description of that table given there for more details on how to generate these quantities. However, for convenience, a short table is presented here to show how the functions can be arranged:

TABLE I

A	B	C	ψ	R-ℓ def.	R-ℓ		
				1	2		
1	1	0	0	C_1^0	EC_1^0		
				2	3		
2	2	0	1	C_2^0	EC_2^0		
2	2	1	2	$C_1^1\psi_0$	$EC_1^1C_1^0$	$C_1^1EC_1^0$	
				3	4		
3	3	0	3	C_3^0	EC_3^0		
3	3	1	4	$C_2^1\psi_0$	$EC_2^1C_1^0$	$C_2^1EC_1^0$	
3	3	1	5	$C_1^1\psi_1$	$EC_1^1C_2^0$	$C_1^1EC_2^0$	
3	3	1	6	$C_1^1\psi_2$	$E(C_1^1)^2C_1^0$	$C_1^1EC_1^1C_1^0$	$(C_1^1)^2EC_1^0$

where $A = R - \ell$

$B = r - \ell$

$C = s$

ψ = sequential count of functions with $R = r$

def. = definition of the function.

To illustrate the proper interpretation of these entries, consider one such entry

$$C_1^1EC_1^1C_1^0 = \sum_{j_1} R_4 \, f_{ij_1}^{R_4-1} \, D^0(D_{N_1} X \circ u_{j_1}) \sum_{j_2} R_3 \, g_{j_1j_2} \sum_{j_3} R_2 \, f_{j_2j_3}^{R_2-1}$$
$$D^0(D_{N_2} X \circ u_{j_3})_{N_1} \sum_{j_4} R_1 \, f_{j_3j_4}^{R_1-1} \, D^\ell(X \circ u_{j_4})_{N_2} \qquad (22)$$

where $R_i = \ell + i$. This points out that the notation in actuality does not

carry enough information since it is necessary to know the rank of the factors, here R_i. However, it is sufficient for the purpose of establishing a pattern of generation as we have used it. Obviously, this information must be carried along if the functions are generated by an algorithm. It is omitted here for the sake of clarity in presenting the tables.

The abbreviated notation of Butcher[1] will be used for the differentials. Thus, (18) and (20) will be written as

$$W_i = \langle r_o \rangle \tag{18"}$$

$$W_i = \langle r_o; W_1 \ldots W_s \rangle \tag{19"}$$

where

$$R = 1 + r_o + R_1 + \ldots + R_s$$

$$r = 1 + r_o + r_1 + \ldots + r_s.$$

If there are repeated factors in the W_i, then they will be collected together as

$$W_i = \langle r_o; (W_1)^{u_1} \ldots (W_\sigma)^{u_\sigma} \rangle \tag{23}$$

with

$$R = 1 + r_o + \sum_{i=1}^{\sigma} R_i u_i$$

$$r = 1 + r_o + \sum_{i=1}^{\sigma} r_i u_i$$

$$s = u_1 + \ldots + u_\sigma \quad .$$

While the weighted differentials have been defined as elements of $R \to R^n$ and, thus, can be differentiated and can have their properties investigated; this will not be done. Instead, we proceed directly to obtain the desired results in terms of the functions η_i which are the functions whose derivatives we wish to calculate. In all cases, it is assumed that the derivatives are to be calculated for $r \geq \ell + 1$ since the results are known below that order to be identically zero.

Theorem 1

$$D^{R+1}\eta_i(0) = \sum_{j\in S_{R+1}} \alpha_j W_i^{(j)}(0), \quad S_{R+1} = \{j \mid W^{(j)} \text{ has rank } R+1\}.$$

That is, the derivatives of order R + 1 of the function η_i evaluated at 0 can be expanded into the set of weighted differentials of rank R + 1.

Proof: If $R = \ell + 1$, then from (16') is obtained

$D^{(\ell+1)}\eta_i(0) = \sum_j (\ell+1) f_{ij}^{\ell} D^{\ell}(X \circ u_j)(0)$. Thus, the theorem is certainly true for $R = \ell + 1$, the coefficient α being $\alpha_j = (\ell+1) f_{ij}^{\ell}$. An examination of (16') will show that the functions $D^{R+1}\eta_i(0)$ of rank R + 1 are obtained from those of lower order. The results of the theorem follow by induction on R. If there is any doubt about obtaining all the differentials defined in Definition 3, then a careful examination of the index range should convince one that no differentials are left out and that no new ones are obtained through the use of (16').

Next will be defined a set of functions $A \in R \to R^n$. These functions will be called differentials of order r. These are, in fact, a subset of the weighted differentials and can be obtained by setting $g_{ij} \equiv 0$, $f_{ij}^{R-1} = \frac{1}{R}$, and $\theta_i = 1$. It is, however, convenient to have a separate definition and notation for these functions since they will serve as a basis for the expansions which are to be compared when determining the parameters of a particular generalized R. K. scheme. These functions are defined as follows:

Definition 4:

The only elementary differential of order r, degree s = 0 is

$$A = D^{r_o}(X \circ u), \quad r = r_o + 1. \tag{24}$$

A is an elementary differential of order r degree s if, and only if

$$A = D^{r_o}(D_{N_1 \cdots N_s} X \circ u)\, A_{N_1} \cdots A_{N_s} \tag{25}$$

where $r = 1 + r_o + r_1 + \cdots + r_s$

with r_n = order of A_{N_n}. The degree s is bounded by $0 \leq s \leq (r - 1) \div (\ell + 1)$.

Again, it is assumed that $r \geq \ell + 1$. It should be noted that if $\mathcal{H}_r = \{W_i\}$ where W_i are of order r, then there exists a one-to-one correspondence between the sets $\mathcal{H}_r$ and the functions A_r of order r. In fact, if $g_{ij} \equiv 0$, then there exists a one-to-one correspondence between W_i and A since for this particular choice of g_{ij} all the W_i have equal rank and order.

With regard to the A, we have the following theorem:

Theorem 2:

Let x be a solution of $Dx = X \circ x$. Then

$$D^{r+1}x(0) = \sum_{j \in S_{r+1}} \alpha_j A^{(j)}(0), \quad S_{r+1} = \{j \mid A^{(j)} \text{ has order } r + 1\}.$$

That is the derivative of order r + 1 is expandable into the set of differentials of order r + 1 and all α_j are greater than zero.

Proof: In (16'), choose $g_{ij} \equiv 0$, $f_{ij}^r = \frac{\theta_i}{r+1}$. This yields

$$D^{r+1}\eta_i = \theta_i^{r+1} D^{r+1}\eta = \theta_i^{r+1} D^{r+1}(X \circ u) + \sum_s \frac{1}{s!} \sum_{i_1 \cdots i_s} \alpha_{r i_1 \cdots i_s} \theta_i^{r+1} D^{i-i_1}(D_{N_1 \cdots N_s} X \circ u) \cdot D^{i_1 - i_s}\eta_{N_1} \cdots D^{i_s}\eta_{N_s}. \tag{26}$$

Cancel θ_i^{r+1}, evaluate at 0, and do an inductive proof on the order r to obtain the desired results.

It is convenient to have a short hand notation for the differentials A and we shall for these functions interpret the C_j^i operater as

$$C_j^o = D^{j+\ell-1}(X \circ u)$$
$$C_j^s = D^{j-1}(D_{N_1 \cdots N_s} X \circ u). \tag{27}$$

Using this notation, (24) and (25) can be written as

$$C_j^o \quad , \; r_o = j + \ell - 1 \tag{24'}$$

$$C_j^s \, A_{N_1} \cdots A_{N_s} \tag{25'}$$

It is also useful to have a brace notation

$$A = \{r_o; \, A_1 \cdots A_s\} = \{r_o; \, (A_1)^{u_1} \cdots (A_\sigma)^{u_\sigma}\}, \tag{28}$$

where $s = \sum_{i=1}^{\sigma} u_i$

$$r = 1 + r_o + \sum_{i=1}^{\sigma} u_i \, r_i \, .$$

Next to be defined are the weighted polynomials Φ of rank R, order r, degree s. We shall establish a one-to-one correspondence between the Φ and W according to rank, order, degree, and sequential position within the set of a given rank, order, and degree. That is, the pattern of generation for Φ and W is to be identical.

Definition 5:

The weighted polynomials Φ of rank R, order r, degree s = 0 with R = r are defined to be:

$$\Phi_i = \sum_j R \, f_{ij}^{R-1} \, \theta_j^{r-1} \tag{29}$$

Φ_i is a weighted polynomial of rank R, order r, degree s if

$$\Phi_i = \sum_j R \, g_{ij} \, \Phi_j \tag{30}$$

where Φ_j is a weighted polynomial of rank R - 1, order r, degree s, or if

$$\Phi_i = \sum_j R \, f_{ij}^{R-1} \, \theta_j^{r_o} \, \Phi_{1j} \cdots \Phi_{sj} \tag{31}$$

where $r = 1 + r_o + r_1 + \cdots + r_s$

$R = 1 + r_o + R_1 + \cdots + R_s$

with r_i = order Φ_{ij}

R_i = rank Φ_{ij}.

The degree s is bounded by $0 \leqq s \leqq (r-1) + (\ell+1)$, f and g are elements of the real line.

The reader will note that Definition 3 and Definition 5 are identical in form. We have simply factored out the coefficients of the derivatives. With regard to shorthand notation, the following will be used:

$$\begin{aligned} C_j^o &= \sum_{j_1} R\, f_{ij_1}^{R-1}\, \theta_{j_1}^{j+\ell-1} \\ C_j^s &= \sum_{j_1} R\, f_{ij_1}^{R-1}\, \theta_{j_1}^{j-1} \\ E &= \sum_{j_1} R\, g_{ij_1} \,. \end{aligned} \tag{32}$$

The bracket notation used is

$$\Phi = [r_o;\, \Phi_1 \ldots \Phi_s] = [r_o;\, (\Phi_1)^{u_1} \ldots (\Phi_s)^{u_s}] \tag{33}$$

where $s = \sum_{i=1}^{\sigma} u_i$.

Using the weighted differential W, the weighted polynomial Φ, and the elementary differential A, it is possible to write the Taylor's series for x_i and that of ζ_i in terms of a common basis. In order to accomplish this, the following results are needed which are stated below as Theorem 3:

Let $A = \{r_o;\, A_1 \ldots A_s\}$ be of order $r = 1 + r_o + r_1 + \ldots + r_s$ where r_i = order of A_i. Let $W \in \mathcal{W}_r = \{W \mid$ of order r, rank $R \geq r\}$. Let $\Phi \in \Phi_r = \{\Phi \mid$ of order r, rank $R \geq r\}$. Then

$$W = \Phi A \tag{34}$$

where Φ and W are in 1 - 1 correspondence according to order, rank, degree, and sequential position in the set of given order and rank. That is

$$\begin{aligned} \langle r_o;\, W_1 \ldots W_s \rangle &= [r_o;\, \Phi_1 \ldots \Phi_s] \cdot \{r_o;\, A_1 \ldots A_s\} \\ E\ \langle r_o;\, W_1 \ldots W_s \rangle &= E[r_o;\, \Phi_1 \ldots \Phi_s] \cdot \{r_o;\, A_1 \ldots A_s\} \end{aligned} \tag{34'}$$

where $\Phi_i \rightarrow W_i \rightarrow A_i$.

Proof: These results follow directly from the definition of the various quantities. Equations (18), (24), and (29) give

$$W_i = \sum_j R\, f_{ij}^{R-1}\, D^{r_o}(X \circ u_j) = \sum_j R\, f_{ij}^{R-1}\, \theta_j^{r_o}\, D^{r_o}(X \circ u) = \Phi_i A.$$

Thus for R = r, s = 0, the results are true. If this factorization holds for rank R - 1, order r, degree s, then it is true also for rank R, order r, degree s since from (19) and (30) are obtained $W_i = \sum_1 R\, g_{ij}\, \Phi_j\, A = \Phi_i A$. A similar treatment of (20) and (31) will complete the proof.

Having established this correspondence between W and A, it is now possible to write the derivatives of $\eta_i(0)$ in terms of A rather than W. Let $W_i^{(j)}[R, r]$ be the jth function W_i of rank R, order r. Then from Theorem 1

$$D^R \eta_i(0) = \sum_{r=\ell+1}^{R} \sum_j \alpha_{Rr}^j\, W_i^{(j)}[R, r]\,(0) \tag{35}$$

where $j \in S_R = \{j \mid W_i^{(j)}$ has rank R, order r$\}$. This can be rewritten as

$$D^R \eta_i(0) = \sum_{r=\ell+1}^{R} \sum_{a \in S_r} \sum_{j \in S_R} \alpha_{Rr}^j\, \Phi_i^{(j)}[R, r]\, A_a(0) \tag{36}$$

where $S_r = \{a \mid A_a$ has order r$\}$. What is needed is an explicit formula for the coefficient α_{Rr}^j. There can be obtained rather laboriously from (16'). However, instead of calculating these coefficients one by one from (16'), it is possible to obtain a recursive formula for their representation. We shall do this shortly. In what follows, it will always be assumed that the functions are evaluated at the origin. However, in writing down the results this will not be indicated. The expansion (35) is carried out in two steps. First, coefficients called derivative harmonics are defined and then it is proved that these are the coefficients of (35). These coefficients are defined in

Definition 6:

The coefficient α is a derivative harmonic of rank R, order r, degree s = 0 with r = R if, and only if

$$\alpha = 1 \tag{37}$$

α is a derivative harmonic of rank R, order r, degree s if

$$\alpha = 1 \cdot \alpha_1 \tag{38}$$

where α_1 is a derivative harmonic of rank R - 1, order r, degree s, or if

$$\alpha = \frac{(R-1)!}{r_o!} \quad \frac{1}{(u_1)!\ldots(u_\sigma)!} \left[\frac{\alpha_1}{(R_1)!}\right]^{u_1} \cdots \left[\frac{\alpha_\sigma}{(R_\sigma)!}\right]^{u_\sigma} \tag{39}$$

where α_j is a derivative harmonic of rank R_j, order r_j, and

$$R = 1 + \sum_{i=1}^{\sigma} u_i R_i$$

$$r = 1 + \sum_{i=1}^{\sigma} u_i r_i$$

$$s = \sum_{i=1}^{\sigma} u_i \qquad , \qquad 0 \leq r \leq (r - 1) + (\ell + 1).$$

It is assumed that the pattern of generation of the α and the W is identical and that this pattern establishes a 1 - 1 correspondence between α and W according to rank, order, degree, and sequential position within a set of given rank and order.

It is now possible to state

Theorem 4:

$D^R \eta_i = \sum_{r=\ell+1}^{R} \sum_{j \in S_R} \alpha^j_{Rr} W_i^{(j)}[R, r]$ where α^j_{Rr} is the derivative harmonic corresponding to $W_i^{(j)}$. That is, the coefficients of Theorem 1 are the derivative harmonics of Definition 6.

Proof: Substitute the expansion (35) into both sides of (16') and collect terms of the function W_i. At the moment α^j_{Rr} are not assumed to be derivative harmonics; simply coefficients as yet to be determined.

Carrying out this substitution yields

$$D^R\eta_i = \langle R - 1\rangle + \sum_s \frac{1}{s!} \sum_{i_1 \ldots i_s} \alpha_{Ri_1\ldots i_s} \langle R - 1 - i_1; \sum_{r_1=\ell+1}^{i_1-i_2} \ldots \sum_{r_s=\ell+1}^{i_s}$$

$$\cdot \sum_{j_1 \in S_1} \ldots \sum_{j_s \in S_s} W^{(j_1)}[i_1 - i_2, r_1] \ldots W^{(j_s)}[i_s, r_s]\, \alpha^{j_1}_{i_1-i_2 r_1} \ldots \alpha^{j_s}_{i_s r_s} \rangle$$

$$+ E\Big(\sum_{r_1=\ell+1}^{R-1} \sum_{j_1 \in S_1} \alpha^{j_1}_{R-1,r_1} W^{(j_1)}[R - 1, r_1]\Big).$$

The general term on the left is either

$$W = \langle r_o; (W^{(q_1)}[R_1, \bar{r}_1])^{u_1} \ldots (W^{(q_\sigma)}[R_\sigma, \bar{r}_\sigma])^{u_\sigma} \rangle$$

or

$$W = E(W^{(q)}[R - 1, r]).$$

Noting that E and $\langle ; \rangle$ are linear operators, we obtain from the first term of $D^R\eta_i$ the coefficient of terms of degree zero of equal rank and order. They are always 1 and, thus, are the derivative harmonics of degree zero of equal rank and order. From the last term are obtained the coefficients corresponding to E(W) and these are seen to be unchanged. Hence, the derivative harmonics specified by (38) are the correct coefficients. It is now necessary to sort out the coefficients corresponding to $W = \langle r_o; \ldots \rangle$.

The general term on the right hand side is selected by placing

$$\left.\begin{aligned} j_i &= q_i \\ r_i &= \bar{r}_i \end{aligned}\right\} \qquad i = 1 \ldots s$$

$$\begin{aligned} i_1 &= R_1 + R_2 + \ldots + R_s \\ i_2 &= \quad R_2 + \ldots + R_s \\ &\vdots \\ i_s &= \quad\quad\quad R_s \end{aligned} \tag{40}$$

Thus, to each distinct permutation of the triplet $(q_i, R_i, \bar{r}_i)$ there corresponds a valid index set and conversely. The coefficient for any such choice is

$$\frac{(R-1)!}{(r_o)!}\left[\frac{\alpha^{j_1}_{R_1 r_1}}{R_1!}\right]^{u_1}\cdots\left[\frac{\alpha^{j_\sigma}_{R_o r_\sigma}}{R_r!}\right]^{u_\sigma}\quad\frac{1}{s!}$$

when

$$\alpha_{Ri_1\ldots i_s} = \binom{R-1}{i_1}\binom{i_1}{i_2}\cdots\binom{i_{s-1}}{i_s} = \frac{(R-1)!}{(R-1-i_1)}\,\frac{1}{(i_1-i_2)!\ldots(i_s)!}$$

has been used. There are $\dfrac{s!}{(u_1)!\ldots(u_\sigma)!}$ such distinct permutations and they all have the same coefficient. Thus, the coefficient of the general term is

$$\frac{(R-1)!}{r_o!}\,\frac{1}{(u_1)!\ldots(u_\sigma)!}\left[\frac{\alpha^{j_1}_{R_1 r_1}}{R_1!}\right]^{u_1}\cdots\left[\frac{\alpha^{j_\sigma}_{R_\sigma r_\sigma}}{R_\sigma!}\right]^{u_\sigma} \tag{41}$$

To complete the identification with the derivative harmonics of (39) it is only necessary to affix the rank of $W^{(j_1)}$ to $\alpha^{j_i}_{R_i r_i}$ and it is seen that by induction all of our α's are derivative harmonics.

The same theorem is true for the derivatives of x. Thus,

<u>Theorem 5</u>:

$$\mathbb{D}^r x(0) = \sum_{a\in S_r} \alpha^a_{rr}\, A^{(a)}(0)\ , \qquad S_r = \{a \mid A \text{ has order } r\} \tag{42}$$

where if

$$A = \{r_o;\ (A^{(a_1)})^{u_1}\ldots(A^{(a_\sigma)})^{u_\sigma}\}$$

$$r = 1 + r_o + \sum_{i=1}^{\sigma} u_i r_i\ , \qquad r \geq \ell + 1$$

$$s = \sum_{i=1}^{\sigma} u_i \geq 1$$

then

$$\alpha^{a}_{rr} = \frac{(r-1)!}{r_o!}\,\frac{1}{(u_1)!\ldots(u_\sigma)!}\left[\frac{\alpha^{a_1}_{r_1 r_1}}{r_1!}\right]^{u_1}\cdots\left[\frac{\alpha^{a_1}_{r_\sigma r_\sigma}}{r_\sigma!}\right]^{u_\sigma} \tag{43}$$

and for $s = 0$, $\alpha^{a}_{rr} = 1$. That is, the coefficients of Theorem 2 are derivative harmonics of equal rank and order. This implicitly assumes that the pattern of generation of W and A is identical.

Proof: Note that x is derivable from ζ and note that the derivative harmonics are independent of i and of f_{ij} or g_{ij}. The results follow immediately from Theorem 4; or else, one could proceed as with the previous theorem starting from (26).

With regard to shorthand notation for the derivative harmonics, it is convenient to define

$$\begin{aligned} C^{0}_{j} &= 1\cdot \\ C^{s}_{j} &= \frac{(R-1)!}{(j-1)!}\,\frac{1}{(u_1)!\ldots(u_\sigma)!}\left[\frac{1}{(R_1)!}\right]^{u_1}\cdots\left[\frac{1}{(R_\sigma)!}\right]^{u_\sigma}\cdot \\ E &= 1\cdot \end{aligned} \tag{44}$$

which will prove useful in defining the pattern of construction of these quantities. By now it should be evident that one need only give the pattern of constructing the generic quantity ψ by means of the operators C and E. Then, upon properly interpreting these operators, the various quantities previously defined will be obtained.

With the use of the above results it is possible to write the expansions for ζ and x in a form that allows a direct comparison of the terms. For ζ_i, we have

$$\zeta_i = u_i + \sum_{R=\ell+1}^{\infty}\frac{I^R}{R!}\sum_{r=\ell+1}^{R}\sum_{j\in S_R}\sum_{a\in S_r}\alpha^{j}_{Rr}\,\Phi^{(j)}_{i}[R,\,r]\,A^{(a)} \tag{45}$$

where $S_R = \{j \mid \Phi^{(j)}_{i}$ has rank R, order r$\}$

$S_r = \{a \mid A^{(a)}$ has order r$\}$

and it is implicitly assumed that the correspondence $\Phi_i \to W_i \to A$ mentioned earlier has been maintained. In short, for each rank R and order r, we sum over all the differentials W_i of that rank and order. However, each differential can be factored into a weighted polynomial of like rank and order times an elementary differential of like order. This factorization is carried out. We now proceed to factor out the elementary differentials and reverse the order of summation to obtain

$$\zeta_i = u_i + \sum_{r=\ell+1}^{\infty} \sum_{a \in S_r} \left\{ \sum_{R=r}^{\infty} \frac{I^R}{R!} \sum_{j \in S_R} \alpha_{Rr}^{j} \Phi_i^{(j)}[R, r] \right\} A^{(a)}. \tag{46}$$

For the solution x, we have

$$\begin{aligned} x &= u + \sum_{r=\ell+1}^{\infty} \frac{I^r}{r!} \sum_{a \in S_r} \alpha_{rr}^{a} \; A^{(a)} \\ x &= u + \sum_{r=\ell+1}^{\infty} \sum_{a \in S_r} \left\{ \frac{I^r}{r!} \; \alpha_{rr}^{a} \right\} A^{(a)}. \end{aligned} \tag{47}$$

Using (46) and (47), it is possible to write the nonlinear parameter defining equations of a generalized R.K. method as

$$\sum_{R=r}^{\infty} \frac{1}{R!} \sum_{j \in S_R} \alpha_{Rr}^{j} \Phi_i^{(j)}[R, r] = \frac{\theta_i^r}{r!} \; \alpha_{rr}^{a} \tag{48}$$

for $r = \ell + 1, \ell + 2 \ldots \infty$

$a \in S_r = \{a | A \text{ has order } r\}$

$S_R = \{j | \Phi_i^{(j)}[R, r] \to A^{(a)} \text{ of order } r\}.$

The error terms for the local truncation error are, of course, given by the difference of the left and right hand side of (48).

Equation (48) allows the parameter defining equations corresponding to any scheme under investigation to be written down. However, it is possible to factor further these equations. The weighted polynomials Φ_i have a coefficient associated with them that can be factored out and in

practice it is easier to obtain and use these equations if we instead use elementary polynomials $\Gamma_i^{(j)}[R, r]$ containing no numerical factors and numerical coefficients $\gamma_{Rr}^{(j)}$. The definitions of these quantities follows the same pattern as was previously used and they are given below as

Definition 7: The elementary polynomials Γ of rank R, order r, degree s = 0 with R = r are defined to be

$$\Gamma_i = \sum_j f_{ij}^{R-1} \theta_j^{r-1} \tag{49}$$

Γ_i is an elementary polynomial of rank R, order r, degree s if

$$\Gamma_i = \sum_j g_{ij} \Gamma_j \tag{50}$$

where Γ_j is an elementary polynomial of rank R - 1, order r, degree s, or if

$$\Gamma_i = \sum_j f_{ij}^{R-1} \theta_j^{r_o} \Gamma_{1j} \ldots \Gamma_{sj} \tag{51}$$

where $r = 1 + r_o + r_1 + \ldots + r_s$

$R = 1 + r_o + R_1 + \ldots + R_s$

and r_i = order of Γ_{ij}

R_i = rank of Γ_{ij}.

Definition 8: The coefficient γ is a polynomial weight of rank R, order r, degree s = 0 with R = r if, and only if

$$\gamma = R. \tag{52}$$

γ is a polynomial weight of rank R, order r, degree s if

$$\gamma = R \cdot \gamma_1 \tag{53}$$

where γ_1 is a polynomial weight of rank R - 1, order r, degree s, or if

$$\gamma = R \cdot \gamma_1 \cdots \gamma_s \tag{54}$$

where $r = 1 + r_o + r_1 + \ldots + r_s$

$R = 1 + r_o + R_1 + \ldots + R_s$

and r_i = order of γ_i

R_i = rank of γ_i.

With respect to the C and E operators, we have the following for Γ and γ, respectively

$$\begin{aligned} C_j^o &= \sum_{j_1} f_{ij_1}^{R-1}\, \theta_{j_1}^{j+\ell-1} \\ C_j^s &= \sum_{j_1} f_{ij_1}^{R-1}\, \theta_{j_1}^{j-1} \\ E &= \sum_j g_{ij} \end{aligned} \tag{55}$$

$$\begin{aligned} C_j^o &= R \cdot \\ C_j^s &= R \cdot \\ E &= R \cdot \quad . \end{aligned} \tag{56}$$

The factorization of Φ is carried out by means of

Theorem 6:

The weighted polynomial $\Phi_i^{(j)}[R, r]$ of rank R, order r, degree s can be written as

$$\Phi_i^{(j)}[R, r] = \gamma_{Rr}^j\, \Gamma_i^{(j)}[R, r] \tag{57}$$

where $\Gamma_i^{(j)}[R, r]$ is the elementary polynomial of rank R, order r, degree s, sequential position j, and γ_{Rr}^j is the polynomial weight of rank R, order r, degree s, sequential position j. It is implicitly assumed that

Φ, Γ, and γ are all generated from the same pattern.

Proof: The proof is by induction using the definition of Φ, Γ, and γ.

This theorem allows (48) to be rewritten as follows

$$\sum_{R=r}^{\infty} \sum_{j \in S_R} \alpha^j_{Rr} \frac{\gamma^j_{Rr}}{R!} \Gamma^{(j)}_i[R, r] = \frac{\theta^j_i}{r!} \alpha^a_{rr} \tag{58}$$

for $r = \ell + 1, \ell + 2, \ldots, \infty$

$a \in S_r = \{a | A \text{ has order } r\}$

$S_R = \{j | \Gamma^{(j)}_i[R, r] \to \Phi^{(j)}_i[R, r] \to A^{(j)} \text{ of order } r\}$

where we shall refer to the coefficients $\frac{1}{R!}$ as the Taylor coefficients, the α^j_{Rr} as derivative harmonics (harmonics), the γ^j_{Rr} as elementary weights (weights), and the Γ_i as elementary polynomials (polynomials). In Appendix I, the harmonics are tabulated in Table II, the weights in Table III, and the polynomials in Table IV. These tables are described in that appendix and their use is illustrated by means of examples in our subsequent work in Chapter VII, and will not be explained here. However, we note that having selected a generalized R.K. method, the nonlinear parameter defining equations can easily be obtained by simply looking up the quantities in (58) and writing down their linear combination.

In deriving the results, it was assumed that the conditions stated in (13') hold. These conditions constitute part of the system of equations that the parameters must satisfy and need to be written out explictly. For $r < \ell$

$$D^{r+1}x(0) = D^r(X \circ (u + v))(0) = D^r(X \circ u)(0)$$

$$D^{r+1}(X \circ \zeta)(0) = D^r(X \circ (u + \eta))(0) = D^r(X \cdot u)(0). \tag{59}$$

Substituting these results into (13') leads to

$$\sum_j g_{ij}\, x(0) + \sum_j \sum_{r=0}^{\ell-1} \frac{g_{ij}\,\theta_j^{r+1}}{(r+1)!} D^r(X \circ u)(0)$$

$$+ \sum_j \sum_{r=0}^{\ell-1} \alpha_{ij} \frac{\theta_j^r}{r!} D^r(X \circ u)(0) - x(0) - \sum_{r=0}^{\ell-1} \frac{\theta_i^{r+1}}{(r+1)!} D^r(X \circ u)(0) = 0. \tag{60}$$

Collecting terms in (60) and remembering that the validity of this equation for all functions X implies that the coefficients are separately zero, leads to the desired conditions.

In order that the results be easily accessible, we collect and summarize them below:

<u>Problem</u> $\quad Dx = X \circ x, \quad x(a) = b$

<u>Scheme</u> $\quad \xi_i = \sum_j g_{ij}\, \xi_j + \sum_j a_{ij}\, X(\xi_j), \quad i, j \in S = \{0, \ldots, e \times q\}$

<u>Condition A</u> $\quad \xi_i - u(\theta_i) = O(h^{\ell+1})$

for all i leads to

$$\sum_j g_{ij} - 1 = 0$$

$$\sum_j g_{ij} \frac{\theta_i^{r+1}}{(r+1)!} + \sum_j a_{ij} \frac{\theta_j^r}{r!} - \frac{\theta_i^{r+1}}{(r+1)!} = 0 \tag{61}$$

$$r = 0, 1, \ldots, \ell - 1, \; i \in S.$$

<u>Condition B</u> $\quad \xi_i - u(\theta_i) = O(h^m), \quad i = i_o \in S$ leads to

$$\sum_{R=r}^{\infty} \sum_{j \in S_R} \alpha_{Rr}^j \frac{\gamma_{Rr}^j}{R!} \Gamma_i^{(j)}[R, r] = \frac{\theta_i^r}{r!} \alpha_{rr}^a \tag{62}$$

for $r = \ell + 1, \ell + 2, \ldots, m - 1$

$a \in S_r$

where in the elementary polynomials $f_{ij}^{R-1} = a_{ij}$ for constructed approximations. For exact approximations $f_{ij}^{R-1} = 0$, $i \neq j$, $f_{ij}^{R-1} = \frac{\theta_i}{R}$, and $g_{ij} \equiv 0$.

The actual use of these results is comparatively easy once they have been tabulated in a general form. The real problem is to be quite precise in knowing what the scheme is and to understand in what sense one is asking for parameters. That there is an interpretation problem can easily be shown by means of a simple example. Consider the scheme

$$\xi_o = \xi_1 + a_{01} X(\xi_1) + a_{02} X(\xi_2).$$

In the use of the results, values for g_{1i}, f_{1i}, and g_{2i}, f_{2i} must be supplied. Usually, ξ_1 and ξ_2 are considered as exact approximations; that is, $x(\theta_1)$, $x(\theta_2)$ and then the parameters g_{1i}, g_{2i}, f_{1i}, f_{2i} are those which make $\zeta_1(1) = x_1(1)$ and $\zeta_2(1) = x_2(1)$. However, the results are not limited to such an interpretation. ξ_1 and ξ_2 could be considered as constructed approximations and, in that case, f_{1i} and g_{2i} are equal, respectively, to f_{0i}, g_{0i}, and the results give the error after two steps. What is essential for the present development is that some of the points be considered as exact approximations or else the infinite sum $\sum_{R=r}^{\infty}$ will be just that -- infinite. Otherwise, it is self-limiting and in reality a finite sum.

It also becomes apparent upon use of these results that the origin has never been specified, but must be specified to determine θ_i. Naturally, the results should not be origin dependent. That this is true is shown rather easily for the condition (61). In fact, it turns out that the equations associated with degree $s = 0$ are the same ones obtained if r is permitted to increase to $m - 1$ in (61). That is, the equations (61) for $r = \ell, \ell + 1, \ldots, m - 1$ are the equations of degree $s = 0$. Thus,

it can be shown that for any value of r (61) is origin independent. We state this as

Theorem 7:

The solution to the system of equations (61) for a given arbitrary r is origin independent.

Proof: Let r be given and let (61) possess a solution $\theta_0, \theta_1, \ldots, \theta_i, \ldots$. Choose a reference θ, say θ_0. Any θ can be expressed as $\theta_i = \theta_o + \alpha_i$ when α_i is a constant. Now $\theta_o = \theta_o(t)$ is a function of where the origin is set. For r = m, the maximum value, we have

$$f^{(m)}(\theta_o) = \sum_j \cdots \frac{\theta_j^{r+1}}{(r+1)!} + \sum_j \cdots \frac{\theta_j^{r}}{r!} - \frac{\theta_i^{r+1}}{(r+1)!} = 0 \quad , \quad r = m$$

$$Df(\theta_o) = f^{(m-1)}(\theta_o) = 0$$

$$\vdots$$

$$D^j f(\theta_o) = 0 \quad , \quad \text{all } j < \infty.$$

Thus, $f^{(m)}(\theta_o) \equiv 0$ as a function of $\theta_o = \theta_o(t)$ since the function value and all derivatives are zero. The results are obviously the same for $r < m$; hence, the theorem follows.

Corollary: The principal error term for degree s = 0 is a constant independent of the origin.

These results are necessary to have since, in comparing the present work with that of others, it may appear that the results disagree because for higher order error terms the location of the origin enters into the coefficients. It must be remembered that these coefficients multiply derivatives that are evaluated at the selected origin.

For the general set of equations, it is not so obvious that these results are actually independent of the choice of origin. However, all

results are polynomials in θ_i and since the result is not yet proved, in general, it can be explicitly verified for each case by taking a suitable number of values for θ_o and noting that a polynomial of degree n that has zeros are more than $n + 1$ points is the zero polynomial.

III. GENERALIZED RKF METHODS BY MEANS OF SUCCESSIVE SUBSTITUTIONS

In Chapter III, we shall treat the case of a system of p-th order ordinary differential equations written as

$$D^p x = X \circ \xi, \ \xi(a) = b, \ \xi = (D^{p-1}x, \ \ldots, \ D^o x).$$

Our goal will be to define for these systems a generalized Runge-Kutta-Frey (RKF) scheme that includes the previous definition when $p = 1$, and includes the methods using the derivatives of X as introduced by Frey,[7] and to furnish a direct means by which the nonlinear parameter defining equations may be obtained for a given scheme.

In order to arrive at the desired results, we introduce functions A, called differentials, which reduce to the previously defined differentials when $p = 1$.

These differentials are used to establish a generic, pattern-establishing set of symbols y which, when properly interpreted, furnish all the quantities needed to obtain our results. Derivative harmonics β_i are derived from the generic y, Definition 4, and we show by means of Theorem 4 that the derivatives of x can be written as $\sum_i \beta_i A_i$. Using the Taylor's series expansion of x, this result allows x to be written as $\sum_i \beta_i A_i$. We show, Theorem 6, that the operation of substitution $X(\xi)$ can be represented in the coefficient space of the differentials by means of substitution harmonics α which are also derived from the generic y. The operation of multiplication $DX(\xi) \cdot S$ can also be represented in the same coefficient space by means of multiplication harmonics obtained from the generic y, Theorem 7. It turns out that to use our results, we must be able to make a change of basis in the space of differentials.

This is accomplished by means of the translation harmonics, $\gamma(t)$, Definition 9. These, too, are obtained from the generic y.

With the ability to multiply and substitute, we can actually carry out in a direct manner the generation of any scheme and obtain its representation in the coefficient space of the differentials A. The class of schemes that are treated is given in Definition 10 which defines the generalized RKF scheme. The parameter defining equations are obtained by equating the components of the exact solution $\xi(\theta_i)$ and the corresponding approximation ξ_i.

Again, an assumption is made concerning the lower bound of the accuracy of all approximations ξ_i and the resultant equations are given as Theorem 10 where parameter conditions are stated.

These results are collected and summarized at the end of the chapter where we indicate how to actually go about obtaining the parameter defining equations from the scheme definition.

In Chapter II, we developed a formalism which, in effect, generates all the parameter defining equations for all the methods that are encompassed by Definition 2 of that chapter. When a particular scheme is considered, this formalism actually selects from the complete set of equations those that pertain to the chosen scheme. The approach might, thus, be characterized as a global approach to the nonlinear parameter defining equations associated with generalized R.K. schemes. It allows us to see the form of these equations and thereby enables us to discuss them; to investigate the various properties that they may have and, in general, furnishes a means by which they can be treated analytically. In short, it sets before us the whole collection of equations associated with the schemes and in such a fashion that we can, if we wish, talk

about them mathematically and see what they are like.

In this chapter, we shall develop a formalism that is distinctly different from the previous work in that the results can be characterized as local; that is, given any scheme, we shall show how this scheme can be used to generate the parameter defining equations associated with it. These equations will, in essence, belong to that scheme. Given any number of schemes, we can generate the equations associated with this set of schemes, but, again, they are only for this set. We have no information as to what the equations may look like for a scheme that we have not specified in detail. That is, we must specify the scheme to obtain the equations and we will only have equations for those schemes that have been written down; before we had all the equations, the specification of a scheme merely selected those which were of interest to us. This approach is, thus, not in itself very helpful if one wishes to understand the character of the equations associated with the generalized schemes. However, it is a very straightforward way of obtaining these equations and it will be seen that it can be extended rather simply to more general methods than have been considered here; whereas, the previous global approach is not as easily extendable. The present work is patterned after the work of R. DeVogelaere[3] and since the desired equations are derived by actually carrying out, in a very systematic fashion, the substitutions indicated by the scheme, we shall refer to this as a successive substitution approach. The previous chapter treated systems of first-order differential equations. We shall now consider systems of p-th order differential equations. The treatment of higher order differential equations requires that the notation used be more precisely defined than was previously done and that the problem under consideration be stated very precisely and explicitly.

The general case of higher order differential equations can be set forth as follows:

Let R be the real line, $t \in R$. Define the sets

$$N = \{0, 1, \ldots, n - 1\}, \quad P = \{0, 1, \ldots, p - 1\},$$
$$L = \{0, 1, \ldots, m, \ldots, n \times p - 1\} \tag{1}$$

where $m = i + kn$ with $i \in N$, $k \in P$. Let R^n and $R^{n \times p}$ be respectively real n and $n \times p$ dimensional vector spaces. Define $x \in R \to R^n$, $X \in R^{n \times p} \to R^n$, $\xi \in R \to R^{n \times p}$. More explicitly, define

$$\xi(t)\,[m] = D^{p-1-k}\, x(t)\,[i] \tag{2}$$

where $k = m \div n$, $i = m - k \times n$. It is desired to solve the ordinary differential equation initial value problem

$$D^p x = X \circ \xi$$
$$\xi(0) = a_L. \tag{3}$$

This can be written more explicitly as

$$D^p x(t)\,[j] = X(\ldots, D^{p-1-k} x(t)\,[i], \ldots)\,[j]$$
$$D^{p-1-k} x(0)\,[i] = a_L\,[m] \tag{4}$$

where $i, j \in N$, $k \in P$

$m \in L$, $k = m \div n$

$i = m - k \times n$

$t \in E \subset R \ni 0 \in E$.

In order to carry through the analysis, it is necessary to be precise about which elements we are dealing with. We shall use the following notation

$$
\begin{aligned}
&x_N \in (N \times R) \to R,\ x_N[i](t) = x_N(t)[i] = x(t)[i] \\
&D^p x_N[i](t) = D^p x_N(t)[i] = (D^p x)(t)[i] \\
&\xi_L[i](t) = \xi_L(t)[i] = \xi(t)[i] \\
&(X \circ \xi)_N[i](t) = (X \circ \xi)_N(t)[i] = (X \circ \xi)(t)[i] = X(\xi(t))[i].
\end{aligned} \tag{5}
$$

Using this notation, equation (4) can be written as

$$
\begin{aligned}
&D^p x_N = (X \circ \xi)_N \\
&\xi(0) = a_L.
\end{aligned} \tag{6}
$$

It is also convenient to carry through the convention of summing on double indices in the following manner. Let

$$K_{LN} \in (L \times N) \to R;$$

that is, a matrix with values $K_{LN}[m, i]$. We write

$$
\begin{aligned}
&W_L[m](t) = W_L(t)[m] \equiv K_{LN}[m] \cdot T_N(t) = \sum_{i \in N} K_{LN}[m, i] \\
&\qquad\qquad \cdot T_N[i](t) \\
&W_L[m] = K_{LN}[m] \cdot T_N \\
&W_L = K_{LN} \cdot T_N \\
&W_L(t) = K_{LN} \cdot T_N(t) \\
&W = K_{LN}\, T_N
\end{aligned} \tag{7}
$$

which is a consistent set of notation. We will, in general, make no distinction between the various functions that can arise by permuting the arguments; thus

$$W\,[m](t) = W(t)\,[m].$$

Also, the matrices

$$W\,[m, i] = W\,[m]\,[i]$$

will be considered equivalent. Of course, when both arguments of a function come from an index set, then we have that, in general,

$$W\,[m, i] \neq W\,[i, m].$$

Upper case subscripts are the sets defining the domain of definition of the arguments. They are usually omitted when this domain is the real line and usually included when the domain is an index set. Lower case subscripts and all superscripts are indexing quantities; that is, the names of the items. That is, we shall write $\xi_N^{(i)}$ or ξ_{iN} as convenient to indicate the i-th ξ_N in the set of all ξ_N. Whether or not an element is a function defined on R or a vector space point in R^n will be clear from the context or will be explicitly stated. In general, we shall use

$$x(\theta_i) = x_i = x_{iN} = x_N(\theta_i) \in R^n$$

as all being equivalent ways of writing the solution x of (3). We have used some of this notation in the work in Chapter II; however, for our present work, it is necessary that these quantities be precisely defined

so that there will be no misinterpretation of the results.

We proceed much as before. Now, however, our starting point is the approximation. The work that follows is patterned after that of Chapter II so that the results can be easily checked for the special case p = 1 of systems of first order differential equations in those areas that are common to both derivations. This also is a convenient way of contrasting the differences.

Thus, in the case of generalized R. K. schemes for systems of higher order differential equations, we again consider elements $\xi_i \in R^{n\times p}$ which are approximations to $\xi(t_i)$ where t_i is in the interval of interest. A generalized R. K. scheme is now defined as:

Definition 1: The approximation ξ_i is said to be obtained by means of a generalized R. K. scheme if, and only if

$$\xi_i = \sum_j A_{LL_1}^{(i,j)} \xi_{L_1}^{(j)} + \sum_j B_{LN}^{(i,j)} X(\xi_j)_N \qquad (8)$$

where $A_{LL} \in (L \times L \to R)$, $B_{LN} \in (L \times N \to R)$ and ξ_j is an approximation obtained in the same fashion.

In short, any scheme that uses a linear combination of approximations and of function values obtained at approximations is called a generalized R. K. scheme. At the present time, we say nothing further about the coefficient matrices A and B; however, it is obvious that a "sensible" choice of non-zero coefficients must be made when specifying a scheme. We shall say more about this later. It is easily seen from (8) that Definition 2 of Chapter II is contained in our present definition.

As was previously done, equation (8) can be rewritten as

$$\xi_i = u(\theta_i) + \sum_j A^{(i,j)}_{LL_1} [\xi^{(j)}_{L_1} - u(\theta_j)] + \sum_j B^{(i,j)}_{LN} [X(\xi_j)_N - R^{(j)}_N]$$

$$+ \sum_j A^{(i,j)}_{LL_1} u_{L_1}(\theta_j) + \sum_j B^{(i,j)}_{LN} R^{(j)}_N - u(\theta_i) \qquad (9)$$

where $R^{(j)}_N$ will be specified later, and

$$u_L[m] = \sum_{r=o}^{k+\ell} \frac{I^r}{r!} D^{p-1-k+r} x(0)[i] \qquad (10)$$

with $m = i + kn$

$i \in N$, $k \in P$

is the Taylor's expansion of the solution ξ up to and including the $p - 1 + \ell$ derivative. If the assumption is made that there is a lower bound ℓ for the lowest order of accuracy among the approximations ξ_i, then (9) which is an identity can be legitimately written as

$$\xi_i = u(\theta_i) + \sum_j A^{(i,j)}_{LL_1} [\xi^{(j)}_{L_1} - u(\theta_j)] + \sum_j B^{(i,j)}_{LN} [X(\xi_j)_N - R^{(j)}_N] \qquad (11)$$

$$\sum_j A^{(i,j)}_{LL_1} u_{L_1}(\theta_j) + \sum_j B^{(i,j)}_{LN} R^{(j)}_N - u(\theta_i) = 0. \qquad (11')$$

Equation (11) will be considered as the general form of the scheme and equation (11') will be considered as the conditions attached to the parameters A and B. Later we shall add another term to (11) when the scheme is generalized; that can, however, be easily done once the results are established for (11). To actually get started, consider that $A^{(i,j)}_{LL} \equiv 0$. Then (11) reduces to

$$\xi_i = u(\theta_i) + \sum_j B^{(i,j)}_{LN} [X(\xi_j)_N - R^{(j)}_N] . \qquad (12)$$

Equation (12) includes all Runge-Kutta schemes and their generalization

to a scheme with memory, provided one uses only function evaluations from the past. Equation (12) will be used in a very straightforward fashion to obtain the desired equations. We simply choose an index value for which the solution is desired, say t_i, and evaluate $\xi(t_i) = u(\theta_i) + v(\theta_i)$ and evaluate the approximation $\xi_i = u(\theta_i) + \eta_i$ by actually carrying out the indicated substitution.

It is seen that the principal problem is to satisfy

$$v(\theta_i) - \sum_j B_{LN}^{(i,j)}[X(\xi_j)_N - R_N^{(j)}] = O(h^r) \tag{13}$$

where

$$v_L[m] = \sum_{r=k+\ell+1}^{\infty} D^{p-1-k+r}x(0)\,[i]\,\frac{I^r}{r!} \tag{14}$$

and r is maximum in some sense. As was previously done, we must represent the derivatives of x and the expansion of $\eta_i = \xi_i - u(\theta_i)$ in the same basis so that a direct term-by-term comparison can be made. From (14) is seen that we need the derivatives $D^{p+r}x = D^r(X \circ \xi)$ for $r \geq \ell$. This task is accomplished in the same manner as before.

Let

$$\xi = u + v. \tag{15}$$

Then $X \circ \xi$ can be written as

$$X \circ \xi = X(u + v) = X \circ u + \sum_{s=1}^{\infty} \frac{1}{s!}(D_{L_1 \ldots L_s} X \circ u)v_{L_1} \ldots v_{L_s} \tag{16}$$

which when differentiated with respect to t gives rise to

$$D^r(X \circ \xi) = D^r(X \circ u) + \sum_{s=1}^{\infty} \frac{1}{s!} \sum_{i_1=0}^{r} \sum_{i_2=0}^{i_1} \cdots \sum_{i_s=0}^{i_{s-1}} \alpha_{ri_1 \cdots i_s}$$

$$D^{r-i_1}(D_{L_1 \cdots L_s} X \circ u)\, D^{i_1-i_2} v_{L_1} \cdots D^{i_s} v_{L_s} \tag{17}$$

where $$\alpha_{ri_1 \cdots i_s} = \binom{r}{i_1}\binom{i_1}{i_2} \cdots \binom{i_{s-1}}{i_s} .$$

We note that

$$D^r v_L [m] (0) = D^{p-1-k+r} x(0)[i] , \quad r \geq k + \ell + 1$$

$$= 0 , \quad r < k + \ell + 1.$$

Define

$$B^r = D^{p+r} x(0) \tag{18}$$

and

$$I^{(k)}_{LN} [m, i] = \delta_{m,i + kn} \tag{19}$$

where δ is the usual Kroneker delta. Then $D^r v_L$ can be written as

$$D^r v_L [m](0) = \sum_{k_1 \in P} I^{(k_1)}_{LN} [m]\, B_N^{r-k_1-1} . \tag{20}$$

The substitution of (18) and (20) into (17) yields

$$B^r_N = D^r(X \circ u)\,(0) + \sum_{s=1}^{\infty} \frac{1}{s!} \sum_{i_1=0}^{r} \cdots \sum_{i_s=0}^{i_{s-1}} \alpha_{ri_1 \cdots i_s} \sum_{k_1 \in P} \cdots \sum_{k_s \in P}$$

$$D^{r-i_1}(D_{L_1 \cdots L_s} X \circ u)(0)\, I^{(k_1)}_{L_1 N_1} \cdots I^{(k_s)}_{L_s N_s}\, B^{(i_1-i_2-k_1-1)}_{N_1} \cdots B^{(i_s-k_s-1)}_{N_s} . \tag{21}$$

The range of the indices in the index sets of (21) is excessive. By noting that $r \geq k + \ell + 1$, which can also be written as $k \leq r - \ell - 1$, we see that it is possible to write

$$\sum_{k_1 \in P} \cdots \sum_{k_s \in P} \quad \text{as} \quad \sum_{k_1=0}^{i_1-i_2-\ell-1} \sum_{k_2=0}^{i_2-i_3-\ell-1} \cdots \sum_{k_s=0}^{i_s-\ell-1}$$

which in turn implies that $i_s \geq \ell + 1$, ..., $i_2 \geq (s - 1)(\ell + 1)$, $i_1 \geq s(\ell + 1) \leq r$. This allows us to write the summations using the i indices as

$$\sum_{i_1=s(\ell+1)}^{r} \sum_{i_2=(s-1)(\ell+1)}^{i_1-(\ell+1)} \cdots \sum_{i_s=\ell+1}^{i_{s-1}-(\ell+1)} .$$

Thus, the normal range of the indices will be considered to be

$$\begin{aligned}
&0 \leq k_1 \leq i_1 - i_2 - (\ell + 1) \\
&0 \leq k_2 \leq i_2 - i_3 - (\ell + 2) \\
&\quad \vdots \\
&0 \leq k_s \leq i_{s-1} - (\ell + 1) \qquad (22) \\
&s(\ell + 1) \leq i_1 \leq r \\
&(s - 1)(\ell + 1) \leq i_2 \leq i_1 - (\ell + 1) \\
&\quad \vdots \\
&(\ell + 1) \leq i_s \leq i_{s-1} - (\ell + 1) \\
&1 \leq s \leq r \div (\ell + 1) \quad .
\end{aligned}$$

This set of indices will be referred to as the normal index set. We note that $i_1 - i_2$ ranges from $(\ell + 1)$ to r and, hence, $0 \leq k_1 \leq r - (\ell + 1)$. However, $k_1 \in P$ is always implicitly assumed, so $k_1 \leq p - 1$ is to be used should $r - (\ell + 1)$ be larger than $p - 1$. Similarly, $k_i \in P$ is implicitly assumed for all $i = 1, \ldots, s$. In all our subsequent reference

to (21), we shall assume that the indices are in the normal index set.

Formula (21) is analogous to (16') of Chapter II where we derived results for the case of systems of first order equations. We shall again proceed to define differentials of given order and degree and prove that the derivatives of x evaluated at zero have expansions into these functions evaluated at zero. We will then derive explicity recursive formulae for the coefficients of these expansions.

We now define a set of functions $A \in R \to R^n$ called differentials of given order r and degree s where r and s are integers such that $r \geq \ell + 1$ and $0 \leq s \leq (r - 1) \cdot (\ell + 1)$.

Definition 2: A is a differential of order r, degree s if, and only if

$$A_{N_o} = C^{(r_o,k_1,\ldots,k_s)}_{N_oN_1\ldots N_s} A^{(1)}_{N_1} \cdots A^{(s)}_{N_s} \tag{23}$$

where $A^{(i)}_{N_i}$ is a differential of order r_i, $r = 1 + r_o + \sum_{j=1}^{s} (k_j + r_j)$, $k_j \in \overline{P}$ and we define

$$C^{(r_o,k_1,\ldots,k_s)}_{N_oN_1\ldots N_s}[n_o, n_1, \ldots, n_s] = \tag{24}$$
$$D^{r_o}(D_{L_1\ldots L_s} X \circ u)_{N_o}[n_o] I^{(k_1)}_{L_1N_1}[n_1]\ldots I^{(k_s)}_{L_sN_s}[n_s]$$

with $\overline{P} = \{k | k \in P \wedge \xi[m] \text{ appears explicitly in } X \circ \xi\}$. For degree 0, C is to be interpreted as

$$C^{(r_o)}_{N_o} = D^{r_o}(X \circ u)_{N_o}. \tag{25}$$

It is necessary in the use of the differentials A to be sure that all of these quantities are actually identified and used. The questions that arise are concerned mainly with the generation of the A and the

identification of distinct A. We shall treat the latter first since in the generation of these quantities it is absolutely necessary to identify only the distinct A.

The identification of distinct A requires that we be able to determine the effect of permuting the k's and the A's in the definition of A_{N_o}. Let us consider the general term $A = \{r_o, k_1, \ldots, k_s; A^{(1)} \ldots A^{(s)}\} = C^{(r_o,k_1,\ldots,k_s)}_{N_o N_1 \ldots N_s} A^{(1)}_{N_1} \ldots A^{(s)}_{N_s}$ where we have again employed the shortened notation of the braces after the fashion of Butcher[1]. We wish to consider what happens when we derive another function $\bar{A} = \{r_o, \bar{k}_1, \ldots, \bar{k}_s, \bar{A}^{(1)} \ldots \bar{A}^{(s)}\}$ where the set $(\bar{k}_1, \ldots, \bar{k}_s)$ is a permutation of $(k_1, \ldots, k_s)$ and the set $(\bar{A}^{(1)}, \ldots, \bar{A}^{(s)})$ is a permutation of $(A^{(1)}, \ldots, A^{(s)})$. The question to be answered is the following: When is $A = \bar{A}$? Since any permutation can be arrived at by the repeated permutation of only two items, we shall examine in detail

$$A = (D_{L_1 L_2} X \circ u_{N_o})\,[n_o]\, I^{(k_1)}_{L_1 N_1}\,[n_1]\, I^{(k_2)}_{L_2 N_2}\,[n_2]\, A^{(1)}_{N_1}[n_1]\, A^{(1)}_{N_2}[n_2].$$

This can be written as

$$A = \{0, k_1, k_2; A^{(1)}\, A^{(2)}\} =$$

$$\sum_{n_1} \sum_{n_2} (D_{L_1 L_2}\,[n_1 + k_1 n]\,[n_2 + k_2 n]\, X \circ u)_{N_o}\, A^{(1)}_{N_1}[n_1]\, A^{(2)}_{N_2}[n_2] \quad i$$

$$A = \{0, k_2, k_1; A^{(1)}\, A^{(2)}\} =$$

$$\sum_{n_1} \sum_{n_2} (D_{L_1 L_2}\,[n_1 + k_2 n]\,[n_2 + k_1 n]\, X \circ u)_{N_o}\, A^{(1)}_{N_1}[n_1]\, A^{(2)}_{N_2}[n_2] \quad ii$$

$$A = \{0, k_1, k_2; A^{(2)}, A^{(1)}\} =$$

$$\sum_{n_1} \sum_{n_2} (D_{L_1 L_2}\,[n_1 + k_1 n]\,[n_2 + k_2 n]\, X \circ u)_{N_o}\, A^{(1)}_{N_2}[n_2]\, A^{(2)}_{N_1}[n_1] \quad iii$$

$$A = \{0, k_2, k_1; A^{(2)} A^{(1)}\} =$$

$$\sum_{n_1} \sum_{n_2} (D_{L_1 L_2} [n_1 + k_2 n] [n_2 + k_1 n] X \circ u)_{N_o} A^{(1)}_{N_2}[n_2] A^{(2)}_{N_1}[n_1]. \quad iv$$

Now *ii* can be written as

$$\sum_{n_1} \sum_{n_2} (D_{L_1 L_2} [n_2 + k_2 n] [n_1 + k_1 n] X \circ u)_{N_o} A^{(1)}_{N_2}[n_2] A^{(2)}_{N_1}[n_1]$$

since the sets N_1 and N_2 are identical and we sum over all indices of these sets. If the order of differentiation is reversed, then the derivative element will be that of *i*, but the value of $A^{(1)}_{N_1} A^{(2)}_{N_2}$ is incorrect. In fact, we see that *i* and *iii* are the same provided it is permissible to interchange the order of taking the derivative. We shall assume that this is justified. The same procedure applied to *iv* shows that *iv* and *i* are equivalent to each other. If we ask ourselves when are *i* and *ii* equivalent, we see that this is so if $k_1 = k_2$ or if $A^{(1)} = A^{(2)}$. In either case, *i* and *ii* are the same. We summarize these results as

Property 1: In the definition of the differentials A, a permutation of the $\{k_i\}$ with fixed $\{A^{(j)}\}$ is equivalent to a permutation of the $\{A^{(i)}\}$ with fixed $\{k_j\}$ and, in general, a new function will arise from this operation.

Property 2: In the definition of the differentials A, a permutation of the $\{k_i\}$ followed by an identical permutation of the corresponding $\{A_i\}$ does not lead to a new function. That is, any permutation of the couples $\{(k_1, A^{(1)}), \ldots, (k_s, A^{(s)})\}$ gives rise to the same A.

Combining Property 1 and 2 leads to

Property 3: If $k_i = k_j$, then the permutation of A_i and A_j does not give rise to a new A. If $A_i = A_j$, then the permutation of k_i and k_j does not give rise to a new function.

We use property 1 and 3 to obtain

Property 4: Given $A = \{r_o, k_1, \ldots, k_s; A^{(1)}\ldots A^{(s)}\}$ of order r, degree s, we obtain all distinct A of order r, degree s with the same factors by considering all the distinct permutations of $(k_1, \ldots, k_s)$ with $A^{(i)}$ fixed with the understanding that if $A^{(i)} = A^{(j)}$ then that permutation is not distinct. That is, we associate to the couples $(k_i, A^{(i)})$ the integer n_i. Two couples are considered identical if at least one of the components are equal; $k_i = k_j$ or $A^{(i)} = A^{(j)}$ or both. Distinct couples have distinct integers. We form the distinct permutations of $\{n_i\}$; then the correspondence $n_i \rightarrow k_i$ or $n_i \rightarrow A^{(i)}$ will furnish all the distinct A with the same factors.

As before, we can collect together the repeated $A^{(i)}$ and write

$$A = \{r_o, k_1, \ldots, k_s; A^{(1)}\ldots A^{(s)}\} = \{R; (A_1)^{\mu_1}\ldots(A_\sigma)^{\mu_\sigma}\} \qquad (26)$$

with $R = (r_o, k_1, \ldots, k_s)$, $s = \sum_{i=1}^{\sigma} \mu_i$. However, we must be careful to permute $(k_i, A^{(i)})$ together when arriving at this form which we shall refer to as the normal form of the differentials. We note that by appropriate indexing changes, we can always consider the indices as sequentially increasing. It is also convenient to use the same notation for A when all the A's are evaluated at a point; for example, in the expansion of $D^{p+r}x(0)$ we will evaluate them at 0. Whether they are to be considered as functions or points of R^n will be clear from the context. As was previously the case, the operator $\{;\}$ is linear and $\{R; \sum_i \alpha_i A_i\} = \sum_i \alpha_i \{R; A_i\}$, a fact which is needed in collecting terms in expansions subsequently used.

With regard to the establishment of a pattern of generation for these differentials, we refer the reader to Table V of Appendix I and the description of that table. However, a few comments can be given

here. As with our previous work of Chapter II, we proceed to order the differentials starting with those of lowest order and lowest degree. In fact, if $P = \{0\}$, then our pattern of generation is identical with that of the elementary differentials previously defined in Chapter II. If $P = \{0, \ldots, p - 1\}$, then we introduce the differentials with lowest k values and their permutations working from lowest degree to highest degree. As before, we have not established how the pattern of generation is to be continued beyond the orders and degrees that we have given in Table V. If these quantities are generated by an algorithm, as they can be, this pattern must be established in some fixed fashion and preferably in a manner that is consistent for all the different functions that we define, including the work of Chapter II, this chapter, and that of the following chapter.

Since the A's are functions of $R \to R^n$, it is possible to differentiate them and we have the following:

Theorem 1

The differential coefficient of any differential A of order r is a linear combination with non-negative integer coefficients of the differentials of order r + 1. That is,

$$DA = \sum_{i \in S_{r+1}} \alpha_i A_i \quad \text{where } S_{r+1} = \{i \mid A_i \text{ has order } r + 1\}.$$

Note that we only claim that $\alpha_i \geq 0$. In fact, it turns out that many of the coefficients are zero.

Proof: Consider the function A of order $r = \ell + 1$, degree 0. $A = \{\ell\} = D^{\ell}(X \circ u)$. $DA = D^{\ell+1}(X \circ u)$ and the theorem is true for the lowest order and degree. Now let $A = \{R; A_1 \ldots A_s\}$.

$$DA = \{r_o + 1, k_1, \ldots, k_s; A_1 \ldots A_s\} + \{R, DA_1 \ldots A_2\} + \ldots + \{R; A_1 \ldots DA_s\}.$$

$$= \{\bar{R}; A_1 \ldots A_s\} + \{R; \Sigma \alpha_i \bar{A}_i A_2 \ldots A_s\} + \ldots$$

$$= \{\bar{R}; A_1 \ldots A_s\} + \Sigma \alpha_i \{R; \bar{A}_i A_2 \ldots A_s\} + \ldots$$

where, by induction hypothesis, $\bar{A}_i$ is of order $r_i + 1$; thus each term has its order increased by one and the theorem is proved.

Corollary 1: The degree of A is invariant under differentiation.

These derivatives of A are relatively simple to calculate. The first few A's and their derivatives are given below:

$A^{(0)} = \{\ell\}$ $\qquad$ $DA^{(0)} = A^{(1)}$

$A^{(1)} = \{\ell + 1\}$ $\qquad$ $DA^{(1)} = A^{(3)}$

$A^{(2)} = \{0, 0; A^{(0)}\}$ $\qquad$ $DA^{(2)} = A^{(4)} + A^{(5)}$

$A^{(3)} = \{\ell + 2\}$

$A^{(4)} = \{1, 0; A^{(0)}\}$

$A^{(5)} = \{0, 0; A^{(1)}\}$

$A^{(6)} = \{0, 0: A^{(2)}\}$

$A^{(7)} = \{0, 1; A^{(0)}\}$

It is possible to develop a recursive formula for the derivative coefficients. Although the simple case given here has all coefficients equal to 0 or 1, this is not so when more derivatives are taken. With respect to higher order derivatives, the corresponding theorem is true.

Theorem 2

The j-th derivative of the differential A of order r, degree s is a liner combination with non-negative integral coefficients of the differentials of order r + j, degree s.

Proof: That the degree is preserved is obvious from Corollary 1. The theorem is already true for $j = 1$. For larger values of j, we have that $D^j A = D(D^{j-1}A)$ and the results follow by induction on j along with the application of Theorem 1.

We now state and prove the following theorem about the derivatives of x at zero; that is, $D^{p+r}x(o) = B^r$.

Theorem 3:

B_N^r of order $r + 1$ is the sum of the differentials A of order $r + 1$ evaluated at 0. That is

$$B_N^r = \sum_i \alpha_i A^{(i)} \quad \text{where } \alpha_i > 0.$$

Proof: The proof is an inductive one using (21). It is obviously true for $r = \ell$ since $B_N^\ell = D^\ell(X \circ u) = 1 \cdot \{\ell\}$. Now examine the second term of (21) and look at the sum of the orders. The order of the general term is $1 + r - i_1 + k_1 + \ldots + k_s + (i_1 - i_2 - k_1) + \ldots + i_s - k_s = r + 1$. Hence, the order is $r + 1$. The only question is whether using the definition of the differentials A, we will have a one-to-one correspondence between those A of order $r + 1$ and the quantities appearing in (21). Since the indices of (21) stay in the normal index set, all the A of order $r + 1$ appear in (21) and only these A's appear.

This theorem allows us to use the A as a basis for the derivatives and we now need to show how to obtain the coefficients α. It is possible to formulate a definition for these quantities that is almost identical to our previously derived derivative harmonics and, in fact, reduces to that definition when $p = 1$. However, since it is important to keep track of the distinct couples $(k_i, A^{(i)})$ in the generation of the differentials and likewise in the generation of the coefficients α_i, it is

more natural to define a generic set which establishes the pattern of generation and from which the quantities of interest can be easily derived by establishing a proper correspondence to this generic set. This has, in fact, been done in Table VIII of Appendix 1 which is the extension of the work of Chapter II to higher order differential equations. For our present work, the most convenient reference set is the set of differentials. Therefore, all our definitions will be referred to the A which will be used to establish a generic set. There will be no problem making the connection with the work of Chapter IV since there the A's are themselves easily identifiable with a generic, pattern establishing set of quantities. We emphasize this point because it seems quite natural that one concise algorithm could be used to generate all the quantities we have discussed and will subsequently discuss, provided it is kept in mind that they are all special cases of a basic generic set which establishes their pattern of generation.

In order to use the A as a generic set, we rewrite Definition 2 as

Definition 3: The symbol y is a generic y of order r, degree s, position 1 if

$$y^k[r, 1] = Y^k[r] \circ (z[r - 1, 0]). \tag{27}$$

The symbol y is a generic y of order r, degree $s > 0$, position a if

$$y^k[r, a] = Y^k[r_o, k_1, \ldots, k_s] \circ (z^{k_1}[r_1, a_1], \ldots, z^{k_s}[r_s, a_s]) \tag{28}$$

where $z^{k_i}[r_i, a_i]$ are generic z of order r_i, position a_i,

$$r = 1 + r_o + \sum_{i=1}^{s} (k_i + r_i), \quad k_i \in \overline{P}.$$

By convention, if the quantities generated do not depend on k, then the superscript k is suppressed.

It is assumed that the patterns establishing y and A are identical and that properties 1 - 4 are in effect here. In short, we have that if $\overline{P} = \{k | k \in P \wedge \xi [m] \text{ is explicit in } X \circ \xi\}$ and if

$$z [r - 1, 0] \equiv 1, \; z [r_i, a_i] \equiv y [r_i, a_i], \; a_i \neq 0$$

$$Y [r] \circ \equiv C_{N_o}^{(r-1)}.$$

$$Y [r_o, k_1, \ldots, k_s] \circ \equiv C_{N_o N_1 \ldots N_s}^{(r_o, k_1, \ldots, k_s)}.$$

then the differentials A of Definition 2 will be generated. The index a is the sequential position of y [r, a] in the set of order r with the assumption that some order has been established.

We now establish the coefficients α by means of Definition 4 and Theorem 4.

Definition 4: Define

$$z [r - 1, 0] \equiv 1, \quad z [r_i, a_i] \equiv y [r_i, a_i], \; a_i \neq 0$$
$$Y [r] \circ \equiv 1 . \tag{29}$$
$$Y [r_o, k_1, \ldots, k_s] \circ \equiv \frac{(r-1)!}{r_o!} \frac{1}{(\omega_1)! \ldots (\omega_\delta)!} \frac{1}{(k_1 + r_1)! \ldots (k_1 + r_s)!}.$$

where ω_i is the number of times that the triplet (k_i, r_i, a_i) appears in y [r, a]. Then the generic y has as its realization the derivative harmonics $\beta_{ra} = y [r, a]$.

Theorem 4:

Let $B_N^{r-1} = D^{r-1}(X \circ \xi)(0)$. Then we have that

$$B_N^{r-1} = \sum_{a \in S_r} \beta_{ra} A^{(r,a)}(0) \tag{30}$$

where β_{ra} is the derivative harmonic of Definition 4 and

$$S_r = \{a | A^{(r,a)} \text{ has order } r\}.$$

Corollary: $$D^{p+r-1}\dot{x}(0) = \sum_{a\in S_r} \beta_{ra}\, A^{(r,a)}(0). \tag{31}$$

Proof: This theorem is proved, as was its analogue in the p = 1 case, by substituting the expansions into both sides of the equation giving B_N, equation (21), and collecting the coefficients of like terms. Let $B_N^{r-1} = \sum_{a\in S_r} \beta_{ra}\, A_N^{(r,a)}$ where the order of A is r. The substitution of this into (21) gives

$$B_N^{r-1} = C_N^{r-1} + \sum_s \sum_{i_1\ldots i_s} \cdot \sum_{k_1\ldots k_s} \frac{1}{s!}\, \alpha_{ri_1\ldots i_s}\, C_{NN_1\ldots N_s}^{(r-i_1-1,k_1,\ldots,k_s)} \tag{32}$$

$$\cdot \sum_{a_1\ldots a_s} \beta_{i_1-i_2-k_1,a_1} \cdots \beta_{i_s-k_s,a_s}\, A_{N_1}^{(i_1-i_2-k_1,a_1)} \ldots A_{N_s}^{(i_s-k_s,a_s)}.$$

Let $A_N^{(r,a)} = \{r_o, \bar{k}_1, \ldots, \bar{k}_s; (A^{(\bar{r}_1,\bar{a}_1)})^{\mu_1} \cdots (A^{(\bar{r}_\sigma,\bar{a}_\sigma)})^{\mu_\sigma}\}$. We wish to determine the coefficient of the term on the righthand side of (32) corresponding to this term. The general term of the right member is $\{r - i_1, k_1, \ldots, k_s; A^{(i_1-i_2-k_1,a_1)} \ldots A^{(i_s-k_s,a_s)}\}$. It must, therefore, be true that $i_1 - i_2 - k_1 = \bar{r}_1, \ldots, i_s - k_s = \bar{r}_s$ which can be written as

$$\begin{aligned}
i_1 &= (\bar{r}_1 + \bar{k}_1) + (\bar{r}_2 + \bar{k}_2) + \ldots + (\bar{r}_s + \bar{k}_s)\\
&\vdots\\
i_j &= \qquad\qquad (\bar{r}_j + \bar{k}_j) + \ldots + (\bar{r}_s + \bar{k}_s)\\
&\vdots\\
i_s &= \qquad\qquad\qquad\qquad (\bar{r}_s + \bar{k}_s)\\
a_i &= \bar{a}_i \qquad i = 1, \ldots, s\\
k_i &= \bar{k}_i\,.
\end{aligned} \tag{33}$$

Thus, the factors of the term $A^{(r,a)}$ determine uniquely an index set (i, a, k). Property 2 tells us how to find all the terms corresponding

to $A^{(r,a)}$. We write the set of triplets $\{(k_i, r_i, a_i)_i\} = \{n_i\}$ where we associate distinct n_i with distinct triplets. Writing the repeated marks as $n_i^{\omega_i}$ allows us to write the set $\{(n_1)^{\omega_1}\ldots(n_\delta)^{\omega_\delta}\}$ where $s = \omega_1 + \ldots + \omega_\delta$. Any permutation of the $\{n_i\}$ corresponds to a permutation of the set of triplets and leads to a new set of indices and conversely. However, since we are permuting the whole triplet, this does not lead to any new differential A. The $\frac{s!}{\omega_1!\ldots\omega_\delta!}$ permutations of $\{n_i\}$ give us all the terms corresponding to $A_N^{(r,a)}$. That we get all the terms this way is evident if we consider (33) with the left member known and the right member unknown; the solution is unique.

The general coefficient of each term is

$$\frac{1}{s!}\ \frac{(r-1)!\ (\beta_{r_1 a_1})^{\mu_1}\ldots(\beta_{r_\sigma a_\sigma})^{\mu_\sigma}}{(r-1-i_1)!\ (i_1-i_2)!\ldots(i_{s-1}-i_s)!\ (i_s)!}\ .$$

Replacing $i_{j-1} - i_j$ by its value $r_j + k_j$ and multiplying by the multinomial coefficient gives rise to (29). In obtaining the index (i, a, k), one might wonder whether it is possible to choose $A^{(r,a)}$ such that the derived set (i, k, a) was not within the bounds of the normal index range (22). A careful examination of (22) and of Definition 2 will show that for any valid choice of $(\bar{k}, \bar{r}, \bar{a})$ we will derive by means of (32) a set (i, k, a) that is within the normal range. It is obvious that the β_{ra} that we have used here are simply the derivative harmonics of Definition 4 and could just as well have been stated as an integral part of the theorem; however, it is convenient to isolate the generation of these harmonics.

Definition 4 furnishes an explicit, recursive definition of the derivative harmonics. These coefficients will be used later to obtain the expansions of the derivatives of the solution x and then are tabulated in Table VI of Appendix I.

It is possible to give an analogous result for the coefficients of

Theorem 2. We state the results as

Theorem 5:

Let $A^{(r,a)} = \{r_o, k_1, \ldots, k_s; A^{(r_1,a_1)} \ldots A^{(r_s,a_s)}\}$ be the a-th differential in the set of all differentials of order r. Let

$$A^{(r+j,i)} = \{\bar{r}_o, \bar{k}_1, \ldots, \bar{k}_s; (A^{\bar{r}_1,\bar{a}_1})^{\mu_1} \ldots (A^{\bar{r}_\sigma,\bar{a}_\sigma})^{\mu_\sigma}\}$$

with $q = \sum_{i=1}^{\sigma} \mu_i$ be the i-th differential in the set of all differentials of order r + j. Define the coefficients δ by means of Theorem 2 to be

$$D^j(A^{(r,a)}) = \sum_{i \in S_{r+j}} \delta^{(a)}_{r+j,i} \; A^{(r+j,i)} \tag{34}$$

where $S_{r+j} = \{i \mid \text{all } A^{(r+j,i)} \text{ of order } j + r\}$. Then, the non-zero coefficients of this expansion are given by setting

$$\bar{k}_i = k_i, \; r_o \leq \bar{r}_o \leq r_o + j, \; q = s, \; r_i \leq \bar{r}_i \leq r_i + j$$

$$\delta^{(a)}_{r+j,i} = \frac{j!}{(\bar{r}_o - r_o)!} \sum_{i \in Q} \left[\frac{1}{(\bar{r}_1 - r_1)!} \cdots \frac{1}{(\bar{r}_s - r_s)!}\right]_i (\delta^{(a_1)}_{\bar{r}_1\bar{a}_1})^{\mu_1} \ldots (\delta^{(a_\sigma)}_{\bar{r}_\sigma\bar{a}_\sigma})^{\mu_\sigma} \tag{35}$$

where Q = {all permutations of $S_o = \bigcup_{i=1}^{\delta} S_i$ such that after any permutation $r_i \leq \bar{r}_i \leq r_i + j$}. The sets S_i are described below by equation (40). The set Q is essentially the set of distinct permutation of the set of couples $\{(\bar{r}_i, \bar{a}_i)\}$ corresponding to repeated $\bar{k}_i$. The coefficients δ in the right member of (35) are defined as the coefficients in the expansion of

$$D^{(\bar{r}_i - r_i)}(A^{(r_i,a_i)}) = \sum_{\bar{a}_i \in S_{\bar{r}_i}} \delta^{(a_i)}_{\bar{r}_i\bar{a}_i} \; A^{(\bar{r}_i,\bar{a}_i)}. \tag{36}$$

This theorem, in effect, states that if we know the derivatives of the lower order differentials, we can then find those of the higher order differentials.

Proof: Write $A = A^{(r,a)} = \{r_o, k_1, \ldots, k_s; A^{(r_1,a_1)} \ldots A^{(r_s,a_s)}\}$.

Differentiate to obtain

$$D^j A = \sum_{i_1=0}^{j} \sum_{i_2=0}^{i_1} \cdots \sum_{i_s=0}^{i_{s-1}} \alpha_{ji_1\ldots i_2} \{r_o + j - i_1, k_1, \ldots, k_s; D^{i_1-i_2}(A^{(r_1,a_1)}) \ldots D^{i_s}(A^{(r_s,a_s)})\}. \tag{37}$$

We know from Theorem 2 that $D^t(A^{(r_i,a_i)})$ is a linear combination with non-negative integral coefficients of the differentials of order $t + r_i$. Thus, we can replace each derivative in (37) by its expansion. This leads to

$$\sum_{i \in S_{r+j}} \delta^{(a)}_{r+j,i} \{\bar{R}; A^{(\bar{r}_1,\bar{a}_1)} \ldots A^{(\bar{r}_q,\bar{a}_q)}\} = \sum_{i_1=0}^{j} \sum_{i_2=0}^{i_1} \cdots \sum_{i_s=0}^{i_{s-1}} \alpha_{ji_1\ldots i_s} \cdot \sum_{\alpha_1 \ldots \alpha_s} \delta^{(a_1)}_{r_1+i_1-i_2,\alpha_1} \cdots \delta^{(a_s)}_{r_s+i_s,\alpha_s} \{R; A^{(r_1+i_1-i_2,\alpha_1)} \ldots A^{(r_s+i_s,\alpha_s)}\} \tag{38}$$

where $\bar{R} = (\bar{r}_o, \bar{k}_1, \ldots, \bar{k}_q)$ and $R = (r_o + j - i_1, k_1, \ldots, k_s)$.

A look at (38) shows us that many of the terms $A^{(r+j,i)}$ of order $r + j$ are missing - that is, have zero coefficients. We are interested only in the non-zero coefficients and this leads immediately to requiring that

$$q = s$$

$$r_o \leqq \bar{r}_o \leqq r_o + j$$

$$r_i \leqq \bar{r}_i \leqq r_i + j$$

which are the ranges stated in the theorem, and that

$$\begin{aligned} i_1 &= (\bar{r}_1 - r_1) + \ldots + (\bar{r}_j - r_i) + \ldots + (\bar{r}_s - r_s) \\ &\vdots \\ i_j &= \qquad\qquad (r_j - r_j) + \ldots + (r_s - r_s) \\ &\vdots \end{aligned} \tag{39}$$

$$
\begin{aligned}
i_s &= (\bar{r}_s - r_s) \\
\alpha_i &= \bar{a}_i \\
k_i &= \bar{k}_i \qquad i = 1, \ldots, s .
\end{aligned}
$$

We note that $A^{(r,a)}$ determines the k_i and we are not allowed to change them in (38). This, in turn, means that $\bar{k}_i$ are fixed in value. We wish to find permutations of $(\bar{r}_i, \bar{a}_i)$ that will lead to new index sets through the use of (39). However, we know from our previous work that we must permute the triplets $(\bar{k}_i, \bar{r}_i, \bar{a}_i)$ in order that the resultant differential not be changed. For the $k_i = \bar{k}_i$ that are alike, we can permute the $(\bar{r}_i, \bar{a}_i)$ without changing the differential $A^{(r+j,i)}$. Thus, we collect together the sets (r_i, a_i) into sets of like k. That is, we define

$$
\begin{aligned}
S_1 &= \{(\bar{r}_1, \bar{a}_1), \ldots (\bar{r}_{\omega_1}, \bar{a}_{\omega_1})\} \\
&\vdots \\
S_\delta &= \{(\bar{r}_\delta, \bar{a}_\delta), \ldots (\bar{r}_{\omega_\delta}, \bar{a}_{\omega_\delta})\}
\end{aligned}
\tag{40}
$$

where $(\bar{r}_j, \bar{a}_j)_\delta \in S_i$ iff $\bar{k}_j = \bar{k}_i$. We let $S_o = \{(\bar{r}_1, \bar{a}_1), (\bar{r}_2, \bar{a}_2), \ldots, (\bar{r}_s, \bar{a}_s)\} = \bigcup_{i=1} S_i$. We define Q to be the set of all distinct permutations of S_o where, by a permutation of S_o, we mean we permute the elements of each of the S_i. Let us write $S_i = \{(\bar{r}_1, \bar{a}_1)^{\gamma_1^{(i)}} \ldots (\bar{r}_{\zeta_i}, \bar{a}_{\zeta_i})^{\gamma_{\zeta_i}^{(i)}}\}$. Then we have

$$
\frac{(\omega_i)!}{(\gamma_1^{(i)})! \ldots (\gamma_{\zeta_i}^{(i)})!}
$$

permutations of S_i and, therefore,

$$
\frac{(\omega_1)! \ldots (\omega_s)!}{(\gamma_1^{(1)})! \ldots (\gamma_{\zeta_1}^{(1)})! \ldots (\gamma_1^{(\delta)})! \ldots (\gamma_{\zeta_\delta}^{(\delta)})!}
$$

permutations of S_o. Unfortunately, we cannot simply multiply the general

coefficient by this factor since

$$\alpha_{ji_1\cdots i_s} = \binom{j}{i_1}\binom{i_1}{i_2}\cdots\binom{i_{s-1}}{i_s} = \frac{j!}{(\bar{r}_o-r_o)!\,(\bar{r}_1-r_1)!\cdots(\bar{r}_s-r_s)!}$$

and, in general, this coefficient changes as we permute the $\bar{r}_i$ while leaving the r_i fixed. Also, it is necessary to restrict the set further since $r_i \leqq \bar{r}_i \leqq r_i + j$ must always be true. Thus, $Q = \{$all distinct permutations of S_o , $r_i \leqq \bar{r}_i \leqq r_i + j\}$. Equation (35) arises when the general term coefficient is extracted from (38).

As a biproduct of this theorem, we have the following

Corollary: Let $A = \{R; A_1 \cdots A_s\}$. Then R is invariant under differentiation.

Theorem 5 seems to be of little practical interest unless one is considering the generation of these expansions using a suitable constructed algorithm. In that case, it furnishes an explicit recursive definition that enables one to obtain the coefficients of the higher order derivatives provided all the coefficients of the lower order ones are known. The corollary is of interest because it assures us that if $k_i \in \bar{P} \subset P$, we need only concern ourselves with the differentials in the restricted set determined by $\bar{P}$. In actual practice, it is rather easy to obtain a reasonable number of these coefficients by actually carrying out repeated differentiation. It turns out that these differential coefficients are not the ones of interest to us and we shall make no further use of these results; they have been presented here for the sake of completeness.

Before turning to the task of generating the approximate solution by successive substitution, we first use our results to write the true solution ξ in the desired form; that is, in the form of an expansion in terms of the differential A evaluated at the origin.

We have that

$$\xi_L(\theta_j)[m] = u_L(\theta_j)[m] + \sum_{r=k+\ell+1}^{\infty} \frac{\theta_j^r}{r!} B_N^{r-k-1}[i] \tag{41}$$

where $m = i + kn$, $i \in N$, $k \in P$. We substitute into (41) the expansion for B_N^r and use the matrices $I_{LN}^{(k)}$ to obtain

$$\xi_L(\theta_j) = u_L(\theta_j) + \sum_{r=\ell}^{\infty} \sum_{a \in S_r} \Gamma_{LN}^{(j,r,a)} A_N^{(r,a)} \tag{42}$$

where

$$\Gamma_{LN}^{(j,r,a)} = \frac{\theta_j^{r+1}}{(r+1)!} \sum_{k \in P} \frac{\theta_j^k (r+1)!}{(r+1+k)!} \beta_{ra} I_{LN}^{(k)} \tag{43}$$

and $S_r = \{a | A^{(r,a)}$ is of order $r\}$. Equations (42) and (43) furnish the desired representation of ξ_L at θ_j. We shall return to these later after obtaining a representation for the approximate solution.

We shall take equation (11) as our starting point. In order to determine the parameters in the matrices A_{LL} and B_{LN}, it is sufficient that it be possible to write ξ_i in terms of the differentials A. With regard to this, we observe the following fact. If, in the construction of ξ_i, it is true that all ξ_j used in that construction have the form $\xi_j = u(\theta_j) + \Sigma_a \ldots A_N^{(a)}$, then $\xi_i = u(\theta_i) + \Sigma_a \ldots A_N^{(a)}$ will be true provided that $X(\xi_j) - R_N^{(j)} = \Sigma_a \ldots A_N^{(a)}$ and the condition established by (11') hold. We use this to our advantage by simply stating that any ξ_j which we use must have an expansion. How this expansion was obtained is, for the moment, irrelevant. We, of course, must at some stage of the development explicitly display the expansion coefficients or a means of obtaining them, but that will come later. For the present, we simply assume they exist. We note that many times when determining the parameters some of the ξ_i are chosen as exact values $\xi(\theta_i)$. In particular, for Runge-Kutta

methods the initial value $\xi_o = \xi(\theta_o)$ is chosen exactly, while the rest are constructed. With finite difference methods, all ξ_i except the final one can be chosen as $\xi(\theta_i)$. This is not, however, essential for our development and by simply assuming expansions exist, we gain a generality that will allow us to connect the development of a scheme by successive substitutions with the global scheme development. Also, this furnishes us with results that are easily extendable to other classes of methods and should give an insight on how to handle a global error analysis. We shall say more about these matters later.

We note from (11) that to proceed with the construction of ξ_j, we must be able to carry out the substitution $X(\xi_i)$ and that the result $X(\xi_i) - R_N^{(i)}$ must be expandable into the differentials A. We shall thus define $R_N^{(i)}$ as that part of $X(\xi_i)$ which does not possess such an expansion. To carry out the substitution, we shall define substitution harmonics as follows:

Definition 5: Let there be given an element $T_L \in R^{n\times p}$ written as

$$T_L = \sum_{r\in S} \sum_{a\in S_r} \sum_{k\in\bar{P}} \alpha_{rka}\, I_{L_N}^{(k)}\, A^{(r,a)} \quad \text{where}$$

$\bar{P} = \{k \mid \xi^{(k)}$ is explicit in $X \circ \xi\}$. Define

$$z\,[r-1,\,0] \equiv 1,\quad z^{k_i}[r_i,\,a_i] = \alpha_{r_i k_i a_i},\ a_i \neq 0$$

$$Y\,[r]\circ \equiv \frac{\theta^{r-1}}{(r-1)!}\,. \tag{44}$$

$$Y\,[r_o,\,k_1,\,\ldots,\,k_s]\circ (z^{k_i}[r_1,\,a_1],\,\ldots,\,z^{k_s}[r_s,\,a_s]) \equiv$$

$$\frac{\theta^{r_o}}{r_o!}\ \frac{\alpha_{r_1 k_1 a_1}\cdots\alpha_{r_s k_s a_s}}{(\omega_1)!\cdots(\omega_s)!}\,.$$

where ω_i is the number of times the triplet $(k_i,\ r_i,\ a_i)$ appears in

$y[r, a]$ and $s = \sum_{i=1}^{\delta} \omega_i$. Then the generic y has as its realization the substitution harmonic $\beta_{ra} = y[r, a]$ corresponding to T_L.

We can then carry out the substitution using

Theorem 6:

Let $z_N = X(u(\theta) + T_L) - R_N$

where
$$R_N = \sum_{i=o}^{\ell-1} \frac{\theta^i}{i!} D^i(X \circ u)(0)$$

$$T_L = \sum_{r\in S} \sum_{a\in S_r} \sum_{k\in \overline{P}} \alpha_{rka} I_{LN}^{(k)} A_N^{(r,a)}.$$

Then, $z_N = \sum_{r\in S} \sum_{a\in S_r} \beta_{ra} A^{(r,a)}$ where β_{ra} are the substitution harmonics corresponding to T_L.

Thus, given the harmonics of the element T_L we can find, in terms of these harmonics, the corresponding harmonics of z_N. It is interesting to note that with the particular realization given in Definition 5, we can consider the operator $Y \circ z$ to be the substitution operator in a space whose bases consists of the Differentials A.

Proof: The proof is straightforward. We can write

$z = X(u(\theta) + T_L)$ as

$$z = (X \circ u)(\theta) + \sum_s \frac{1}{s!} (D_{L_1 \ldots L_s} X \circ u)(\theta)\, T_{L_1} \ldots T_{L_s}. \tag{45}$$

Substitute into (45) the expansion for T_L. This gives

$$z = \sum_{\ell\in\omega} \{\ell\} \frac{\theta^\ell}{\ell!} + \sum_{s\in\omega_1} \sum_{r_o\in\omega} \sum_{r_1 \ldots r_s \in S} \sum_{a_1 \ldots a_s \in S_{r_j}} \sum_{k_1 \ldots k_s} \cdot \tag{46}$$
$$\frac{1}{s!} \frac{\theta^{r_o}}{r_o!} \alpha_{r_1 k_1 a_1} \ldots \alpha_{r_s k_s a_s} \{r_o, k_1, \ldots, k_s; A^{(r_1,a_1)} \ldots A^{(r_s,a_s)}\}$$

where $\omega = \{0, \ldots, \infty\}$, $\omega_1 = \{1, 2, \ldots, \infty\}$. As usual, we write the general

term on the right side as $\{\bar{r}_o, \bar{k}_1, \ldots, \bar{k}_s; A^{(\bar{r}_1,\bar{a}_1)} \ldots A^{(\bar{r}_q,\bar{a}_q)}\}$ and look for the coefficients of this term on the righthand side. The terms on the right are found by forming all the distinct permutations of the triplets $(\bar{k}_i, \bar{r}_i, \bar{a}_i)$.

If the process is carried out, it will be seen that to each $A^{(r,a)}$ there corresponds a coefficient β_{ra} whose pattern of generation is identical to that of the A. The problem is simply to identify the correct realization of y and z in Definition 4. This is seen to be Definition 5.

We note that to stay within the set of differentials that have been defined, it is necessary to subtract off

$$\sum_{r=0}^{\ell-1} \{r\} \frac{\theta^r}{r!} = \sum_{r=0}^{\ell-1} D^r(X \circ u)(0) \frac{\theta^r}{r!} = R_N.$$

We now use this to define the $R_N^{(j)}$ of (11) and (11').

Definition 6:
$$R_N^{(j)} = \sum_{r=0}^{\ell-1} D^r(X \circ u)(0) \frac{\theta_j^r}{r!}$$

$$R_N^{(j)} = \sum_{r=0}^{\ell-1} D^{p+r} x(0) \frac{\theta_j^r}{r!} . \qquad (47)$$

We shall need these explicit results when actually writing out the conditions that (11') represent.

It is now actually possible to carry out any substitution that a particular scheme may require. The result of that substitution is in the space spanned by the differentials. That is, the space spanned by the differentials A is closed under the operation of substitution. At this point of the development, it is natural to consider another operation, that of multiplication by the Jacobian matrix associated with X. We shall develop a multiplication theorem and show that the space spanned by the differentials A is also closed under multiplication.

What we have in mind is the following. We wish to be able to treat

schemes that make use of the Jacobian matrix $(D_L X)_N$ associated with X. This will allow us to treat schemes that are of the Frey[7] type and generalizations of these schemes.

To be more precise, define $(D_L X)_N \in R^{n\times p} \to ((L\times N) \to R)$ and then

$$J_{NL}[j][m] = (D_L X)_N(\xi)[m,j] = D_m X(\xi_1, \ldots, \xi_{n\times p})[j] \tag{48}$$

where $m = i + kn$, $i \in N = \{0, \ldots, n-1\}$, $k \in P = \{0, \ldots, p-1\}$, $j \in N$ is the matrix of partial derivatives of X evaluated at some point. We can multiply J_{NL} times any element of $R^{n\times p}$ and we propose to create new approximations by using the products $J_{NL}\, S_L$ where the S_L are constructed in a manner to be described shortly. This idea is not in itself new[7], but its full utilization is complex and while we will not attempt to use all the variants of the schemes it suggests, we will set up the necessary formalism for obtaining any particular scheme that one might wish to investigate. What is needed is a multiplication theorem and this is arrived at in a fashion quite similar to the substitution theorem.

Definition 7: Let there be given the elements T_L and $S_L \in R^{n\times p}$ written as

$$\begin{aligned} T_L &= \sum_{r\in S} \sum_{a\in S_r} \sum_{k\in \overline{P}} \gamma_{rka}\, I_{LN}^{(k)}\, A_N^{(r,a)} \\ S_L &= \sum_{r\in S} \sum_{a\in S_r} \sum_{k\in P} \alpha_{rka}\, I_{LN}^{(k)}\, A_N^{(r,a)} \end{aligned} \tag{49}$$

where $\overline{P} = \{k \mid \xi^{(k)}$ is explicit in $(D_L X)_N\} \subset P = \{0, \ldots, p-1\}$. Define

$$\begin{aligned} &z[r-1, 0] \equiv 0,\ z^{k_1}[r_1, a_1] = \alpha_{r_1 k_1 a_1},\ k_1 \in P \\ &z^{k_i}[r_i, a_i] = \gamma_{r_i k_i a_i},\ k_i \in \overline{P},\ i = 2, \ldots, s \\ &Y[r-1]\circ \equiv 1 \cdot \\ &Y[r_o, k_1, \ldots, k_s] \circ (z^{k_1}[r_1, a_1], \ldots, z^{k_s}[r_s, a_s]) \equiv \end{aligned} \tag{50}$$

$$\frac{\theta^{r_o}}{r_o!}\ \frac{1}{(s-1)!}\sum_{i\in Q}[\alpha_{r_1k_1a_1}\gamma_{r_2k_2a_2}\cdots\gamma_{r_sk_sa_s}]_i\ .$$

where Q = {all distinct permutations of the triplets (k_i, r_i, a_i)}. Then the generic y has as its realization the multiplication harmonics $\beta_{ra} = y[r, a]$ corresponding to T_L and S_L.

We can carry out the desired multiplication using

Theorem 7:

Let T_L and S_L be as defined by (49). Let $J_{NL} = D_L X(u(\theta) + T_L)$ and $W_N = J_{NL}S_L$. Then, $W_N = \sum_{r\in S}\sum_{a\in S_r}\beta_{ra}A_N^{(r,a)}$ where β_{ra} are the multiplication harmonics corresponding to T_L and S_L.

Proof: This is proved in a straightforward fashion by performing the expansions and collecting the terms. Since we have carried through in detail the other proofs, we shall be quite brief here. Start with

$$D_L X(u(\theta) + T_L) = \sum_{s\in\omega}\frac{\theta^s}{s!}D^s(D_L X\circ u)(0) + \sum_{s\in\omega_1}\frac{1}{s!}\sum_{q\in\omega}\frac{\theta^q}{q!}$$

$$\cdot\, D^q(D_{L_1\ldots L_s}D_L X\circ u)(0)\,T_{L_1}\ldots T_{L_s}\ .$$

Substitute for T_L and multiply by S_L to obtain

$$W_N = \sum_{s\in\omega}\sum_{r\in S}\sum_{a\in S_r}\sum_{k\in P}\{s, k; A^{(r,a)}\}\frac{\theta^s}{s!}\alpha_{rka} + \tag{51}$$

$$\sum_{s\in\omega_2}\sum_{q\in\omega}\sum_{r_1\ldots r_s}\sum_{a_1\ldots a_s}\sum_{k_1\ldots k_s}\alpha_{r_1k_1a_1}\gamma_{r_2k_2a_2}\cdots\gamma_{r_sk_sa_s}$$

$$\frac{\theta^q}{q!}\ \frac{1}{(s-1)!}\cdot\{q, k_1, \ldots, k_s; A^{(r_1,a_1)}\ldots A^{(r_q,a_q)}\}.$$

As was previously done, we isolate the general term of the righthand side and an examination of the pattern of construction of the β_{ra} will show us that they are as defined by Definition 7.

It is interesting to note that, to a certain degree, the multiplication

by J_{NL} is equivalent to a substitution. This seems encouraging until we note that all zero degree terms have $\beta_{r1} = 0$. This proves unfortunate since in trying to obtain higher order methods we find that the zero degree terms are exactly those that cause difficulties by being inconsistent for a fixed number of θ_j. The evaluation of J_{NL} does not overcome this difficulty since the point θ at which the evaluation is carried out does not enter into these terms, nor likewise do any of the $J_{NL}S_L$ that we create help us for these zero degree terms.

Again, we see that the space spanned by the differentials is closed under multiplication by J_{NL} and that, as defined in Definition 7, the realization of $Y \circ z$ is the multiplication operator in the coefficient space.

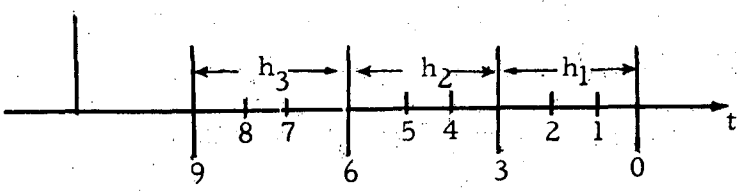
The substitution and multiplication harmonics along with the derivative harmonics have been tabulated in Table VI of Appendix I.

We are now in a position to describe a constructive approach to the generation of the nonlinear parameter defining equations associated with any particular scheme we may wish to investigate. Thus, we choose a scheme and actually carry out the indicated operations of substitution, linear combination, etc., thereby generating the associated equations. Since the work is actually carried out by an explicitly defined algorithm, this turns out to be an extremely easy way to obtain these quantities. We shall describe this generating alorithm later. We proceed now to explicitly define how the approximations should be ordered so that we will have a common systematic arrangement for all schemes. The ordering and definitions that appear below are to be considered to hold for all our work; in particular that of Chapter II and IV.

The interval of the independent variable, here considered to be t,

is discretized. The points of the discrete set are indicated as t_i. It is convenient to classify some of these points as major points and some as minor points. This is schematically shown in Figure 1 which we shall refer to at various times to help describe our organization of the approximations.

Figure 1:



In Figure 1, the major points have been indicated by a large mark and consist of t_0, t_3, t_6, t_9, while the minor points are t_1, t_2, t_4, t_5, t_7, and t_8. In general, the name rank is assigned to the number of points in one major interval. This is not quite true for the last major interval since if we count the last point, we have one extra point; but, in general, there are q points per major interval where q is the rank of the method. Conceptually, the difference between major and minor points is characterized by the fact that the minor points are constructed; that is, they are assumed to have a constructive representation, while the major points are assumed to have "known" expansions. For example, when deriving the parameter defining equations for a particular scheme, it is sufficient to take the major points t_q, t_{2q}, t_{3q} ... as points for which the approximation is exact, $\xi_i = \xi(\theta_i)$, while at the minor points the approximation is constructive $\xi_i = \Sigma_j \cdots \xi_j + \Sigma_j \cdots X(\xi_j)$. For the exact approximations, the expansion is known to be the usual Taylor's expansion when we replace the derivatives by their expansions into the basic set of differentials $A^{(a)}$.

We do not, in general, require that the sequential ordering of the t_j will correspond to the actual ordering by magnitude when the position is allowed to be a parameter of the method and is then solved for. In fact, the minor points may actually lie outside the major interval with which they are associated. Witness Chapter VII, where an example is given for which this happens. We simply state that we have approximate solutions ξ_{nq}, n = 0, 1, 2 ... which are associated with major points t_{nq} in the sense that $\xi_{nq} = u(t_{nq}) + \ldots$ is an approximation to $\xi(t_{nq})$, and that there are other approximations $\xi_i = u(t_j) + \ldots$, which are approximations to $\xi(t_j)$, associated with the minor points t_j where $n \times q < j < (n + 1) \times q$ means that t_j is associated with the major interval n + 1. With regard to Figure 1, the rank q = 3, the first major interval, consists of the points t_o, t_1, t_2; the second major interval of the points t_3, t_4, t_5; and the last major interval of the points t_6, t_7, t_8. While t_9 actually belongs to the next major interval, it furnishes a natural boundary to the number of intervals and is included in the scheme. In practice, it is usually the case that the major points are the most accurately known and are all of equal order of accuracy with regard to their local truncation error, while the minor points are the constructed approximations that are of varying degrees of accuracy. We shall see this in more detail later.

Now to any scheme there is associated a period after which the scheme is repeated. We call this the <u>period</u> and by definition the period is the number of major intervals after which the scheme is repeated. That is, to any approximation ξ_i outside the interval defined by p×q where p ≡ the period and q ≡ the rank, there is an identically defined approximation ξ_j within the period. For example, with regard to Figure 1, if p=2 and q=3,

then $\xi_7 = \xi_1$ in the sense that the constructive representation of ξ_7 and ξ_1 are identical; that is, the coefficients and index sets defining $\xi_7 = \Sigma \ldots \xi_j + \Sigma \ldots X(\xi_j)$ are identical to those defining $\xi_1 = \Sigma \ldots \xi_j + \Sigma \ldots X(\xi_j)$.

For schemes that have used previous approximations; that is, generalized R.K. schemes with memory, there must be a way of indicating how far back the scheme extends. We define the <u>extent</u> of the scheme to be the number e such that eXq, where q is the rank, is the totality of points considered in the scheme.

Lastly, for the present, we have associated with each major interval the distance $h_i = (t_{(i-1)q} - t_{iq})$ which we define as the <u>major interval</u>.

We collect together here these definitions for future reference and shall refer to them as scheme interval parameters. These are a subset of the totality of parameters which define a particular scheme.

<u>Definition 8</u>: We define for a particular generalized R. K. scheme

⟨approximation⟩ :: = ⟨any approximation from the generalized R. K. scheme⟩

⟨point⟩ :: = ⟨any t_i for which there is an approximation⟩

⟨major point⟩ :: = ⟨a point for which the approximation has a "known" expansion⟩

⟨minor point⟩ :: = ⟨a point for which the approximation is constructed⟩

Then, the <u>scheme interval parameters</u> (SIP) are

1) ⟨major interval h_j⟩ :: = ⟨the distance $(t_{(i-1)q} - t_{iq})$ between two major points⟩

2) ⟨period⟩ :: = ⟨the number of distinct major intervals⟩

3) ⟨rank q⟩ :: = ⟨the number of points (approximations) in one major interval⟩

4) ⟨extent e⟩ :: = ⟨number of major intervals in the scheme⟩

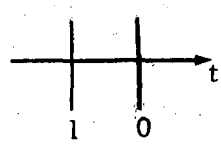
In practice, the major interval h_i is usually fixed at h, the step size, throughout the extent of the scheme, the period is usually 1 and the minor points usually lie within the closed major interval. There is, of course, nothing new in organizing the scheme in the fashion we have indicated. The scheme interval parameters are written out explicitly so that we will have a consistent, systematic approach with which to classify schemes. The workers in this field have, to-date, been somewhat undecided in their various use of terminology for the same ideas using such terms as rank, height, stage, etc. In the foregoing, we have strived to present the simplest set that will serve our purpose.

Before furnishing some simple examples, we note that the major points can be interpreted in a variety of ways. We shall, when obtaining the parameter defining equations, assume that these points are exact solution values. However, suppose that one were interested in obtaining the cumulative error after, say, two steps. Then this could be accomplished by considering the scheme to have the same formal construction, carrying out the construction for two major steps and, thus, letting the "known" expansion at ξ_q be the constructed expansion. Such a scheme would have an extent of $e = 2$. The same results could also be obtained by considering a scheme with a major interval twice as large as the first and then letting one of the constructed minor points be the intermediate solution value.

Suppose one wished to simply consider that the major points had approximations $\xi_i = u(\theta_i) + \varepsilon(\theta_i, h)$ when ε_i is the error. Then our formalism should carry through provided that one can obtain an expansion of the error term $\varepsilon(\theta_i, h)$. We shall say more later on how this could be taken care of.

We illustrate our classification using three simple examples. First, note that purely Runge-Kutta schemes all have an extent e = 1, a rank q of various values, and a period = 1. While finite difference methods always have a rank q = 1, a period = 1, and an extent e of various values.

1. For Euler's method, we would have



SIP 1) h

2) period = 1

3) rank q = 1

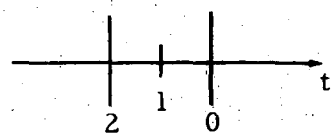
4) extent e = 1

Scheme $\eta_o = X(\xi_1)$

$\xi_o = \xi_1 + B_o\eta_o$.

The parameter B_o is chosen so that $\xi_o - \xi(\theta_o) = O(h^2)$.

2. A simple Runge-Kutta scheme would be as follows:



SIP 1) h

2) period = 1

3) rank q = 2

4) extent e = 1

Scheme $\eta_o = X(\xi_2)$

$\xi_1 = \xi_2 + B_o\eta_o$; $\eta_1 = X(\xi_1)$

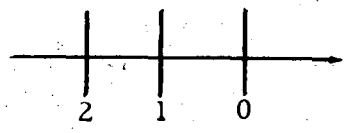
$\xi_o = \xi_2 + B_1\eta_o + B_2\eta_1$.

The parameters B_o, B_1, B_2 are adjusted so that $\xi_o - \xi(\theta_o) = O(h^3)$, that is a second order method. We see immediately that Euler's method is contained in the scheme, $B_o = B_2 = 0$.

3. A simple finite difference method of the Adams type is given as

SIP 1) $h_i = h$

2) period = 1



3. contd.

3) rank q = 1

4) extent e = 2

Scheme $\eta_o = X(\xi_2)$; $\eta_1 = X(\xi_1)$

$\xi_o = \xi_1 + B_o\eta_o + B_1\eta_1$

The parameters B_o and R_1 are adjusted so that $\xi_o - \xi(\theta_o) = O(h^3)$.

With this type of classification there is no real distinction between Runge-Kutta methods, finite difference methods, or predictor corrector methods provided the last mentioned ones are used with only a few iterations, as is usually the case in practice.

Our basic approach in this chapter is, as we have already mentioned, to construct the parameter defining equations by actually carrying out the substitutions. Using example 2, we would write

$\xi_2 = u(\theta_2) + \sum_i \beta_i^{(2)} A^{(i)}$ a major point has a "known" expansion

$\eta_o = R_2 + \sum_i \delta_i^{(2)} A^{(i)}$ by the substitution theorem

$\xi_1 = u(\theta_1) + \sum_i [\beta_i^{(2)} + B_o\delta_i^{(2)}]A^{(i)}$ with conditions on B_o

$\eta_1 = R_1 + \sum_i \delta_i^{(1)} A^{(i)}$ by the substitution theorem

$\xi_o = u(\theta_o) + \sum_i [\beta_i^{(2)} + B_1\delta_i^{(2)} + B_2\delta_i^{(1)}]A^{(i)}$ with conditions on B_1, B_2

which is to be matched with

$\xi(\theta_o) = u(\theta_o) + \sum_i \gamma_i A^{(i)}$.

An examination of this example shows immediately that while the scheme itself is easy to write down, it would be extremely tedious to carry out the actual computation in all its detail. However, it can be carried out systematically with the aid of a set of appropriate substitution tables. We take advantage of this fact and write the defining algorithms

as ALGOL 60 procedures so that all we need ever do is write down the scheme, in some manner to be defined later, and having furnished the tables, the results of the substitution are furnished to us as output.

The next thing to notice is that an origin must be chosen for $u(\theta_2)$ and $A^{(i)}$. The choice normally made is, for this example, $\theta_2 = 0$ since we consider the solution known at $\theta_2 = t_2$ and we wish to advance one h interval, one major step, using one intermediate calculation. However, since we are considering points from the past, it turns out that it is more convenient to choose θ_0 as the origin. That is, the origin is set at the next point forward where we wish to know the solution. The principal advantage in such a choice is that all the θ_i's are the same sign and this helps in constructing the program. If we wish to compare our results directly with those of the global approach, then none of the θ_i's should be set to zero before the equations are generated. The best procedure is to leave the actual choice of the values of θ_i open until such time as the equations defining the parameters are to be solved. At that time, the θ_i's are made explicit. If this is done, then the results are easily comparable with our previous work and also the full symmetry of the equations presents itself and this, in itself, may prove useful in finding a solution. The disadvantage to this is that our results are not directly comparable with the work of others unless we make the same choice of origin as they have made. This, however, is a minor detail since we need simply choose the appropriate $\theta_i = 0$ or else directly verify the results by substitution of the values of the parameters. This latter process holds only through the principal error term since higher order terms are origin dependent.

We see that we are immediately in need of the coefficients $\beta_i^{(2)}$.

We have chosen $\xi_2 = \xi(\theta_2)$ so these are the product of the Taylor coefficients and the derivative harmonics. However, for all schemes with memory this problem will present itself; we shall need the coefficients in the expansion and they will not necessarily be available.

Therefore, we now address ourselves to the problem of finding the expansion parameters for those approximations ξ_j that are used in constructing ξ_i, but do not themselves have a constructive expansion, or do not yet have a constructive expansion. We might note that the global approach of Chapter II seems to have side-stepped this problem entirely. We say "seems" because it will subsequently be shown that a direct connection can be made between the global and substitution approach.

The solution to our problem is deceptively simple. The expansion is simply written down with undetermined parameters and the corresponding approximation is used as if the coefficients were well known. Thus, the problem becomes a new one. We need to be able to eventually establish the values of the undetermined parameters. The solution to this problem can best be obtained by looking further at Example 2 and examining ξ_2 more closely. Once we have established what it is that needs to be done, we shall derive the results in general and in detail. For the time being, we shall continue to use the abbreviated notation that has been used in Examples 1 - 3; notation that is correct for $p = 1$ and which is still correct for $p > 1$ provided the elements are interpreted to be in the correct spaces.

Let us return to the definitions of u and v, equations (10) and (14), respectively. We see that they have been written down as if ξ were expanded about the origin. If an arbitrary origin τ had been chosen, then they would have been written as

$$u_L[m](\tau, t) = \sum_{r=o}^{k+\ell} \frac{t^r}{r!} D^{p-1-k+r} x(\tau)[i]$$

$$v_L[m](\tau, t) = \sum_{r=k+\ell+1}^{\infty} \frac{t}{r!} D^{p-1-k+r} x(\tau)[i]. \tag{52}$$

Then $\xi(t) = u_L(\tau, t - \tau) + v_L(\tau, t - \tau)$ would have been the case. All the differentials A would then be $A(\tau, t)$ and we would be talking about expansions of $D^{\ell}(X \circ \xi)(\tau)$ in terms of $A(\tau, 0)$. We have developed all our results with $\tau = 0$, but they hold equally well for any other value of τ.

There are essentially three ways of treating the $\beta_i^{(2)}$ of Example 2. We can write

$$\xi_2 = u(0, \bar{\theta}_2) + \sum_i \bar{\beta}_i^{(2)} A^{(i)}(0, 0) \tag{53}$$

where the $\bar{\beta}_i^{(2)}$ are undetermined parameters and use ξ_2 as it is. Then we translate the origin and write

$$\xi_2 = u(\tau, \theta_2) + \sum_i \beta_i^{(2)} A^{(i)}(\tau, 0) \tag{54}$$

where $\beta_i = T(\bar{\beta})$ are functions of the $\bar{\beta}_i$ and the translation interval, while $\theta_2 = \bar{\theta}_2 - \tau$. In the present example, $\xi_2 = \xi(\bar{\theta}_2)$ the real solution; but, $\xi(\bar{\theta}_2)$ always has an expansion

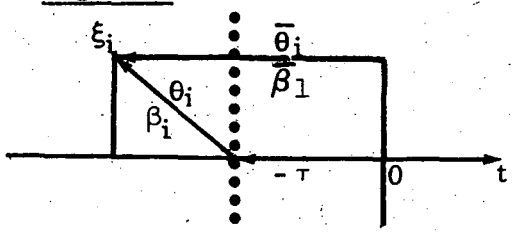
$$\xi(\bar{\theta}_2) = u(\tau, \theta_2) + \sum_i \gamma_i A^{(i)}(\tau, 0). \tag{55}$$

The coefficients γ_i are "known". In the case where $\tau = -h$, we have $\gamma_i = 0$ in our present example. Thus, the undetermined parameters $\bar{\beta}$ are defined by the equations

$$T(\bar{\beta}) = \gamma\ . \tag{56}$$

This is what we call a backward translation and we indicate it schematically as

Figure 2



i use $\sum_i \bar{\beta}_i A_i$

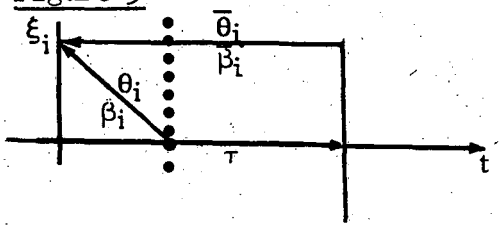
ii solve $T(\bar{\beta}) = \gamma$.

We could equally well have done the reverse process. That is, define the undetermined parameters β_i by setting them equal to γ_i

$$\beta_i = \gamma_i \tag{57}$$

and then perform a forward translation to obtain the parameters $\bar{\beta}_i = \bar{\beta}_i(\beta)$ which are functions of β_i and then use ξ_2 as usual. Again if $\tau = h$, then γ_i are all zero and $\beta_i = 0$, but the fact that β_i are zero does not mean that all $\bar{\beta}_i$ are zero. We indicate this schematically as

Figure 3



i solve $\beta_i = \gamma_i$

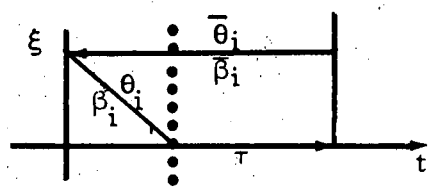
ii use $\sum_i T_i(\beta) A_i$.

In a sense we have the least number of undetermined parameters in our system if we do a forward translation in this fashion; or, we could proceed in the same manner as with the first case using $\bar{\beta}_i$, but, instead, we apply the translation to β and solve

$$\bar{\beta}_i = T_i(\beta). \tag{58}$$

We indicate this schematically as

Figure 4



i use $\sum_I \bar{\beta}_i A_i$

ii solve $\beta_i = \gamma_i$

iii solve $\bar{\beta}_i = T_i(\beta)$.

Each of these approaches has its advantages and disadvantages. In the first case, the nonlinear parameter defining equations have a form directly related to those derived in Chapters II and IV. However, the solution for $\bar{\beta}_i$ may not be obvious. In the second case, the solution for β_i is obvious, straightforward, but the parameter defining equations may look different than those derived globally. The third approach has a direct obvious solution for β_i and also for $\bar{\beta}_i$; the results for the parameter equations are the same as the first case and, thus, directly related to the work of Chapters II and IV. However, it has the disadvantage that it is, in general, necessary to translate a fully constructed vector component γ_i and this may cause the storage capacity of the machine to be exceeded. If it were not for this, the third approach might well be the best.

For any particular scheme, the following approach will be used. Whenever the constructive representation of an approximation ξ_i is not available for use in constructing the approximation ξ_j, write a representation of ξ_i in terms of undetermined parameters.

$$\xi_i = u(\delta, \theta_i) + \sum_a \beta_a^i A^{(a)}(\delta, 0). \tag{59}$$

Translate (59) to the point where all the substitutions are to be performed,

$$\xi_i = u(\zeta, \bar{\theta}_i) + \sum_a \bar{\beta}_a^i A^{(a)}(\zeta, 0) \tag{60}$$

and use this translated representation in the same manner as any other

approximation. Note that this translation may, in reality, be the identity translation analogous to Figure 3 or Figure 4, or it could be a translation corresponding to $\delta - \zeta$ which would correspond to Figure 2. To determine the undetermined parameter, either translate (59) to a place where the expansion is known or translate the known parameters to the point δ. Then solve for the parameters by equating the appropriate one of the two representations

$$\begin{aligned} T(\beta) &= \gamma \\ \beta &= T(\gamma) \end{aligned} \tag{61}$$

where T is the translation operator.

In order to carry out the translations, we see that it is necessary to be able to write

$$\begin{aligned} u(\delta, \theta) &= u(\zeta, \phi) + \sum_a \ldots A^{(a)}(\zeta, 0) \\ A(\delta, 0) &= \sum_a \ldots A^{(a)}(\zeta, 0). \end{aligned} \tag{62}$$

The translation of u is carried out by means of

Theorem 8

Let $u^{(k)}[i] = u[i + kn]$ where $i \in N$, $k \in P$. Then

$$u^{(k)}(\delta, \theta) = u^{(k)}(\zeta, \phi) + \sum_{r=\ell+1}^{\infty} \sum_{a \in S_r} \alpha_{ra}^{(k)} A^{(r,a)}(\zeta, 0) \tag{63}$$

where $A^{(r,a)}$ has order r, position a in the set $S_r = \{\text{all A of order r}\}$. The coefficients α are given as

$$\alpha_{ra}^{(k)} = \sum_{j=0}^{k+\ell} \frac{\sigma^{(k+r-j)}}{(k+r-j)!} \frac{\theta^j}{j!} \beta_{ra} \tag{64}$$

where $\phi = \delta + \theta - \zeta$, $\sigma = \delta - \zeta$, and β_{ra} are the derivative harmonics of Definition 4.

Proof: Define $u^{(k)}[i] = u[i + kn]$. Use (52) to write

$$u^{(k)}(\delta, \theta) = \sum_{r=o}^{k+\ell} \frac{\theta^r}{r!} D^{p-1-k+r}x(\delta).$$

But, we have that

$$D^{p-1-k+r}x(\delta) = \sum_{i=r}^{\infty} D^{p-1-k+i}x(\zeta) \frac{\sigma^{(i-r)}}{(i-r)!}$$

where $\sigma = \delta - \zeta$.

This result is substituted in the previous equation to obtain

$$u^{(k)}(\delta, \theta) = \sum_{r=o}^{k+\ell} \sum_{i=r}^{\infty} D^{p-1-k+i}x(\zeta) \frac{\sigma^{(i-r)}}{(i-r)!} \frac{\theta^r}{r!} . \tag{65}$$

Break the sum in (65) into two sums

$$\sum_{r=o}^{k+\ell} \sum_{i=r}^{k+\ell} + \sum_{i=k+\ell+1}^{\infty} \sum_{r=o}^{k+\ell}$$

interchange the order of summation in the first term, note that

$$\frac{(\sigma+\theta)^i}{i!} = \sum_{r=o}^{i} \frac{\sigma^{(i-r)}\theta^r}{(i-r)!\, r!}$$

and the following result is obtained

$$u^{(k)}(\delta, \theta) = u^{(k)}(\zeta, \phi) + \sum_{i=o}^{\infty} \sum_{r=o}^{k+\ell} D^{p+\ell+i}x(\zeta) \frac{\sigma^{(k+\ell+1+i-r)}}{(k+\ell+1+i-r)!} \frac{\theta^r}{r!} \tag{66}$$

where $\phi = \delta + \theta - \zeta$, $\sigma = \delta - \zeta$. We are able to expand the derivatives $D^j x(\zeta)$ into the basis $A(\zeta, 0)$ using the derivative harmonics of Definition 4. When this is done (63) and (64) will be obtained.

In order to effect the translation of the differential A, we shall define functions $\gamma \in R \to R$ called translation harmonics. The translation harmonic γ is associated with two differentials A and $\overline{A}$ which we write

as $A^{(r,a)}$, $\overline{A}^{(\overline{r},\overline{a})}$. The corresponding γ is written as $\gamma^{(\overline{r},\overline{a})}_{r,a}$. Since the definition of these harmonics is rather involved, we present first the following translation theorem.

Theorem 9

Let the differentials A be sequentially indexed as $A^{(j)}$. Let the differentials $A^{(i)}(\delta, 0)$ and $A^{(j)}(\zeta, 0)$ be given along with their associated translation harmonics γ^{i}_{j}. Then

$$A^{(i)}(\delta, 0) = \sum_{j \in S_j} \gamma^{i}_{j} (\delta - \zeta) A^{(j)}(\zeta, 0), \quad i \in S_i = S_j \tag{67}$$

where we have evaluated the translation harmonics at the point $\delta - \zeta$ corresponding to the interval of translation and S_j = {set of all differentials}.

Corollary: $$A^{(j)}(\zeta, 0) = \sum_{i \in S_i} \gamma^{j}_{i} (\zeta - \delta) A^{(i)}(\delta, 0)$$

that is
$$\sum_{i \in S_i} \gamma^{j}_{i} (\zeta - \delta) \gamma^{i}_{k} (\delta - \zeta) = \delta_{jk} \tag{68}$$

where $\delta_{jj} = 1$, $\delta_{jk} = 0$ if $j \neq k$.

Thus, the translation harmonics allows us to express $A(\delta, 0)$ in terms of $A(\zeta, 0)$, effectively a change of basis, and if we wish to obtain the inverse transformation, we need simply replace $\delta - \zeta$ by $-(\delta - \zeta)$. It is implicitly assumed that the A are independent.

In the following definition of the translation harmonics, we shall assume that we are generating two sets of differentials

$$\begin{aligned} \overline{A} = A^{(\overline{r}, \overline{a})} &= \{\overline{r}_o, \overline{k}_1, \ldots, \overline{k}_s; A^{(\overline{r}_1, \overline{a}_1)} \ldots A^{(\overline{r}_s, \overline{a}_s)}\} \\ A = A^{(r, a)} &= \{r_o, k_1, \ldots, k_s; A^{(r_1, a_1)} \ldots A^{(r_s, a_s)}\} \end{aligned} \tag{69}$$

and their corresponding translation harmonic $\gamma^{(\overline{r}, \overline{a})}_{r,a}$. Then the γ are defined in

Definition 9: Let γ be the translation harmonic corresponding to A, $\bar{A}$ of (69). Then we have that for

$$i) \quad s < \bar{s} \qquad \gamma(t) = 0 \tag{70.1}$$

$$ii) \quad s = \bar{s}$$

$$a) \quad (\bar{k}_1, \ldots, \bar{k}_s) \neq (k_1, \ldots, k_s) \qquad \gamma(t) = 0 \tag{70.2}$$

$$b) \quad (\bar{k}_1, \ldots, \bar{k}_s) = (k_1, \ldots, k_s)$$

$$1) \quad r_o < \bar{r}_o \qquad \gamma(t) = 0 \tag{70.3}$$

$$2) \quad r_o \geq \bar{r}_o$$

$$\gamma(t) = \frac{(\omega_1)! \ldots (\omega_g)!}{(\delta_1^{(1)})! \ldots (\delta_{\sigma_1}^{(1)})! \ldots (\delta_1^{(g)})! \ldots (\delta_g^{(g)})!} \cdot \frac{t^{r_o - \bar{r}_o}}{(r_o - \bar{r}_o)!} \tag{70.4}$$

$$\cdot \gamma_{r_1, a_1}^{(\bar{r}_1, \bar{a}_1)}(t) \ldots \gamma_{r_s, a_s}^{(\bar{r}_s, \bar{a}_s)}(t)$$

where $S_i = \{(r_j, a_j) \mid k_j = k_i,\ 1 \leq j \leq s = \bar{s}\} =$

$$\{(r_{i_j}, a_{i_j}) \mid j = 1, \ldots, \omega_i\}$$

$\delta_j^{(i)}$ = the repetition factor of $(r_{i_j}, a_{i_j}) \in S_i$

$$\omega_i = \sum_{j=1}^{\sigma_i} \delta_j^{(i)}$$

$$S_o = \{(r_i, a_i) \mid i = 1, \ldots, s\} = \bigcup_{i=1}^{g} S_i$$

$$iii) \quad s > \bar{s},\ s - \bar{s} = q$$

$$a) \quad (\bar{k}_1, \ldots, \bar{k}_{\bar{s}}) \not\subset (k_1, \ldots, k_s) \qquad \gamma(t) = 0 \tag{70.5}$$

$$b) \quad (\bar{k}_1, \ldots, \bar{k}_{\bar{s}}) \subset (k_1, \ldots, k_s) = (\bar{k}_1, \ldots, \bar{k}_{\bar{s}}, k_1, \ldots, k_q)$$

$$1) \quad \exists\, M_i = \Phi \qquad \gamma(t) = 0 \tag{70.6}$$

$$2) \quad \nexists\, M_i = \Phi$$

$$\gamma(t) = \frac{1}{q!} \sum_{i \in Q} \Bigg[\sum_{i_1 \in M_1} \cdots \sum_{i_q \in M_q} \tag{70.7}$$

$$\frac{\bar{r}_o!}{(\bar{r}_o - i_1)!(i_1 - i_2)! \cdots (i_q)!(k_1 + r_1 - (i_1 - i_2))! \cdots (k_q + r_q - i_q)!}$$

$$\frac{t^j}{(r_o - \bar{r}_o + i_1)!} \beta_{r_1 a_1} \cdots \beta_{r_q a_q} \gamma^{(\bar{r}_1, \bar{a}_1)}_{r_{q+1}, a_{q+1}}(t) \cdots \gamma^{(\bar{r}_{\bar{s}}, \bar{a}_{\bar{s}})}_{r_{q+\bar{s}}, a_{q+\bar{s}}}(t) \Bigg]_i$$

where $j = r_o - \bar{r}_o + \sum_{i=1}^{q} (k_i + r_i)$

$$M_1 = \{i \mid \max(0, \bar{r}_o - r_o) \leq i \leq \bar{r}_o\} \subseteq J_1 = \{0, \ldots, \bar{r}_o\}$$

$$M_2 = \{i \mid \max(0, i_1 - (k_1 + \ell)) \leq i \leq i_1\} \subseteq J_2 = \{0, \ldots, i_1\}$$

$$\vdots \tag{70.8}$$

$$M_q = \{i \mid \max(0, i_{q-1} - (k_{q-1} + \ell)) \leq i \leq \min(k_q + \ell, i_{q-1})\} \subseteq$$

$$J_q = \{0, \ldots, i_{q-1}\}.$$

The $\gamma^{(\bar{r}_i, \bar{a}_i)}_{r_i a_i}$ are translation harmonics corresponding to $A^{(\bar{r}_i, \bar{a}_i)}$ and $A^{(r_i, a_i)}$.

$Q = \{$the set of all distinct permutations of S_o that leave $(\bar{k}_1, \ldots, \bar{k}_q)$ unchanged$\}$.

$S_o = \{(\bar{k}_i = k_{q+i}, r_{q+i}, a_{q+i}) \mid i = 1, \ldots, \bar{s}\} \cup \{k_i, r_i, a_i) \mid i = 1, \ldots, q\}$

and the $\beta_{r_i a_i}$ are the derivative harmonics, corresponding to $A^{(r_i, a_i)}$, that are defined in Definition 4.

Having defined the translation harmonics γ, it is necessary to interpret the definition correctly in the special cases given below. If $s = \bar{s} = 0$, then $\gamma(t) = \frac{t^{r_o - \bar{r}_o}}{(r_o - \bar{r}_o)!}$ provided $r_o \geq \bar{r}_o$. This is case *ii*)b)2. That is, the empty multiplier is considered to be 1. In case *iii*)a), we see that the set $(\bar{k}_1, \ldots, \bar{k}_{\bar{s}})$ is not a subset of $(k_1, \ldots, k_s)$. If only

some of the $\bar{k}_i$ appear in $(k_1, \ldots, k_s)$, then we still have case *iii*)a). In case *iii*)b), we have that the set $(\bar{k}_1, \ldots, \bar{k}_{\bar{s}})$ is completely contained in $(k_1, \ldots, k_s)$. Before using (70.6), we permute the triplets (k_i, r_i, a_i) of $A^{(r,a)}$ so that we have, after a suitable re-indexing, $(k_1, \ldots, k_s) = (\bar{k}_1, \ldots, \bar{k}_{\bar{s}}, k_1, \ldots, k_q)$. We are then ready to use (70.8). It is possible that the sets $M_j = \Phi$, the null set. That is, while it is implicitly assumed that $i_j \in J_j$, it may not be true that the conditions defining M_j make sense. For example, $0 \leq i_q - (k_q + \ell) \geq \min (k_q + \ell, i_{q-1})$ may be true. In the cases where the conditions defining the M_j do not make sense, $M_j = \Phi$, the null set. If some $M_i = \Phi$, then the corresponding $\gamma(t) \equiv 0$, this is (70.5). To obtain Q, we permute the set $S_o = \{(k_i, r_i, a_i)\}$ and then ask whether the permutation is a new one and whether the set $(\bar{k}_1, \ldots, \bar{k}_{\bar{s}})$ is the same as before. In this permutation, the letters establish the position and the numbers are permutted into the positions. If the permutation is new and $(\bar{k}_1, \ldots, \bar{k}_{\bar{s}})$ is <u>unchanged</u>, then it is accepted as a valid permutation to be included in the set Q. If $\bar{s} = 0$ and $s > 0$ is the case, then we consider that case *iii*(b) holds which is consistent with the fact that $(\bar{k}_1, \ldots, \bar{k}_{\bar{s}}) = \Phi \subset (k_1, \ldots, k_s)$. There are, of course, no γ since $\bar{s} = 0$.

<u>Proof</u> of Theorem 9: The results of Theorem 8 permit us to write

$$u(\delta, t - \delta) = u(\zeta, t - \zeta) + \sum_{r=\ell+1}^{\infty} \sum_{a \in S_r} \sum_{k \in P} \alpha_{ra}^{(k)} I_{LN}^{(k)} A_N^{(r,A)}(\zeta, 0) \quad (71)$$

where $\alpha_{ra}^{(k)} = \alpha_{ra}^{(k)}(t)$ is a function of t. By hypothesis, we assume that for the lower order A we can write

$$A^{(\bar{r}_i, \bar{a}_i)}(\delta, 0) = \sum_{r_i=\ell+1}^{\infty} \sum_{a_i \in S_{r_i}} \gamma_{r_i a_i}^{(\bar{r}_i, \bar{a}_i)} A^{(r_i, a_i)}(\zeta, 0) \quad (72)$$

where the coefficient γ depend only on $\delta-\zeta$ and not on t. At the moment we do not assume that the γ are translation harmonics, although it will, of course, turn out that the definition of the translation harmonics is precisely the constructive definition of the above γ.

We now write

$$D_{L_1\cdots L_{\bar{s}}} X(u(\delta, t-\delta)) = D_{L_1\cdots L_{\bar{s}}} X(u(\zeta, t-\zeta)) + \sum_{q=1} \frac{1}{q!}$$

$$D_{L_1\cdots L_q L_1\cdots L_{\bar{s}}} X(u(\zeta, t-\zeta)) \cdot \sum_{r_1\cdots r_q} \sum_{a_1\cdots a_q} \sum_{k_1\cdots k_q} \tag{73}$$

$$\alpha_{r_1 a_1}^{(k_1)}\cdots\alpha_{r_q a_q}^{(k_q)} \; I_{L_1 N_1}^{(k_1)}\cdots I_{L_q N_q}^{(k_q)} \; A_{N_1}^{(r_1, a_1)}(\zeta, 0)\cdots A_{N_q}^{(r_q, a_q)}(\zeta, 0)$$

which has been obtained by substituting (71) into $D_{L_1\cdots L_s} X(u(\delta, t-\delta))$ and expanding into a Taylor's series about $u(\zeta, t-\zeta)$. To avoid unnecessary complexity of notation, we shall simply write the index sets of the partial derivatives of $X \circ u$ using the same L_i since the interpretation should be clear from the contect in which they appear. We now differentiate (71) and evaluate at $t = \delta$ remembering that the coefficients α are themselves functions of t. However, before carrying out the differentiation, we write (73) as

$$D_{L_1\cdots L_{\bar{s}}} X(u(\delta, t-\delta)) = \sum_{j=o}^{\infty} \frac{(t-\zeta)^j}{j!} D^j(D_{L_1\cdots L_{\bar{s}}} X(u(\zeta, 0))) +$$

$$\sum_{q=1}^{\infty} \frac{1}{q!} \sum_{j=o}^{\infty} \frac{(t-\zeta)^j}{j!} D^j(D_{L_1\cdots L_{\bar{s}} L_1\cdots L_q} X(u(\zeta, 0))) \Sigma \cdots \tag{73.1}$$

After carrying out the differentiation, we obtain

$$D^{\bar{r}_o}(D_{L_1\cdots L_{\bar{s}}} X(u(\delta, 0))) = \sum_{j=o}^{\infty} \frac{(\delta-\zeta)^j}{j!} D^{j+\bar{r}_o}(D_{L_1\cdots L_{\bar{s}}} X(u(\zeta, 0))) +$$

$$\sum_{q=1}^{\infty} \sum_{j=o}^{\infty} \sum_{r_1\cdots r_q} \sum_{a_1\cdots a_q} \sum_{k_1\cdots k_q} \sum_{i_1=o}^{\bar{r}_o} \sum_{i_2=o}^{i_1} \cdots \sum_{i_q=o}^{i_{q-1}} \frac{1}{q!} \frac{(\delta-\zeta)^{j-\bar{r}_o+i_1}}{(j-\bar{r}_o+i_1)!}$$

(continued)

$$\times\ \alpha_{\overline{r}_o i_1 \ldots i_q} D^{j}(D_{L_1}\ldots L_{\overline{s}} L_1 \ldots L_q\ X(u(\zeta,0)))\ D^{i_1-i_2}(\alpha^{(k_1)}_{r_1 a_1})\ldots$$

$$D^{i_q}(\alpha^{(k_q)}_{r_q a_q})\ I^{(k_1)}_{L_1 N_1}\ldots I^{(k_q)}_{L_q N_q}\ A^{(r_1,a_1)}_{N_1}\ldots A^{(r_q,a_q)}_{N_q}. \tag{74}$$

The coefficient α are given in Theorem 8 as

$$\alpha^{(k)}_{ra}(t) = \sum_{j=0}^{k+\ell} \frac{(\delta-\zeta)^{k+r-j}}{(k+r-j)!}\ \frac{(t-\delta)^{j}}{j!}\ \beta_{ra} \tag{75}$$

where the β_{ra} are derivative harmonics. We see from (75) that the $\alpha^{(k)}_{ra}$ are polynomials of order $k+\ell$ and, thus, their higher order derivatives are identically zero. We, thus, limit the index ranges to

$$\begin{aligned} 0 &\leq i_1 - i_2 \leq k_1 + \ell \\ 0 &\leq i_2 - i_3 \leq k_2 + \ell \\ &\vdots \\ 0 &\leq i_q \leq k_q + \ell \end{aligned} \tag{76}$$

since for values outside these ranges the derivatives are zero. Differentiate (75) and evaluate at $t = \delta$ to obtain

$$D^{j}(\alpha^{(k)}_{ra})(\delta) = \frac{(\delta-\zeta)^{k+r-j}}{(k+r-j)!}\beta_{ra} \qquad 0 \leq j \leq k+\ell. \tag{77}$$

We now write

$$\overline{A}(\delta,0) = D^{\overline{r}_o}(D_{L_1}\ldots L_{\overline{s}}\ X(u(\tau,0)))\ A^{(\overline{r}_1,\overline{a}_1)}(\delta,9)\ \ldots\ A^{(\overline{r}_{\overline{s}},\overline{a}_{\overline{s}})}(\ ,0)$$

$$I^{(\overline{k}_1)}_{L_1 N_1}\ldots I^{(\overline{k}_{\overline{s}})}_{L_{\overline{s}} N_{\overline{s}}}. \tag{78}$$

Substitute (77) into (74), then substitute that results along with (72) into (78) to obtain the desired expanions.

$$\overline{A}(\delta, 0) = \sum_{j=0}^{\infty} \sum_{r_1 \cdots r_{\overline{s}}} \sum_{a_1 \cdots a_{\overline{s}}} \{\overline{r}_o + j, \overline{k}_1, \ldots, \overline{k}_{\overline{s}}; A^{(r_1,a_1)}(\zeta, 0) \ldots$$

$$\cdot A^{(r_{\overline{s}},a_{\overline{s}})}(\zeta, 0) \cdot \frac{(\delta-\zeta)^j}{j!} \gamma^{(\overline{r}_1,\overline{a}_1)}_{r_{\overline{s}}a_{\overline{s}}} \ldots \gamma^{(\overline{r}_{\overline{s}},\overline{a}_{\overline{s}})}_{r_{\overline{s}}a_{\overline{s}}} + \sum_{r_o=0}^{\infty} \sum_{q=1}^{\infty} \sum_{i_1 \in J_1} \cdots \sum_{i_q \in J_q}$$

$$\cdot \sum_{r_1 \cdots r_{\overline{s}}} \sum_{a_1 \cdots a_{\overline{s}}} \sum_{r_{\overline{s}+1} \cdots r_s} \sum_{a_{\overline{s}+1} \cdots a_s} \sum_{k_{\overline{s}+1} \cdots k_s} \tag{79}$$

$$\cdot \{r_o, \overline{k}_1, \ldots, \overline{k}_{\overline{s}}, k_{\overline{s}+1}, \ldots, k_s; A^{(r_1,a_1)}(\zeta, 0) \ldots A^{(r_{\overline{s}},a_{\overline{s}})}(\zeta, 0)$$

$$\cdot A^{(r_{\overline{s}+1},a_{\overline{s}+1})}(\zeta, 0) \ldots A^{(r_s,a_s)}(\zeta, 0)\} \cdot \frac{1}{q!} \frac{(\delta-\zeta)^j}{j!} \alpha_{\overline{r}_o i_1 \ldots i_s}$$

$$\cdot \frac{(\delta-\zeta)^{(k_{\overline{s}+1}+r_{\overline{s}+1}-(i_1-i_2))}}{(k_{\overline{s}+1}+r_{\overline{s}+1}-(i_1-i_2))!} \cdots \frac{(\delta-\zeta)^{(k_s+r_s-i_q)}}{(k_s+r_s-i_q)!}$$

$$\cdot \gamma^{(\overline{r}_1,\overline{a}_1)}_{r_1 a_1} \ldots \gamma^{(\overline{r}_{\overline{s}},\overline{a}_{\overline{s}})}_{r_{\overline{s}}a_{\overline{s}}} \beta_{r_{\overline{s}+1}a_{\overline{s}+1}} \cdots \beta_{r_s a_s}$$

where $s = \overline{s} + q$

$$j = r_o - \overline{r}_o + i_1$$

in the second term of the right number of (79). We write

$$A^{(\overline{r},\overline{a})}(\delta, 0) = \sum_{r=\ell+1}^{\infty} \sum_{a \in S_r} \gamma^{(\overline{r},\overline{a})}_{r,a} A^{(r,a)}(\zeta, 0)$$

and let the general term on the right side of this relation be represented by

$$A^{(r,a)} = \{R_o, K_1, \ldots, k_s; A^{(b_1,c_1)} \ldots A^{(b_s,c_s)}\}. \tag{80}$$

We wish to find this term and its coefficient on the right side of (79). An examination of (79) shows us that $\gamma^{(\overline{r},\overline{a})}_{r,a} = 0$ if $s < \overline{s}$, since there are no terms in (79) with $s < \overline{s}$ and hence an identification of the γ with the

translation harmonics is correct in this case. For $s = \bar{s}$, we need only consider the first term of (79). Let $r_o = R_o = \bar{r}_o + j$. We see that if $r_o < \bar{r}_o$ then again $\gamma = 0$ and the translation harmonics are correct. For $j \geq 0$, we will, indeed, have the translation harmonics provided the coefficient is correct. We have that

$$\left.\begin{aligned} R &= r_o = \bar{r}_o + j \\ K_i &= k_i = \bar{k}_i \\ b_i &= r_i \\ c_i &= a_i \end{aligned}\right\} \quad i = 1, \ldots, s = \bar{s} . \tag{81}$$

We can find the valid index sets of (79) by considering permutations of (81) that leave the differential unchanged. In general, $\bar{k}_i$ are fixed by the definition of $\bar{A}(\delta, 0)$, that is by (78). However, if $\bar{k}_i = \bar{k}_j$ then we can permute the couples (b_i, c_i) and (b_j, c_j) to arrive at another valid index set through the use of (81). Thus, we proceed as before in Theorem 5. Collect into sets S_i the couples (r_i, a_i) of like k_i. The couple $(r_j, a_j) \in S_i$ if, and only if, $k_j = k_i$. Let

$$S_i = \{(r_{i_1}, a_{i_1})^{\delta_i^{(i)}}, \ldots, (r_{i_{\sigma_i}}, a_{i_{\sigma_i}})^{\delta_{\sigma_i}^{(i)}}\}$$

where $\delta_j^{(i)}$ is the repetition factor indicating the number of times (r_{i_j}, a_{i_j}) appears in S_i. We have the $\omega_i = \sum_{j=1}^{\sigma_i} \delta_j^{(i)}$ is the total number of terms in S_i. Define $S_o = \bigcup_{i=1}^{g} S_i$. Again, let Q be the set of all distinct permutations of S_o where by a permutation we mean that for each S_i we permute its elements and then take the union of the permuted sets. The general coefficient

$$\frac{(\delta-\zeta)^j}{j!} \gamma_{r_1 a_1}^{(\bar{r}_1, \bar{a}_1)} \cdots \gamma_{r_s a_s}^{(\bar{r}_{\bar{s}}, \bar{a}_{\bar{s}})}, \quad j = r_o - \bar{r}_o$$

appears in all of these permutations and may be factored out. The coefficient left is the number of such permutations that lead to distinct permutation sets and this is

$$\frac{(\omega_1)!\ldots(\omega_g)!}{(\delta_1^{(1)})!\ldots(\delta_\sigma^{(1)})!\ldots(\delta_1^{(g)})!\ldots(\delta_{\sigma_g}^{(g)})!}$$

and again we see that if $\gamma_{r_i a_i}^{(\bar{r}_i,\bar{a}_i)}$ are defined as the translation coefficients $\gamma(\delta - \zeta)$ corresponding to $A^{(\bar{r}_i,\bar{a}_i)}$, $A^{(r_i,a_i)}$, then we have $\gamma_{r,a}^{(\bar{r},\bar{a})}$ is the translation coefficient $\gamma(\delta - \zeta)$ corresponding to $A^{(\bar{r},\bar{a})}$, $A^{(r,a)}$.

For terms of degree $s > \bar{s}$, we proceed as follows. We have that

$$\begin{aligned} R &= r_o \\ K_i &= \bar{k}_i = k_i \quad , i = 1, \ldots, \bar{s} \\ K_i &= k_i \quad , i = \bar{s} + 1, \ldots, \bar{s} + q = s \\ b_i &= r_i \\ c_i &= a_i \end{aligned} \quad \Bigg\} \quad , i = 1, \ldots, \bar{s} + q = s. \tag{82}$$

The indexing above is slightly different than that used in the definition of the translation harmonics; however, this is immaterial since we can suitably re-index the items provided that we are consistent and keep the triplets (k_i, r_i, a_i) with the same common index.

Equation (82) establishes one valid index set. To obtain all other valid sets, let S_o be defined as

$$S_o = \{(\bar{k}_1, r_1, a_1), \ldots, (\bar{k}_{\bar{s}}, r_{\bar{s}}, a_{\bar{s}}), (k_{\bar{s}+1}, r_{\bar{s}+1}, a_{\bar{s}+1}), \ldots (k_s, r_s, a_s)\}$$

with $s = \bar{s} + q$. Let Q be the set of all permutations of S_o that are distinct and leave the set $(\bar{k}_1, \ldots, \bar{k}_s)$ unchanged. It is important to

understand that we really mean that the set is identically the same before and after the permutation. Consider (78) when $\bar{A}$ is defined and also (79). We see that $\bar{k}_1, \ldots, \bar{k}_{\bar{s}}$ do not enter into any sums and, therefore, are fixed. From (79), we notice that we must be careful to keep the i indices within their bounded range indicated in (76). In general, we have that $0 \leq i_1 \leq r_o$, $0 \leq i_2 \leq i_1, \ldots, 0 \leq i_q \leq i_{q-1}$ and for any choice of k_i the indices will take on all values. The problem is that some of these values correspond to terms that are identically zero. One way to overcome this difficulty is to apply the following bounds to the indices. Define the sets M_i as in (70.8). That is, the sets are defined in the same way as they are used in Definition 9, but because of our indexing we interpret k_i as $k_{\bar{s}+1}$. Now if M_i is a set for which the bounds are unattainable; that is, the bounds are contradictory; then $M_i = \Phi$, the null set. We consider $\Phi \subset J_i$ and then we make the convention that if $i \in J_i - M_i$ then the term we are looking for on the right-hand side has a zero coefficient. This is, of course, true since for i outside these ranges the derivatives of α are zero.

In general, we do not have an identical coefficient for each of the permutations, so we do not need to establish the multinomial coefficient for the set Q. We write the general coefficient for this case as

$$\frac{1}{q!} \sum_{i \in Q} \left[\frac{\bar{r}_o}{(\bar{r}_o - i_1)!\ (i_1 - i_2)! \cdots (i_q)!\ (k_{\bar{s}+1} + r_{\bar{s}+1} - (i_1 - i_2))! \cdots (k_s + r_s - i_q)!} \right.$$
$$\left. \cdot \frac{(\delta - \zeta)^j}{(r_o - \bar{r}_o + i_1)!}\ \gamma_{r_1 a_1}^{(\bar{r}_1, \bar{a}_1)} \cdots \gamma_{r_{\bar{s}} a_{\bar{s}}}^{(\bar{r}_{\bar{s}}, \bar{a}_{\bar{s}})} \beta_{r_{\bar{s}+1} a_{\bar{s}+1}} \cdots \beta_{r_s a_s} \right]_i \tag{83}$$

where $j = r_o - \bar{r}_o + \sum_{i=\bar{s}+1}^{s} (k_i + r_i)$, $s = \bar{s} + q$ and again we see that $\gamma_{r,a}^{(\bar{r},\bar{a})} = \gamma(\delta - \zeta)$ where γ is the translation harmonic corresponding to

$A^{(\bar{r},\bar{a})}$, $A^{(r,a)}$ provided we make the analogous interpretation of $\gamma^{(\bar{r}_i,\bar{a}_i)}_{r_i a_i}$.

Thus, given $A^{(\bar{r},\bar{a})}(\delta,0)$ we are able to write this in terms of the differentials $A^{(r,a)}(\zeta, 0)$ by means of the $\gamma^{(\bar{r},\bar{a})}_{r,a}(\delta - \zeta)$ evaluated at the interval of translation provided we know the expansion of the factors $A^{(\bar{r}_i,\bar{a}_i)}(\delta, 0)$ that appear in $A^{(\bar{r},\bar{a})}$. These factors are all of lower order $\bar{r}_i$, so an induction on order $\bar{r}_i$ can be applied and we need only show that we can get started. Since the start is with order $\bar{r} = \ell + 1$ and degree $\bar{s} = 0$, we have actually to do an induction on order and degree. However, a consideration of how the differentials are constructed will show that if we can write the results for $\bar{r} = \ell + 1$, $\bar{s} = 0$, then we can obtain all the terms since we will, at each step, have already obtained the necessary lower order expansions. The lowest order $\bar{r} = \ell + 1$, $\bar{s} = 0$ term is $D^{\bar{r}-1}(X \circ u)$ where $\bar{r} = \ell + 1$. This term is obtained by setting $\bar{s} = 0$ in (74), or equally in (79). We then check that Definition 9 equations (70.4) and (70.7) do give the right coefficients when evaluated at $t = \delta - \zeta$, $\bar{s} = 0$. This then establishes Theorem 9 and although the proof is somewhat involved, the generation of the γ should be a straightforward operation that can be systematically carried out.

The corollary to Theorem 9 follows quite easily since we never said what δ and ζ were to be. We give the necessary steps below. We can write

$$A^{(j)}(\delta, 0) = \sum_{i \in S_i} \sum_{k \in S_k} \gamma^j_i(\delta - \zeta)\, \gamma^i_k(\zeta - \delta)\, A^{(k)}(\delta, 0)$$

$$A^{(j)}(\zeta, 0) = \sum_{i \in S_i} \sum_{k \in S_k} \gamma^j_i(\zeta - \delta)\, \gamma^i_k(\delta - \zeta)\, A^{(k)}(\zeta, 0)$$

where $S_i = S_k = \{$set of <u>all</u> differentials A$\}$. If the A are independent, then it follows that

$$\sum_{i \in S_i} \gamma_i^j(t)\, \gamma_k^i(t) = \delta_{jk},\ j,\ k \in S_i \tag{84}$$

where δ_{jk} is the Kroneker delta. It is interesting to note that the γ are independent of the function x or X ∘ x that define the differentials A and, hence, of the set S_i. The results of the corollary must be true no matter how many elements we have in the set S_i. This could prove useful in checking the γ.

As a further aid in checking these harmonics, we note that they have the following.

Property 1: Let γ be the translation harmonic of Definition 9 corresponding to the differentials $\overline{A}$, A of order $\overline{r}$ and r, respectively. Then

$$\gamma(t) = c\, t^{(r-\overline{r})} \tag{85}$$

where c is a constant independent of t.

That this must, indeed, be the case can easily be seen by considering the physical units that might be associated with A and t in a particular case; say, for example, when t is a time variable. That it does follow from Definition 9 can be shown by an inductive proof on the order r. The statement is trivially true for all γ that are identically zero. It is also true for the case of lowest order and lowest degree $\gamma_r^{(\ell+1)}$, since for these functions $\overline{s} = 0$ in (70.3) and (70.7). In the case of (70.3), write $t^{r_o-\overline{r}_o} = t^{1+r_o-(\overline{r}_o+1)} = t^{r-\overline{r}}$ and in the case of (70.7), write $j = r_o - \overline{r}_o + \sum_{i=1}^{q} k_i + r_i = r_o + 1 + \sum_{i=1}^{q} k_i + r_i - (\overline{r}_o + 1) = r - \overline{r}$. Thus, the induction can be started. Referring to the definition of the γ, we see that, as was true for the differentials A, any γ of orders r, $\overline{r}$ is formed from γ of orders r_i, $\overline{r}_i$ where $r_i < r$, $\overline{r}_i < \overline{r}$. Therefore, we assume that the results hold for these lower orders and write for (70.4)

$$\gamma(t) = c\, t^{r_1-\bar{r}_1}\cdots t^{r_s-\bar{r}_s}\, t^{r_o-\bar{r}_o} = c\, t^{j}$$

where $j = \sum_{i=1}^{s}(r_i + k_i) + r_o + 1 - (\sum_{i=1}^{\bar{s}} \bar{r}_i + \bar{k}_i + \bar{r}_o + 1) = r - \bar{r}$

since $s = \bar{s}$ and $k_i = \bar{k}_i$. For (70.7), we write

$$\gamma(t) = \sum_{i\in Q} \{c_i\, t^{r_{q+1}-\bar{r}_1}\cdots t^{r_{q+s}-r_s}\, t^{(r_o-\bar{r}_o+\sum_{i=1}^{q} k_i+r_i)}\}$$

$$\gamma(t) = \sum_{i\in Q} \{c_i\, t^{j_i}\}$$

where $j_i = \sum_{i=q+1}^{q+\bar{s}} r_i + \sum_{i=1}^{q}(r_i + k_i) + \sum_{i=q+1}^{q+\bar{s}} k_i + r_o + 1$

$$- \{\sum_{i=1}^{s} (\bar{k}_i + \bar{r}_i) + \bar{r}_o + 1\} = r - \bar{r}$$

since $\bar{k}_i = k_{q+i}$. Noting that j_i is independent of the permutations, we can factor t out and obtain the desired result that $\gamma(t) = c\, t^{r-\bar{r}}$ where c is independent of t.

We can also prove inductively in the same fashion

Property 2: Let γ be the translation harmonic corresponding to $\bar{A}$, A of order $\bar{r}$, r, respectively. If $r < \bar{r}$, then $\gamma(t) = 0$. Thus, if we think of $\bar{A}$ as forming the rows and A the columns of the matrix of harmonics γ, this matrix is lower triangular.

Since the derivation of these harmonics is rather involved, there is the possibility of propagating mistakes. These harmonics can be checked to any order by performing a translation on an approximation $\xi_i = \sum_a \bar{\beta}_a\, \bar{A}^{(a)}$ to obtain $\sum_a \beta_a\, A^{(a)}$ when the $\bar{\beta}$ are treated as undetermined parameters. Then we consider $\xi_i = \xi(\theta_i)$ for which the expansion is known in terms of Taylor coefficients and derivative harmonics. The relation $\beta_a = T(\bar{\beta}_a)$ should hold when the appropriate coefficients replace the β_a and $\bar{\beta}_a$. Since the translation represents a change of basis for the A and $u(\theta_i)$, it is sufficient to check only this special case.

The translation harmonics have been tabulated for arbitrary ℓ in Table VII of Appendix I. This table is then used to obtain the various particular results needed to treat the various examples.

It is now possible, by combining the results of Theorems 8 and 9, to obtain the required representation of any ξ_j in our scheme. To actually carry out the required work, it is necessary to specify the expansion origins and translation intervals that are used. The following approach has been taken. The major intervals are considered to be half open intervals with uniform length $h_i = h$. Thus, the first major interval contains the points $t_o, t_1, \ldots, t_{q-1}$; the second major interval, the points $t_q, t_{q+1}, \ldots, t_{2q-1}$; and from the last major interval, we use only the point t_{exq}. We shall consider θ_i to be the distance of the point t_i from the origin. The actual location of this origin is immaterial, although it is possibly advantageous when solving the parameter defining equations to choose it at $\theta_q = 0$. To each major interval, we associate a local origin which is located at $-\alpha_j h$ for the major interval j. The coefficient α_j is a non-negative integer with $\alpha_o = 0$. In effect, if $\theta_o = 0$, then the local origins are $\theta_o, \theta_q, \ldots, \theta_{exq}$. Given any approximation ξ_j, we have two representations for it; one with respect to the origin and one with respect to the local origin. These are respectively written as

$$\begin{aligned} \xi_j &= u(0, \theta_j) + \sum_{r=\ell+1}^{\infty} \sum_{a \in S_r} B_{LN}^{(j,r,a)} A_N^{(r,a)}(0, 0) \\ \xi_j &= u(-\alpha_j h, \bar{\theta}_j) + \sum_{r=\ell+1}^{\infty} \sum_{a \in S_r} \bar{B}_{LN}^{(j,r,a)} A_N^{(r,a)}(-\alpha_j h, 0) \end{aligned} \tag{86}$$

where $\bar{\theta}_j = \theta_j + \alpha_j h$, with $h \geq 0$, and $0 \leq \alpha_j \leq e$, is the distance from the local origin of the point t_j corresponding to ξ_j and

$$\bar{B}^{(j,r,a)}_{LN} = \sum_{k\in P} \bar{b}_{jrak} I^{(k)}_{LN}$$

$$B^{(j,r,a)}_{LN} = \sum_{k\in P} b_{jrak} I^{(k)}_{LN} \qquad (87)$$

are the coefficients of the expansion. Since these are simply two different representations of the same approximation, the coefficients (87) are related through the use of Theorems 8 and 9. If we have done a forward translation, then the $\bar{B}$ are the undetermined parameters, while B are functions of $\bar{B}$ and the translation interval. Whereas, for a backward translation, the situation is reversed. Just how these translations are selected and carried out will be described in detail in the description of the ALGOL procedure that carry out the substitutions. We have chosen to use the backward and forward translations of Figure 2 and 3.

The defining equations for these parameters are determined in a straightforward fashion. If there is a point at which the expansion is "known", then we carry out one of the previously mentioned translations to obtain equations that define these unknown parameters; otherwise, we assume that for any approximation that has undetermined parameters there exists either a constructed representation of that approximation, or an approximation that is assumed to be the same as the one which has only a representation in terms of undetermined parameters. That is, if we use ξ_j in constructing ξ_i and if ξ_j is not yet constructed (implicit methods - closed methods), then we generate such an expansion and when ξ_j is finally constructed, we then have a constructive representation; or if ξ_j is from memory, then since the scheme is periodic there will be found among the ξ_i that we are constructing a similar ξ_i and since ξ_i and ξ_j are identically constructed approximations, they must have identical expansions about their local origins and we can consider the constructed ξ_i to be the constructive

representation of ξ_j. Hence, for every ξ_j for which there exists an expansion into undetermined parameters and for which we do not "know" the harmonics, there also exists a constructive representation. When considered to be expanded about their local origins, these expansions must be identical and we thus obtain the defining equations for these parameters by equating the harmonics of these expansions. In general, we see that since all our construction is done with reference to the origin, we might, in principle, have to translate both the constructed representation and that with respect to undetermined parameters, called here an implicit representation, to their respective local origins. In practice, however, unless the period is greater than one, most of the construction takes place in the first major interval and, thus, the constructed representations are already with respect to their local origin θ_o.

Using the above results, we now indicate how we shall carry out a constructive approach to developing schemes and their associated parameter defining equations.

Define the scheme interval parameters consisting of the major interval, period, rank q, extent e. Define a scheme constructively as

$$\xi_i = \sum_{j_1} \ldots \xi_{j_1} + \sum_{j_2} \ldots X(\xi_{j_2}) \quad , \quad i \in S_i,\ j_1 \in S_{j_1},\ j_2 \in S_{j_2} \tag{88}$$

where $S_i \cup S_{j_1} \cup S_{j_2} \subset S = \{j \mid 0 \leq j \leq exq\}$.

Consider the i to be sequentially ordered in S_i and proceed to carry out the operations of the construction using the representation

$$\xi_j = u(0, \theta_j) + \sum_r \sum_a \ldots A^{(r,a)}(0, 0)$$

to obtain the construction

$$\xi_i = u(0, \theta_i) + \sum_r \sum_a \Gamma_{LN}^{(i,r,a)} \; A^{(r,a)}(0, 0)$$

where it has been assumed that any ξ_j without a constructive representation has been expanded using undetermined parameters and that an appropriate translation has been effected so that there exists the representations with respect to the local origin and with respect to the origin. This is carried out using the substitution theorem and where necessary the translation theorem. Continue the construction until the exhaustion of the set S_i at which time choose a particular ξ, say ξ_o, of that set to be equal to the corresponding $\xi(t)$. Equate the harmonics which are obtained when these two quantities are expanded about their local origin, θ_o, to obtain the nonlinear parameter defining equations

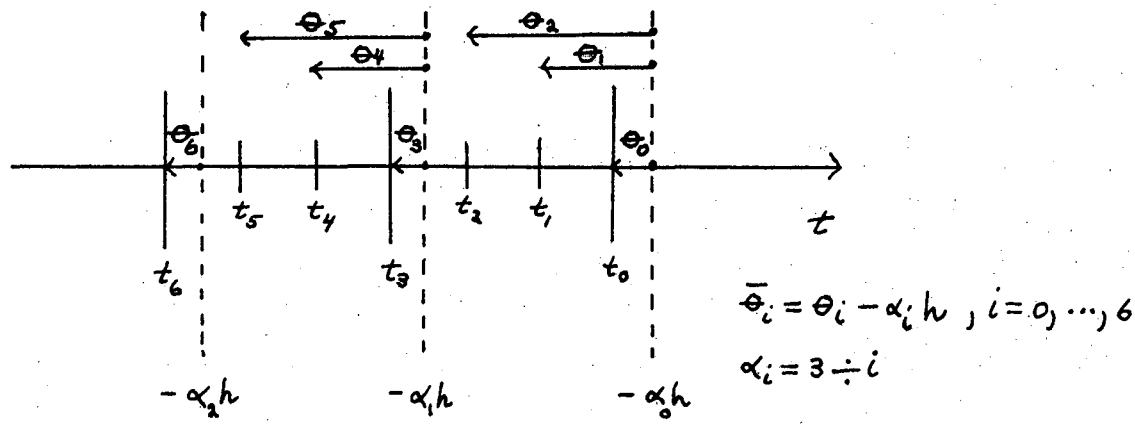
$$\Gamma_{LN}^{(i_o,r,a)} = \sum_{k \in P} \beta_{ra} \frac{\theta_j^{r+l+k}}{(r+l+k)!} \; I_{LN}^{(k)} \tag{89}$$

Finally, for each ξ_j, $j \in S_j$ that has an implicit representation and is not a major point, equate the harmonics of the implicit representation expanded about its local origin to the harmonics of the constructed representation expanded about its local origin.

$$\overline{B}_{LN}^{(j,r,a)} = \Gamma_{LN}^{(j,r,a)}. \tag{90}$$

Remember major points have a known expansion and we carry out one of the previously mentioned translations to obtain the appropriate equations. Solve (89) and (90) to obtain the parameters of the scheme as defined by (88). Below in Figure 5, we illustrate schematically the layout for the case q = 3, e = 2, period = 1 where we have taken the origin to the right of t_o.

Figure 5



Since the period is 1, we have that ξ_4 is constructed in the same manner as ξ_1 and that the expansion of ξ_4 about $-\alpha_1 h$ is the same as the expansion of ξ_1 about $\alpha_0 h$, $\alpha_0 = 0$. Similarly, for the point ξ_5 and ξ_2.

The various quantities needed to carry out the construction have been presented in Appendix I where we have tabulated the differentials, the results of doing one substitution, and of doing one multiplication along with the translation of one major interval. The translation has been normalized to one interval since be repeating the translation, any number of intervals can be obtained and in this fashion the table does not change for each interval. The other alternative is to use one arbitrary interval and let the correct number be inserted when the translation is carried out.

At this point, we shall turn our attention to the generalization of the scheme to include the matrix of the derivatives of X, the matrix whose determinant is the Jacobian and which we shall refer to as the Jacobian matrix J_{NL}. Theorem 7, the multiplication theorem, allows us to use this quantity, but we have given no scheme definition indicating how it is used.

It is possible to introduce another free parameter by evaluating $D_L\, X(\xi_j)$ where θ_j is to be included in our set of free parameters. We are actually free to use any ξ from our set of approximations. Whether the inclusion of the new term will be of any practical value must be determined by examining various schemes that make use of $D_L\, X(\xi_j)$. However, to do this, we need a way of characterizing the schemes. For that purpose, we shall introduce some more terminology since schemes that use J_{NL} are able to generate many more items to be used in the construction of approximations than the previously considered schemes.

We define generalized Runge-Kutta-Frey type integration methods using

Definition 10: The approximation ξ_i is said to be obtained by means of a generalized Runge-Kutta-Frey type integration method (RKF), if and only if

$$\xi_i - u(\theta_i) = \sum_{j_1 \in S_1} A_{LL_1}^{(i,j_1)} [\xi_{j_1} - u(\theta_{j_1})]_{L_1} + \sum_{j_2 \in S_2} B_{LN}^{(i,j_2)}\, \eta_N^{(j_2)} \tag{91}$$

where $i \in S_o$. The approximators η_N are defined to be either

$$\eta_N^{(i)} = X(\xi_i) - R_i \,, \quad i \in S_2$$

or as $$\eta_N^{(i)} = J_{NL}^{(j)}\, S_L^{(i)} \,, \quad j \in S_3 \tag{92}$$

with $$J_{NL}^{(i)} = D_L\, X(\xi_i) \,, \quad i \in S_3 \tag{93}$$

and the sums S are defined to be

$$S_L^{(i)} = \sum_{j_4 \in S_4} E_{LL_1}^{(i,j_4)} [\xi_{j_4} - u(\theta_4)]_{L_1} + \sum_{j_5 \in S_5} F_{LN}^{(i,j_5)}\, \eta_N^{(j_5)} \tag{94}$$

constructed using approximations ξ and η. The index sets are such that $\bigcup_{i=1}^{5} S_i \subset S = \{0, 1, \ldots, exq\}$ where e is the extent and q the rank

of the scheme.

In the foregoing definition, it should be kept in mind that there are actually two different spaces that are of interest; the vector space R^n and $R^{n\times p}$ and the coefficient spaces that contain the harmonics of the expansion with respect to the differentials. In describing the scheme and in its actual use, it is the vector spaces that are of interest, while in the actual construction of the parameter defining equations, it is the coefficient space that is used. Definition 10 is, essentially, given in the coefficient space since $\xi_i - u(\theta_i)$ and $X(\theta_i) - R_i$ have expansions $\sum_a \ldots A_N^{(a)}$. Thus, we shall, when constructing the scheme, interpret the definition as being in the coefficient space and use the results of the translation, substitution, and multiplication theorem which are given in that space. We can multiply the matrix J_{NL} times any element of $R^{n\times p}$; this operation is, however, to be carried out in the coefficient space and, therefore, we restrict ourselves to previously constructed approximations or approximators. In the case $p = 1$ of a system of first order differential equations, it does not really make sense to take the sums S as linear combination and then to again form linear combinations after multiplying by J_{NL}, since the coefficient factors out, one linear combination will suffice; however, if $p > 1$, then the parameters do not factor out and it does make sense to take both linear combinations to introduce more parameters. Whether this is helpful in achieving higher order methods must be found out by constructing the scheme and examining the equations.

In describing and using the scheme, we wish to work in the vector spaces R^n and $R^{n\times p}$. It is implicitly assumed here that the conditions (11') hold and, thus, we can replace $\xi_i - u(\theta_i)$ by ξ_i and $X(\xi_i) - R_i$ by $X(\xi_i)$ in Definition 10. The fact that some of the approximators η_i are

obtained by substitution and some by multiplication means that we must keep track of those that are obtained by substitution since their parameters enter into the conditions (11').

To complete the work of generating a scheme by means of successive substitutions, it is necessary to give explicitly the coefficient matrices A, B, E, F and to display explicitly the conditions of (11'). After doing this, we shall summarize our results so that they may be more conveniently applied. Before doing this, we comment here that in any systematic search for new methods using schemes such as are defined by Definition 10, it would be necessary to create a more complete set of scheme defining parameters. Some of these parameters, the interval scheme parameters, have been given here; but there will be needed some measure of the work we will have to do versus the accuracy (or some other criteria) that is achieved. For schemes that do not use the matrix J_{NL}, this is handled in a fairly satisfactory manner, but the introduction of the use of J_{NL} gives such a wealth of possible combinations that some thought should be given to a suitable classification. We give here only a very general scheme definition; a definition which enables us to construct schemes but which does not contain in itself an effective strategy for its use.

The definition of the parameter matrices has, up until now, been left open. We have simply mentioned that the choice should be sensible. It is assumed that the schemes are independent of the vectors; that is, independent of n where n is the dimension of R^n, the space in which X, A_N, and the various derivatives lie. However, they can depend on the order of the differential equation, that is on $k \in P = \{0, \ldots, p - 1\}$. Keeping in mind these simple requirements we wish to introduce, or at least

to have the ability to introduce, at each step as many new parameters as is possible. We, thus, define the B and F matrices of Definition 10 to be of the form

$$B_{LN}^{(i,j)} = \sum_{k\in P} b_{ij}^{(k)} \; I_{LN}^{(k)} \tag{95}$$

where $I_{LN}^{(k)}$ is defined as usual and $b_{ij}^{(k)} \in R^1$ is a scalar, a parameter of the scheme. The coefficients of ξ_i are defined slightly differently. We might immediately note that $\xi_i^{(k)} \sim D^{p-1-k}x(\theta_i)$ and that, therefore, it would not necessarily be the "sensible" thing to do if we used a linear combination of lower order derivatives when approximating a higher order derivative. For example, one does not usually write

$$\xi_i^{(1)} = A\xi_i^{(3)} + B\xi_i^{(2)} + C\xi_i^{(1)} + D\xi_i^{(0)}$$

unless A = B = 0. We will, however, not require, for the present, any such set of conditions since it is easier to work with square matrices and then later to impose the condition that some coefficients are zero for one reason or another. We say this in this fashion because it turns out that the condition established by (11') will actually serve to help establish this "sensible" choice. The A and E matrices are defined to be of the form

$$A_{L_1L_2}^{(i,j)} = \sum_{k_1\in P} \sum_{k_2\in P} a_{ik_2j}^{(k_1)} \; I_{L_1N}^{(k_1)} \; I_{L_2N}^{(k_2)} \tag{96}$$

where $I_{LN}^{(k)}$ have their usual meaning and $a_{ik_2j}^{(k_1)} \in R^1$ is a scalar parameter of the scheme. As usual, we sum over the repeated index sets N. Using the notation $\xi^{(k)}[i] = \xi[i + kn]$, the approximation ξ_i is written explicitly as

$$\xi_i^{(k)} - u^{(k)}(\theta_i) = \sum_{j_1 \in S_1} \sum_{k_1 \in P} a_{ik_1j_1}^{(k)} [\xi_{j_1}^{(k_1)} - u^{(k_1)}(\theta_{j_1})]$$

$$+ \sum_{j_2 \in S_2} b_{ij_2}^{(k)} \eta_N^{(j_2)} \tag{97}$$

or if one wishes to think in terms of the vector space representation, replace $\xi_i^{(k)} - u(\theta_i)$ by $\xi_i^{(k)}$ in the above equation.

Knowing the specific representation of the parameter matrices, it is now possible to develop the conditions that we intend to impose on these matrices. These allow us to represent ξ_i as either $\xi_i - u(\theta_i) = \sum_a \ldots A^{(a)}$ or use $\xi_i = \Sigma \ldots \xi_j + \ldots$ by means of (11'). The results are presented as

Theorem 10

In order that it be possible to represent the approximation ξ_i in the two forms

$$\xi_i = \sum_{j_1} \ldots \xi_{j_1} + \sum_{j_2} \ldots X(\xi_{j_2}) + \Sigma \ldots D_L\, X(\xi_j)\, S$$

$$\xi_i = u(\theta_i) + \sum_a \ldots A^{(a)} \tag{98}$$

thus allowing the construction and utilization of generalized RKF schemes, it is necessary and sufficient that the elements of the parameter matrix A multiplying ξ_{j_1} and the elements of the parameter matrix B multiplying $X(\xi_{j_2})$ satisfy the following relations

$$\sum_{j_1 \in S_1} \sum_{r=o}^{t} a_{i,(r-t+p-1),j_1}^{(k)} \frac{\theta_{j_1}^r}{r!} = 0 \tag{99}$$

$$t = 0, 1, \ldots, p - 2 - k$$

$$k = 0, \ldots, p - 1.$$

$$\sum_{j_1 \in S_1} \sum_{r=o}^{t} a^{(k)}_{i,(r-t+p-1),j_1} \frac{\theta^r_{j_1}}{r!} = \frac{\theta_i^{t-p+1+k}}{(t-p+1+k)!} \tag{100}$$

$$t = p - 1 - k, \ldots, p - 1$$

$$k = 0, \ldots, p - 1.$$

$$\sum_{j_1 \in S_1} \sum_{r=t-p+1}^{t} a^{(k)}_{i,(r-t+p-1),j_1} \frac{\theta^r_{j_1}}{r!} + \sum_{j_2 \in S_2} b^{(k)}_{ij_2} \frac{\theta^{t-p}_{j_2}}{(t-p)!} = \frac{\theta_i^{t-p+1+k}}{(t-p+1+k)!} \tag{101}$$

$$t = p, p + 1, \ldots, p + \ell - 1$$

$$k = 0, 1, \ldots, p - 1$$

where we interpret values of $t < 0$ as being empty conditions; that is, the quantities referred to are non-existent and θ_j is the distance of ξ_j from the origin.

These results can be presented in a slightly different form and since it is advantageous to have them, they are given below as

Theorem 10'

In order that it be possible to represent the approximations ξ_i in the two forms given in (98), it is necessary and sufficient that the polynomial in θ

$$P_i^k(A, B, \theta) = \sum_{j_1 \in S_1} \sum_{r=o}^{p-1} a^{(k)}_{ir j_1} \frac{\theta^{\ell+r}_{j_1}}{(\ell+r)!} + \sum_{j_2 \in S_2} b^{(k)}_{ij_2} \frac{\theta_i^{\ell-1}}{(\ell-1)!} - \frac{\theta_i^{k+\ell}}{(k+\ell)!} \tag{102}$$

be the zero polynomial for $k = 0, \ldots, p - 1$.

That is, when we replace θ_j by, say, $\theta_o - \alpha_j h$, then we obtain a polynomial in θ_o. The coefficients must be such that as a function of

θ_o the polynomial be identically zero. Since this in turn implies that all the derivatives $D^j(P_i^k)(\theta_o) \equiv 0$, we obtain the equations (101), (100), and (99) simply by differentiating (102) and setting the derivative to zero. This, then, is an easy way to obtain equations that establish the conditions on the parameters. In the case of finite difference methods, we can obtain all the necessary parameter defining equations in this manner. The equations obtained can easily be seen to be exactly those that arise by requiring that the method be exact for polynomials of a certain order and then using the polynomials 1, t, t^2, ...

It is immediately obvious that we have the following

Corollary: The solution to the system of equations (99), (100), and (101) is origin independent for arbitrary ℓ.

Proof of Theorem 10: The proof is very straightforward and simply consists of substituting the definitions of the terms into (11') and properly collecting terms. Upon doing this, (11') becomes

$$\sum_{j_1} \sum_{k_1=o}^{p-1} \sum_{r=o}^{k_1+\ell} a^{(k)}_{ik_1j_1} \frac{\theta_{j_1}^r}{r!} D^{p-1-k_1+r}x(0) + \sum_{j_2} \sum_{r=o}^{\ell-1} b^k_{ij_2} \frac{\theta_{j_1}^r}{r!} D^{p+r}x(0)$$
$$- \sum_{r=o}^{k+\ell} \frac{\theta_i^r}{r!} D^{p-1-k+r}x(0) = 0$$

where $k \in P = \{0, \ldots, p-1\}$, $j_1 \in S_1$, $j_2 \in S_2$. We require that the conditions hold independent of the function x; therefore, we collect the coefficients of the derivatives of like order and set these terms separately to zero. When this is done, equations (99), (100), (101) will be obtained.

The proof of Theorem 10' is trivial; we have already stated it. Simply differentiate and set all derivatives to zero. The equations

obtained are (99), (100), and (101). Likewise, the corollary is obvious.

A few comments are in order considering these conditions established by ξ_i. If we use ξ_i in a substitution $X(\xi_i)$, then in reality all that is necessary is that the conditions hold for $k \in \overline{P}$ when $\overline{P} = \{k | \xi^{(k)}$ is explicit in $X(\xi)\}$ since only these ξ must have the representation $\xi_i^{(k)} = u^{(k)}(\theta_i) + \ldots$, $k \in \overline{P}$.

We now have all that is necessary to carry through the construction of the parameter defining equations by means of successive substitutions for any generalized RK or RKF scheme. The necessary results are summarized below.

Summary: To determine the nonlinear parameter defining equations by means of successive substitutions, we proceed as follows:

1) Problem $D^p x = X \circ \xi$, $\xi(0) = b$

2) SIP Definition 8 of the scheme interval parameters

3) Scheme Definition 1 (RK), Definition 10 (RKF)

4) Creation Carry out the scheme using as required

Theorem 6 - Substitution

Theorem 7 - Multiplication

Theorem 8, Theorem 9 - Translation

5) Condition A $\xi_i^{(k)} - u^{(k)}(\theta_i) = O(h^{\ell+k+1})$

Use Theorem 10.

6) Condition B $\xi_o^{(k)} - \xi^{(k)}(\theta_o) = O(h^{m+k})$

Equate the harmonics of the created approximation and of the solution (Theorem 4).

7) Condition C Carry out the necessary translations to obtain the definitions of any harmonics that have been used as undetermined parameters.

The equations defining the parameters of the scheme are the totality of Conditions A, B, C. The work necessary to actually carry out the above is formidable; however, this work is preserved in an ALGOL 60 program described in Chapter V and given in its entirety in Appendix II. Using this program, we need only supply Steps 1 - 3; Step 4 is done for us and the conditions appear as the result of carrying out Step 4. Because there is much flexibility in the definition of any particular scheme, we must in reality specify somewhat more than is indicated here to obtain a set of equations, but not much more, and the generation of schemes and their associated parameter defining equations is quite simple; the solution of these equations and the systematization of a search for new schemes is another problem.

We thus have all the equations necessary for the definition of the parameters of any constructed scheme. We have

i) Equations that arise by requiring that all approximations be of at least a given minimum order of accuracy.

ii) Equations that arise by matching the Taylor's series of the true solution with the coefficients of a chosen approximation, this approximation having been obtained constructively.

iii) Equations that arise by equating the appropriate implicit representation to the appropriate constructive representation, or to the appropraite "known" expansion.

Note that *ii* are the basic Runge-Kutta type equations, *i* are the finite difference methods equations, and *iii* are the result of using closed methods or else of mixing the two methods; that is, generalized RK or RKF methods.

IV. GENERALIZED R. K. METHODS FOR SYSTEMS OF p-th ORDER DIFFERENTIAL EQUATIONS

In Chapter IV, we treat the case of a system of p-th order ordinary differential equations written as $D^p x = X \circ \xi$, $\xi(a) = b$, $\xi = (D^{p-1}x_1 \ldots, D^o x)$. This is the same system treated in Chapter III; however, the development here will allow us to obtain an overall, global, view of the parameter defining equations. This chapter, thus, is an extension of the work of Chapter II to the $p > 1$ case and follows the development of that chapter to a large extent. In Chapter III, once a particular scheme was chosen, the results developed there enabled one to obtain the equations for that scheme. In this chapter, we obtain the totality of equations that are associated with the general scheme definition. The choice of a particular scheme simply selects from that set those equations that define the scheme parameters. This is, however, done at an expense -- our schemes are generalized Runge-Kutta (RK) schemes and not generalized Runge-Kutta-Frey (RKF) schemes.

In order to arrive at our results, we again, as was done in Chapter II, introduce approximations $\zeta \in R \to R^{n\times p}$, approximations ξ_i obtained from a generalized R.K. scheme, and a generic set of functions y. Using the generic y, we obtain from them weighted differentials W, derivative harmonics α, differentials A, weighted polynomials Φ, elementary polynomials Γ, polynomial weights γ, product coefficients Π, generalized RK harmonics H.

In a manner similar to Chapter II, we show that the derivatives of $\zeta = \Sigma\alpha_i W_i$, Theorem 1. Theorem 2 shows that $D^{r-1}(X \circ \xi) = \Sigma\, \alpha_i A_i$. We again have the factorization of $W = \Phi A$, Theorem 3. Using the results of Theorem 4, a general factorization theorem with respect to the generic y, we have from Theorem 5 that $\alpha\Phi = \alpha\gamma\Gamma = \Pi\Gamma = H$. Equation (45) then shows

us that the parameter defining equations for all ξ_i have the form $\Sigma H_i = \beta \frac{\theta^j}{j!}$ where on the left side we have the sum of generalized RK harmonics of the approximation ξ_i and on the right side the Taylor harmonics of the exact solution $\xi(\theta)$. We finally conclude the chapter with a definition of approximation harmonics, again denoted by α, derived from the generic z and use these harmonics to establish a direct connection between this chapter's work and that of Chapter III by means of Theorem 6 showing that $\xi_i = \Sigma \alpha_i A_i$ where the α's are approximation harmonics.

This chapter will be developed in a manner analogous to Chapter II. There the special case p = 1 was treated; here, the value of p is arbitrary. This will lead to a profusion of equations; essentially a new set for every value of $k \in P = \{0, \ldots, p - 1\}$. However, while this limits the actual use of the approach, in practice it is manageable for small values of p and particularly so if the first derivative is missing from X; that is, if $k \in \overline{P} = \{1\} \subset P = \{0, 1\}$ is the case. In general, when the k = 0 terms of $\xi^{(k)}$ are not explicit in X, there is a great simplification of the results. Thus, the work of this chapter will furnish not only a global view of the parameter defining equations, but also a means of checking some of the cases of the previous chapter. We shall also see that there is another way of viewing the work of Chapter III that could possibly lead to a simpler, or systematic development than has been carried out there. Since much of the work presented here is either an obvious extension of the analogous results of Chapter II or an obvious use of the work in Chapter III, we shall be briefer in our presentation and wherever possible simply indicate the appropriate section from which results come.

The problem to be solved is the same as that of Chapter III. We

restate it here for convenience.

Let R be the real line, $t \in R$. Define

$$N = \{0, \ldots, n-1\},\ P = \{0, \ldots, p-1\},\ L = \{0, \ldots m \ldots, n\times p - 1\}$$

where $m = i + kn$, $i \in N$, $k \in P$. Let R^n and $R^{n\times p}$ be respectively real n and $n\times p$ dimensional vector spaces. Define $x \in R \to R^n$, $X \in R^{n\times p} \to R^n$, $\xi \in R \to R^{n\times p}$. We wish to solve

$$\begin{aligned} D^p x &= X \circ \xi \\ \xi(0) &= a_L \end{aligned} \tag{1}$$

This problem is identical to that of Chapter III and a complete, explicit definition is given there.

To help carry out the analysis, we shall define functions $\zeta \in R \to R^{n\times p}$ using the notation $\zeta[m] = \zeta^{(k)}[i]$, $m = i + kn$ where $i \in N$, $k \in P$.

Definition 1:

$\zeta \in R \to R^{n\times p}$ is an approximation if, and only if

$\zeta = u + \eta$ where

$$\begin{aligned} u^{(k)} &= \sum_{r=0}^{\ell+k} \frac{I^r}{r!} D^{p-1-k+r} x(0) \\ \eta^{(k)} &= \sum_{r=\ell+k}^{\infty} \frac{I^{r+1}}{(r+1)!} \frac{(r+1)!}{(r-k)!} D^{r-k} \delta_r^{(k)}(0) \end{aligned} \tag{2}$$

$$D^{r-k} \delta_r^{(k)} = \sum_{j,k_o} g_{ik_oj}^{(k)} \eta^{(k_o)} + \sum_j f_{ijr}^{(k)} (X \circ \zeta)$$

where $k, k_o \in P = \{0, \ldots, p-1\}$; $g_{ik_oj}^{(k)}$, $f_{ijr}^{(k)} \in R$ are constants; $\zeta^{(k)}$, $u^{(k)}$, $\eta^{(k)}$, $\delta_r^{(k)}$ are $\in R \to R^n$ and $I \in R \to R$ such that $I(t) = t$.

As was previously done, define $\phi_i = \theta_i I \in R \to R$ and for any function y define $y_i = y \circ \phi_i$. Then associated with (2) is the set of functions which can be written as

$$\zeta_i = u_i + \eta_i$$

$$u_i^{(k)} = \sum_{r=o}^{\ell+k} \frac{(I\theta_i)^r}{r!} D^{p-1-k+r}x(0)$$

$$\eta_i^{(k)} = \sum_{r=\ell+k}^{\infty} \frac{I^{r+1}}{(r+1)!} \frac{(r+1)!}{(r-k)!} D^{r-k}\delta_{ir}^{(k)}(0) \tag{3}$$

$$\delta_{ir}^{(k)} = \sum_{j,k_o} g_{ik_oj}^{(k)} \eta_j^{(k_o)} + \sum_j f_{ijr}^{(k)} (X \circ \zeta_j).$$

In the work below, we shall, occasionally, omit the r dependence on δ and f. It should never be forgotten that such a dependence exists; especially when changing the range over which that index is summed.

When $\zeta_i^{(k)}$ is written out explicitly, we obtain

$$\zeta_i^{(k)} = u_i^{(k)} + \sum_{r=\ell+k}^{\infty} \frac{I^{r+1}}{(r+1)!} \frac{(r+1)!}{(r-k)!} \left[\sum_{j,k_o} g_{ik_oj}^{(k)} D^{r-k} \eta_j^{(k_o)}(0) + \sum_j f_{ijr}^{(k)} D^{r-k}(X \circ \zeta_j)(0) \right]. \tag{4}$$

Let $g_{ik_oj}^{(k)} \equiv 0$, $f_{ijr}^{(k)} = 0$ for $i \neq j$ and $f_{iir}^{(k)} = \frac{(r-k)!}{(r+1)!} \theta_i^{k+1}$.

Substitute these into (4) to obtain

$$\zeta^{(k)}(\theta_i I) = u^{(k)}(\theta_i I) + \sum_{r=\ell+k}^{\infty} \frac{(I\theta_i)^{r+1}}{(r+1)!} D^{r-k}(X \circ \zeta)(0). \tag{5}$$

We also have that $D^p x = X \circ \xi$ where $\xi = u + v$ and v is defined as

$$v^{(k)} = \sum_{r=k+\ell}^{\infty} \frac{I^{r+1}}{(r+1)!} D^{p-k+r}x(0).$$

Thus

$$\xi^{(k)}(\theta_i) = u^{(k)}(\theta_i) + \sum_{r=\ell+k}^{\infty} \frac{\theta_i^{r+1}}{(r+1)!} D^{r-k}(X \circ \xi)(0). \tag{6}$$

A comparison of (5) and (6) shows that

$$\zeta_i(1) = \xi(\theta_i). \tag{7}$$

These results are summarized as

Property 1: If ζ_i is an approximation defined by (7) and if x is a solution to $D^p x = X \circ \xi$, then $\zeta_i(1) = \xi(\theta_i)$ provided that $g^{(k)}_{ik_oj} \equiv 0$ and

$$f^{(k)}_{ijr} = 0, \quad i \neq j$$

$$f^{(k)}_{iir} = \frac{(r-k)!}{(r+1)!}\,\theta_i^{k+1}.$$

That is, with a suitable choice of parameters, the approximation ζ reduces to the vector $\xi = (D^{p-1}x, \ldots, D^o x)$ derived from the true solution x.

In the present development, we shall not use the Jacobian matrix J_{NL} and will treat only generalized R.K. schemes the definition of which is given in Definition 1 of Chapter III and which we repeat here in a slightly more convenient form.

Definition 2: The approximation ξ_i is said to be obtained by means of a generalized R.K. scheme if, and only if

$$\xi^{(k)}_i = \sum_{j,k_o} g^{(k)}_{ik_oj}\,\xi^{(k_o)}_j + \sum_j a^{(k)}_{ij}\,X(\xi_j) \tag{8}$$

where $X \in R^{n\times p} \to R^n$ is that of (1), the g and a are elements of R, and ξ_j is an approximation obtained in the same fashion.

Our previous comments in Chapter II concerning ξ_j are relevent here. However, now $\xi_j \in R^{n\times p}$ is an approximation to $\xi(\theta_j) = (D^{p-1}x(\theta_j), \ldots, D^o x(\theta_j))$ consisting of all the derivatives of order lower than the order of the differential equation. Also, we shift back and forth between R^n and $R^{n\times p}$ using $\xi^{(k)}$ and ξ, respectively. We shall do this repeatedly and the meaning should be clear from the context.

We know that if the conditions established by Theorem 10 of Chapter III hold, then we can write any approximation as

$$\xi_i^{(k)} = u_i^{(k)} + \sum_{j,k_o} \cdots (\xi_j^{(k_o)} - u_j^{(k_o)}) + \sum_j \cdots (X(\xi_j) - R_j). \tag{9}$$

We can, by means of a proper choice of coefficients, write $\zeta_i^{(k)}$ in the same form. Write $\zeta_i^{(k)}$ as

$$\zeta_i^{(k)} - u_i^{(k)} = \eta_i^{(k)} = \sum_{r=o}^{\infty} \frac{I^r}{r!} D^r \eta_i^{(k)}(0). \tag{10}$$

For $r < \ell$, the derivatives of $\eta_i^{(k)}$ evaluated at zero are equal to zero. Thus (10) becomes

$$\sum_{r=\ell}^{\infty} \frac{I^r}{r!} D^r \eta_i^{(k)}(0) = \eta_i^{(k)} = \zeta_i^{(k)} - u_i^{(k)}. \tag{11}$$

By expanding $X \circ \zeta_i$ in a Taylor's series and defining

$$\bar{R}_i = \sum_{r=o}^{\ell-1} D^r(X \circ \zeta_i)\,(0)\ \frac{I^r}{r!} \tag{12}$$

we obtain

$$\sum_{r=\ell}^{\infty} \frac{I^r}{r!} D^r(X \circ \zeta_i)(0) = X \circ \zeta_i - \bar{R}_i. \tag{13}$$

Now take (4), factor out I^{k+1} and substitute (11) and (13) to obtain

$$\zeta_i^{(k)} = u_i^{(k)} + I^{k+1} \left[\sum_{j,k_o} g_{ik_oj}^{(k)} \{\zeta_j^{(k)} - u_j^{(k)}\} + \sum_j f_{ijr}^{(k)} \{X \circ \zeta_j - \bar{R}_j\} \right]. \tag{14}$$

The evaluation of (14) at 1 shows that if we use the previous definition of R_j in (9), that is (49) of Chapter III, then we have that $\zeta_i^{(k)} = \xi_i^{(k)}$ provided $f_{ijr}^{(k)} \equiv a_{ij}^{(k)}$ for all r, provided we are consistent in matching all the ξ_i and ζ_i, and provided

$D^r(X \circ \zeta_i)(0) = \theta_i^r D^r(X \circ u)(0)$. This last requirement is true since $\zeta_i = u_i + \eta_i$ and for $r < \ell$ we have that $D^r\zeta_i = Du_i$, $D^r\eta_i = 0$.

We summarize these results as

Property 2. If $\zeta_i \in R \to R^{n\times p}$ is an approximation defined by (3) and if ξ_i is an approximation defined by means of a generalized R. K. scheme, then $\zeta_i(1) = \xi_i$ provided the parameters $g_{ik_oj}^{(k)}$, $f_{ijr}^{(k)} = a_{ij}^{(k)}$ common to both approximations satisfy the conditions of Theorem 10, Chapter III referred to here as Conditions A.

The problem of finding a generalized R. K. scheme can then be stated as follows: Let $\xi \in R \to R^{n\times p}$ be the solution vector of the differential equation $D^p x = X \circ \xi$. Let $\zeta_i(1)$ as defined by (3) be the desired approximation to $\xi(\theta_i)$. Write the Taylor expansions of ξ and ζ_i as

$$\begin{aligned} \xi^{(k)}(\theta_i) &= u^{(k)}(\theta_i) + \sum_{r=\ell}^{\infty} \frac{\theta_i^{r+1+k}}{(r+1+k)!} D^r(X \circ \xi)(0) \\ \zeta_i^{(k)}(1) &= u_i^{(k)}(1) = \sum_{r=\ell}^{\infty} \frac{1}{(r+1+k)!} D^{r+1+k} \zeta_i^{(k)}(0). \end{aligned} \tag{15}$$

Choose the parameters g and f that appear in ζ_i so that these two series agree to a given order. When the parameters are chosen, then Property 2 establishes the explicit representation of the scheme.

Note that in practice we usually will find ourselves working from the scheme definition, by means of Property 2, back to the parameter defining equations.

We see that, as was the case in Chapter II with $p = 1$, we are again in the position of having to find expansions for the derivatives in (15) and that we must do so in such a fashion that the two series can actually be compared term by term. Although this task is more complicated than for the $p = 1$ case, we can easily outline what the results will be. The

basic set of functions that we use for the derivatives of $X \circ \xi$ we already have; they are the differentials of Chapter III. As was done before, we shall define weighted differentials and all the related quantities that went with them. However, we shall have more functions than previously; all these quantities will now depend on $k \in P = \{0, \ldots, p - 1\}$. The actual manner in which we define these quantities will be made clear later, but we have the result that unless $k = 0$, or some small number (unless $\overline{P} \subset P$ is "small"), the multiplicity of functions and resultant equations soon becomes unmanageable when they are actually written out.

In the development that follows, we shall use a mixture of the notation that has been previously defined. However, if one ascertains from the context the space in which a function lies, then there should be little confusion. In short, X is consistently the right side of the differential equation; ξ the solution vector $(D^p x, \ldots, D^o x)$; ζ an approximation in the function space; ξ_i are approximation in the vector space. The parameter k is always an element of P, or $\overline{P} \subset P$, where $P = \{0, \ldots, p - 1\}$. Functions, or vectors, with a superscript k that lie in R^n can always be written as being in $R^{n \times p}$ by dropping the k; for example, $\zeta^{(k)} = \zeta[i + kn]$ and conversely. We shall always sum on repeated capital subscripts; incidentally, this means that $A = B_{N_1} B_{N_1} B_{N_1} = \sum_{i_1} B_{N_1}[i_1] B_{N_1}[i_1] B_{N_1}[i_1]$ where all factors have the same index value. On the other hand, we shall never use lower case subscripts to denote components, only to denote elements of a set. Thus, as was the case in Chapter III, $\zeta_{iN}^{(k)}$ is the i-th element in the set of all $\zeta_N^{(k)}$ and these have components in the set N. We make no distinction between ζ_L and ζ and will likewise use such notations as $\zeta_N^{(k)}$ and $\zeta^{(k)}$ as being equivalent. As we have said, the context in which the items appear should identify the space of which they are an element.

Referring to (15), we see that it is necessary to calculate the derivatives of $\zeta_i^{(k)}$ starting with $D^{\ell+1}\zeta_i^{(o)}(0)$. This is equivalent to evaluating $D^j \eta_i^{(k)}(0)$ for $j = r + k + 1 \geq \ell + 1$. More specifically, we have that

$$D^{r+1+k}\,\zeta_i^{(k)}(0) = D^{r+1+k}\,\eta_i^{(k)}(0) = \frac{(r+1+k)!}{r!}\,D^r\,\delta_i^{(k)}(0) \tag{16}$$

for $r \geq \ell$. By means of (3), we can write

$$\delta_i^{(k)} = \sum_{j,k_o} \ldots \eta_j^{(k_o)} + \sum_j \ldots X(u_j + \eta_i) \tag{17}$$

which when the expansion of the second term is carried out becomes

$$\delta_i^{(k)} = \sum_{j,k_o} \ldots \eta_j^{(k_o)} + \sum_j \ldots [X \circ u_j + \sum_s \frac{1}{s!}\,D_{L_1}\ldots L_s\, X \circ u_j\, \eta_{jL_1}\ldots\eta_{jL_s}]. \tag{18}$$

We now differentiate (18), multiply both sides by $\frac{(r+1+k)!}{r!}$, make use of the matrices $I_{LN}^{(k)}$ and equation (16) to obtain the following result

$$\begin{aligned} D^{r+1+k}\,\eta_i^{(k)}(0) = {} & \sum_j \frac{(r+1+k)!}{r!}\,f_{ijr}^{(k)}\; D^r(X \circ u_j)(0) + \\ & \sum_j \sum_{s=1}^{\infty} \sum_{i_1\ldots i_s} \sum_{k_1\ldots k_s} \frac{1}{s!}\,\alpha_{ri_1\ldots i_s}\,\frac{(r+1+k)!}{r!}\,f_{ijr}^{(k)} \cdot \\ & D^{r-i_1}(D_{L_1}\ldots L_s\, X \circ u_j)(0)\; I_{L_1N_1}^{(k_1)}\ldots I_{L_sN_s}^{(k_s)} \\ & D^{i_1-i_2-k_1+k_1}\,\eta_{jN_1}^{(k_1)}(0)\ldots D^{i_s-k_s+k_s}\,\eta_{jN_s}^{(k_s)}(0) + \\ & \sum_j \sum_{k_o} \frac{(r+1+k)!}{r!}\; g_{ik_oj}^{(k)}\; D^{r-k_o+k_o}\,\eta_j^{(k_o)}(0). \end{aligned} \tag{19}$$

We have deliberately put in the extra value of k_i and as usual $\alpha_{ri_1\ldots i_s} = \binom{r}{i_1}\binom{i_1}{i_2}\ldots\binom{i_{s-1}}{i_s}$. Equation (19) will serve for this

chapter the same central position that (16') did for Chapter II and, in fact, for p = 1, P = {0}; this reduces to that equation as may easily be verified.

Since $D^r\eta_i^{(k)}(0) = 0$ for $r < \ell + 1 + k$, the range of the indices in (19) can be materially reduced. This leads to the same normal index set that was given in equations (22), Chapter III. We write this normal index range here for reference

$$
\begin{aligned}
&0 \leqq k_1 \leqq i_1 - i_2 - (\ell + 1)\\
&0 \leqq k_2 \leqq i_2 - i_3 - (\ell + 1)\\
&\quad\vdots\\
&0 \leqq k_s \leqq i_s - (\ell + 1)\\
&0 \leqq k_o \leqq r - (\ell + 1)\\
&s(\ell + 1) \leqq i_1 \leqq r\\
&(s - 1)(\ell + 1) \leqq i_2 \leqq i_1 - (\ell + 1)\\
&\qquad\qquad\vdots\\
&(\ell + 1) \leqq i_s \leqq i_{s-1} - (\ell + 1)\\
&1 \leqq s \leqq r \div (\ell + 1)
\end{aligned}
\tag{20}
$$

where it is assumed that $k_i \in \overline{P}$, $i = 0, 1, \ldots$ for all k. Thus, the upper bounds on k_i will not be achieved if the upper bound is not in the set $\overline{P}$. We also have the following bounds

$$
\begin{aligned}
&(\ell + 1) \leqq i_1 - i_2 \leqq r + (1 - s)(\ell + 1)\\
&(\ell + 1) \leqq i_2 - i_3 \leqq r + (1 - s)(\ell + 1)\\
&\qquad\qquad\vdots\\
&(\ell + 1) \leqq i_s \leqq r + (1 - s)(\ell + 1)
\end{aligned}
\tag{21}
$$

and these, in turn, give

$$
\begin{aligned}
&0 \leqq k_1 \leqq r - s(\ell + 1)\\
&0 \leqq k_2 \leqq r - s(\ell + 1)\\
&\quad\vdots\\
&0 \leqq k_s \leqq r - s(\ell + 1).
\end{aligned}
\tag{22}
$$

It is useful to have these bounds when determining whether all the weighted differentials and their related quantities have been obtained.

We see that (19) expresses $D^{r+1+k}\,\eta_i^{(k)}(0)$, for any $k \in \overline{P}$, in terms of lower order derivatives; therefore, if we know the lowest order derivative, we can, indeed, calculate the next higher one and thus proceed to obtain all the necessary derivatives.

As was previously done, we shall define functions $W \in R \to R^n$ which are again called weighted differentials of given rank R, order r, degree s where $R \geqq r \geqq \ell + 1$ are integers, $s \geqq 0$ is an integer, and we assume that $k_i \in \overline{P} = \{k \mid \xi^{(k)}$ is explicit in $X \circ \xi\}$. We shall, however, make use of a generic definition to do this since this will help simplify the presentation and, at the same time, illustrates that all these quantities are obtained in, essentially, the same manner. To do this, we need only extend slightly the quantities defined in Definition 3 of Chapter III. We are thus lead to

Definition 3: The symbol y_i^k is a generic y of rank R, order r, degree s = 0, position 1, with R = r if, and only if

$$
y_i^k\,[R, r, 1] = Y_i^k\,[R] \circ (z_j^k\,[R\text{-}1, r\text{-}1, 0]) \tag{23}
$$

where z_j^k is a generic z of rank R - 1, order r - 1, position 0, degree 0. The symbol y_i^k is a generic y of rank R, order r, degree s, position a if

$$
y_i^k\,[R, r, a] = Y_i^k\,[R, k_o] \circ (z_j^{k_o}\,[R_o, r_o, a_o]) \tag{24}
$$

where $R = 1 + R_o + k_o$, $r = r_o$, and $z_j^{k_o}$ is a generic z of rank R_o, order r_o, degree s, position $a_o \neq 0$, or if

$$y_i^k [R, r, a] = Y_i^k [R, r_o, k_1, \ldots, k_s] \circ (z_j^{k_1} [R_1, r_1, a_1], \ldots, z_j^{k_s} [R_s, r_s, a_s]) \tag{25}$$

where $$R = 1 + r_o + \sum_{j=1}^{s} (k_i + R_i)$$

$$r = 1 + r_o + \sum_{j=1}^{s} (k_i + r_i)$$

and $z_j^{k_i}$ are generic z of rank R_i, order r_i, position $a_i \neq 0$. In all cases, $Y_\circ$ is the generic operator of the definition.

In the work that follows, we shall assume that Properties 1 - 4 of Chapter III are true here. This is not inherent in our definition; however, the pattern of generation is to be identical for all of the quantities that are the specific realization of Definition 3 and, in particular, the weighted differentials. Because of the way that they are defined, we are lead to Properties 1 - 4 which are stated here in terms of the generic y and z.

Property 1: In the definition of the generic y, a permutation of the set $\{k_i\}$ is equivalent to the same permutation of the set $\{(R_i, r_i, a_i)\}$ and, in general, a new generic y will arise from this operation.

Property 2: In the definition of the generic y, a permutation of the $\{k_i\}$ followed by an identical permutation of the corresponding $\{(R_i, r_i, a_i)\}$ does not lead to a new generic y. That is, any permutation of the set $\{(k_i, R_i, r_i, a_i)\} = \{n_i\}$ gives rise to the same generic y.

Property 3: If $k_i = k_j$, then the permutation of (R_i, r_i, a_i) and (R_j, r_j, a_j) does not give rise to a new y. Likewise, if (R_i, r_i, a_i) is identical to (R_j, r_j, a_j), then the permutation of k_i and k_j does not give rise to a new y.

Property 4: Given y defined by (25) of rank R, order r, degree s, position a, we obtain all distinct y of rank R, order r, degree s with the same factors by considering all the destinct permutations of $(k_1, \ldots k_s)$ with the understanding that if $(R_i, r_i, a_i) = (R_j, r_j, a_j)$ then that permutation is not distinct. That is, we associate to the couples $(k_i, (R_i, r_i, a_i))$ the integer n_i. Two couples are considered identical if either $k_i = k_j$ or $(R_i, r_i, a_i) = (R_j, r_j, a_j)$ or both. Distinct couples have distinct integers n_i. We form the distinct permutations of the set $\{n_i\}$; then the correspondence $n_i \to k_i$ or $n_i \to (R_i, r_i, a_i)$ will furnish all the distinct y with the same factors.

We can, if we wish, again arrive at a normal form

$y = Y \circ ((z_1)^{\mu_1} \ldots (z_s)^{\mu_\sigma})$, with $s = \sum_{i=1}^{\sigma}$, by suitable permutations and re-indexing. In the above definition and subsequent realizations, it is always assumed that $R \geq r \geq \ell + 1$ and $k_i \in \overline{P} = \{k | \xi^{(k)}$ is explicit in $X \circ \xi\}$.

We now proceed to define the weighted differentials.

Definition 4: Define

$$
\begin{aligned}
& z_j^k [R-1, r-1, 0] \equiv 1, \; z_j^{k_i} [R_i, r_i, a_i] \equiv y_j^{k_i} [R_i, r_i, a_i], \; a_i \neq 0 \\
& Y_i^k [R] \circ \equiv \sum_j \frac{(R+k)!}{(R-1)!} f_{ijR}^{(k)} C_{jN_o}^{(R-1)} . \\
& Y_i^k [R, k_o] \circ \equiv \sum_j \frac{(R+k)!}{(R-1)!} g_{ik_oj}^{(k)} . \qquad (26) \\
& Y_i^k [R, r_o, k_1, \ldots, k_s] \circ \equiv \sum_j \frac{(R+k)!}{(R-1)!} f_{ijR}^{(k)} C_{jN_oN_1 \ldots N_s}^{(r_o, k_1, \ldots k_s)} .
\end{aligned}
$$

where

$$
C_{jN_oN_1 \ldots N_s}^{(r_o, k_1, \ldots, k_s)} = D^{r_o} (D_{L_1 \ldots L_s} X \circ u_j) I_{L_1N_1}^{(k_1)} \ldots I_{L_sN_s}^{(k_s)} .
$$

Then the generic $y_i^{k'}$ has as its realization the weighted differentials

W_i^k [R, r, a] of rank R, order r, position a, degree s.

As was done before, we make use of the shortened notation and write W as $\langle R - 1\rangle^{(k)}$ for (23), $E^{(k)}(W^{(k_o)})$ for (24) and $\langle r_o, k_1, \ldots, k_s; W_1 \ldots W_s\rangle^{(k)}$, or $\langle K; (W_1)^{\mu 1} \ldots (W_\sigma)^{\mu\sigma}\rangle$, for (25).

The quantities are tabulated in Table VIII of Appendix I using a slightly different, shorter notation than that which defines the generic y. This table is actually a generic table in which we have, essentially, identified the z to be the same as the y. The construction of that table is completely described in Appendix I. However, a few comments will be made here. The pattern of generation of the generic y is the same as we previously used in Chapters II and III. We start with lowest order and rank, degree zero, and work upward in degree and order; but, now we have the possibility of items with rank R greater than the order. It is convenient to envision the elements of constant order r and increasing rank R to form the rows and the elements of constant rank R and increasing order r the column of a square array when generating these functions. This is what has been done in Table VIII. The best way to become familiar with how to generate these quantities is to actually carry out some examples.

We give below a short table to show how this can be done. It will immediately be noticed that we have implicitly assumed a one-to-one correspondence between the y and the z, except for z^k [R - 1, r - 1, 0]. This is not inherent in Definition 3, but since, in fact, we shall have this correspondence in all our realizations of the generic y, we shall assume that this is the situation. A comparison of Table I with Table VIII of Appendix I will show that this is just a very short section of that table. The notation of Table VIII has been shortened to aid in its construction; however, if these quantities are generated by means of an algorithm, their complete identification is needed; this is furnished in their definition.

TABLE I

Name	Definition	Name	Definition
y^k [R, r, 1]	Y^k [R] ∘ (z^k [R-1, r-1, 0])	y^k [R+1, r, 1]	Y^k [R+1, 0] ∘ (z^o [R, r, 1])
y^k [R+1, r+1, 1]	Y^k [R+1] ∘ (z^k [R, r, 0])	y^k [R+2, r+1, 1]	Y^k [R+2, 0] ∘ (z^o [R+1, r+1, 1])
y^k [R+1, r+1, 2]	Y^k [R+1, 0, 0] ∘ (z^o [R, r, 1])	y^k [R+2, r+1, 2]	Y^k [R+2, 0] ∘ (z^o [R+1, r+1, 2])
y^k [R+2, r+2, 1]	Y^k [R+2] ∘ (z^k [R+1, r+1, 0])	y^k [R+3, r+2, 1]	Y^k [R+3, 0] ∘ (z^o [R+2, r+2, 1])
y^k [R+2, r+2, 2]	Y^k [R+2, 1, 0] ∘ (z^o [R, r, 1])	y^k [R+3, r+2, 2]	Y^k [R+3, 0] ∘ (z^o [R+2, r+2, 2])
y^k [R+2, r+2, 3]	Y^k [R+2, 0, 0] ∘ (z^o [R+1,r+1,1])	y^k [R+3, r+2, 3]	Y^k [R+3, 0] ∘ (z^o [R+2, r+2, 3])
y^k [R+2, r+2, 4]	Y^k [R+2, 0, 0] ∘ (z^o [R+1,r+1,2])	y^k [R+3, r+2, 4]	Y^k [R+3, 0] ∘ (z^o [R+2, r+2, 4])
y^k [R+2, r+2, 5]	Y^k [R+2, 0, 1] ∘ (z^1 [R, r, 1])	y^k [R+3, r+2, 5]	Y^k [R+3, 0] ∘ (z^o [R+2, r+2, 5])

It is possible to expand the derivatives $D^{r+l+k}\,\eta_i^{(k)}(0)$ into these differentials. However, the fact that $k \in \overline{P}$ and not simply zero, as was previously the case, requires that we modify our results. In the work which follows, we shall always assume that there is established a one-to-one correspondence between the various items that are generated as specific realizations of the generic y. Thus, when we write $\alpha_{Rrj}^{(k)}$ and $W_i^k\,[R, r, j]$, we implicitly assume that these items were generated in the same manner. The generic y establishes a pattern for all quantities derived from them and this pattern, once established, remains fixed.

The expansions of the derivatives of $\eta_i^{(k)}$ are carried out using Definition 5 of the derivative harmonics and Theorem 1. We state the theorem first.

Theorem 1

Let $R \geq \ell + 1$, $k \in \overline{P} = \{k | \xi^{(k)}$ is explicit in $X \circ \xi\}$, $S_R = \{j | W_i^{(k)}$ $[R, r, j]$ has rank R, order r$\}$. Let α_{Rrj}^k be the derivative harmonic corresponding to the weighted differential $W_i^k\,[R, r, j]$. Then

$$D^{R+k}\,\eta_i^{(k)}(0) = \sum_{r=\ell+1}^{R} \sum_{j \in S_R} (R + k)!\; \alpha_{Rrj}^k\; W_i^{(k)}[R, r, j](0). \qquad (27)$$

The derivative harmonics are non-negative, rational coefficients independent of i and are defined in

Definition 5: Define

$$z_j^k\,[R\text{-}1, r\text{-}1, 0] \equiv 1,\; z_j^k\,[R, r, a] \equiv y_j^k\,[R, r, a],\; a \neq 0$$

$$Y_i^k\,[R] \circ \equiv \frac{1}{(R+k)!}\,.$$

$$Y_i^k\,[R, k_o] \circ \equiv \frac{(R_o + k_o)!}{(R + k)!}\,. \qquad (28)$$

$$Y_i^k\,[R, r_o, k_1, \ldots k_s] \circ \equiv \frac{(R\text{-}1)!}{(R+k)!}\;\frac{1}{r_o!\;(\omega_1)!\ldots(\omega_\delta)!}\,.$$

where ω_i is the number of times that (k_i, R_i, r_i, a_i) appears as a factor

in y_i^k [R, r, a]. Then the generic y_i^k [R, r, a] has as its realization the derivative harmonic α^k_{Rra}. We note that when R = r these derivative harmonics, when multiplied by (R + k)!, are identical to those of Chapter III.

Proof of Theorem 1: The proof of this theorem is essentially the same as that of the constructive case in Chapter III where we found the expansion of the derivatives of x. Consider first the case $R = \ell + 1$. From (19) we have that

$$D^{\ell+1+k}\,\eta_i^{(k)}(0) = 1 \cdot \langle \ell \rangle^{(k)}(0) \tag{29}$$

where we have used the fact that, because the indices lie in the normal set, there are no terms other than the first term. Now, we assume the results are true for all ranks less than R and substitute (27) into (19) to obtain

$$D^{R+k}\,\eta_i^{(k)}(0) = \langle R-1 \rangle^{(k)}(0) + \sum_s \sum_{i_1\cdots i_s} \sum_{k_1\cdots k_s} \sum_{r_1\cdots r_s} \sum_{j_1\cdots j_s} \frac{1}{s!}$$

$$\alpha_{R-1,i_1\cdots i_s}\ \alpha^{k_1}_{R_1 r_1 j_1}\cdots\alpha^{k_s}_{R_s r_s j_s}\ \langle R-1-i_1,\ k_1,\ldots,k_s;$$

$$W^{(k_1)}[R_1,\ r_1,\ j_1]\cdots W^{(k_s)}[R_s,\ r_s,\ j_s]\rangle\,(0) + \sum_{k_o}\sum_{r_o}\sum_{j_o}\alpha^{k_o}_{R_o r_o j_o}\cdot$$

$$E^{(k)}(W^{(k_o)}[R_o,\ r_o,\ j_o])(0). \tag{30}$$

where we have left out the scaling factor $\frac{1}{(R+k)!}$ in the definition of the α. We see that the derivative harmonics furnish the correct coefficient for the degree zero terms and for the last term obtained using the E operator. The only question is the general coefficient of the second term in the right member of (30). We proceed exactly in the same manner as previously. The expansion (27) is written for the left side (30) and

then we establish the valid choices of indices for the right side by permutting the couples (k_i, R_i, r_i, j_i) that appear in the factors of the general term. We will obtain as the general coefficient

$$\alpha^{k}_{Rrj} = \frac{(R-1)!}{r_o!} \frac{1}{(\omega_1)!\ldots(\omega_\delta)!} \frac{\alpha^{k_1}_{R_1 r_1 j_1}}{(R_1+k_1)!} \ldots \frac{\alpha^{k_s}_{R_s r_s j_s}}{(R_s+k_s)!} \tag{31}$$

where ω_i is the number of times that (k_i, R_i, r_i, j_i) appears as a factor in $W^k_i [R, r, j]$. If we now multiply (31) by $\frac{1}{(R+k)!}$ and refer to the definition of the derivative harmonics, we see that they are indeed the correct coefficients.

Since we eventually plan to use the differentials A of Chapter III as our basis, it is convenient to have them defined directly in terms of the generic y. It is possible to actually consider the A as a subset of the W by using a proper choice of the parameters g, f, and θ that appear in W. If we set all $g \equiv 0$, $f^i_{ii} = \frac{(R-1)!}{(R+k)!}$, $f^k_{ij} = 0$ if $i \neq j$, $\theta_i = 1$, then we can write $A [r, a] = W [r, r, a]$ and have the desired differentials. This, however, is not a convenient way to generate the differential A and we give below an explicit definition of these quantities.

Definition 6: Define

$$\begin{aligned}
&z^k_j [R - 1, r - 1, 0] \equiv 1, \; z^k_j [R, r, a] \equiv y^k_j [R, r, a], \; a \neq 0 \\
&Y^k_i [R] \circ \equiv C^{(R-1)}_{N_o} \\
&Y^k_i [R, k_o] \circ \equiv 0 \\
&Y^k_i [R, r_o, k_1, \ldots, k_s] \circ \equiv C^{(r_o, k_1, \ldots, k_s)}_{N_o N_1 \ldots N_s}
\end{aligned} \tag{32}$$

where

$$C^{(r_o, k_1, \ldots, k_s)}_{N_o N_1 \ldots N_s} = D^{r_o}(D_{L_1 \ldots L_s} X \circ u) \, I^{(k_1)}_{L_1 N_1} \ldots I^{(k_s)}_{L_s N_s}$$

then the generic y_i^k [R, r, a] has as its realization the differential A [r, a] = y_i^k [r, r, a] and the trivial set zero for those y with R > r.

By convention, we ignore this set of zeros and for each set $\{y^k [R, r, a] | k \in \overline{P}\}$, since all items are identical and independent of k, we choose one of these items as a representative and any subsequent reference to y^k [R, r, a] refers to this representative.

The previously introduced notation $\{r_o, k_1, \ldots, k_s, A_1, \ldots A_s\}$ will be used and also its normal form wherever convenient. This has been previously described in Chapter III

We next define the weighted polynomials Φ of given rank R, order r, position a, degree s. There is established by the generic y a one-to-one correspondence between the W and Φ which we write as $\Phi^{(k)}$ [R, r, a] $\longleftrightarrow$ $W^{(k)}$ [R, r, a].

Definition 7: Define

$$
\begin{aligned}
& z_j^k [R - 1, r - 1, 0] \equiv 1, \; z_j^k [R, r, a] \equiv y_j^k [R, r, a], \; a \neq 0 \\
& Y_i^k [R] \circ \equiv \sum_j \frac{(R+k)!}{(R-1)!} f_{ijR}^k \theta_j^{R-1} . \\
& Y_i^k [R, k_o] \circ \equiv \sum_j \frac{(R+k)!}{(R-1)!} g_{ik_oj}^{(k)} . \\
& Y_i^k [R, r_o, k_1, \ldots, k_s] \circ \equiv \sum_j \frac{(R+k)!}{(R-1)!} f_{ijR}^{(k)} \theta_j^{r_o} .
\end{aligned}
\tag{33}
$$

then the generic y_i^k [R, r, a] has as its realization the weighted polynomial Φ_i^k [R, r, a].

Again, we use the abbreviated notation $[r - 1]^{(k)}$ for (23), $G^{(k)}(\Phi^{k_o})$ for (24), and $[r_o, k_1, \ldots, k_s; \Phi_1^{(k_1)} \ldots \Phi_s^{(k_s)}]$ for (25).

We note that there exists a one-to-one correspondence between the differentials A [r, a] of order r, degree s and the set of all W_i^k [R, r, a] of order r, rank R $\geq$ r, degree s. This is the same correspondence previously

encountered in Chapter II; however, now $k \in \overline{P}$.

With regard to the derivatives of x, we already know that they can be expanded into the derivative harmonics β_{ra} of Chapter III; these harmonics are, however, the same as $(R + k)!\ \alpha^{k}_{rra}$ which can easily be verified. We restate that fact here for reference.

Theorem 2:

$$D^{r-1}(X \circ \xi)(0) = \sum_{a \in S_r} \beta_{ra}\ A^{(r,a)}(0) \tag{34}$$

where $\beta_{ra} = (r + k)!\ \alpha^{k}_{rra}$ for any $k \in \overline{P}$ and $S_r = \{a | A^{(r,a)}$ has order $r\}$.

The connection between the W and A is easily made by

Theorem 3:

$$W_i^{(k)}[R, r, j] = \Phi_i^{(k)}[R, r, j] \cdot A[r, a] \tag{35}$$

where W, Φ, and A have their usual meanings. This leads to the explicit relations

$$\begin{aligned}
&\langle r - 1\rangle^{(k)} = [r - 1]^{(k)} \cdot \{r - 1\} \\
&E^{(k)}(\langle K; W_1^{(k_1)} \ldots W_s^{(k_s)}\rangle^{(k_o)}) = G([K; \Phi_1^{(k_1)} \ldots \Phi_s^{(k_s)}]) \cdot \\
&\qquad\qquad \{K; A_1^{(r_1)} \ldots A_1^{(r_s)}\} \\
&\langle K; W_1^{(k_1)} \ldots W_s^{(k_s)}\rangle^{(k)} = [K; \Phi_1^{(k_1)} \ldots \Phi_s^{(k_s)}] \cdot \{K; A_1^{(r_1)} \ldots A_s^{(r_s)}\}
\end{aligned} \tag{36}$$

where $K = (r_o, k_1, \ldots, k_s)$, the correspondence $W_i^{(k_i)} = \Phi_i^{(k_i)}\ A^{(r_i)}$ is assumed, $(k, k_i) \subset \overline{P}$, and $j \in S_R = \{j | W$ has rank R, order $r\}$.

Note that the use of the generic definition without a change in its pattern of creation ensures that the proper correspondence in sequential position is maintained between all terms of (35). We are thus able to factor out the A in any expansion in which W appears.

Proof: The proof is an inductive one using the definitions of the W, Φ, and A. It is true for s = 0, R = r so the inductive process can be started. If the results are true for degree s, order r, rank R, then they are likewise true for degree s, order r, rank R + 1 as can easily be verified. Thus, if the results are true for any R = r element, then they are true for the whole row with R > r. On the other hand, if they are true for any row, then they are true for the appropriate row with higher rank and order. Thus, they are true for all elements.

We are now in a position to rewrite our expansion (27) of the derivatives of η_i. Upon substituting into (27) the factorization of W into ΦA, we obtain

$$D^{R+k}\,\eta_i^{(k)}(0) = \sum_{r=\ell+1}^{R} \sum_{a\in S_r} \sum_{j\in S_{Ra}} (R+k)!\;\alpha^{k}_{Rrj}\,\Phi_i^{(k)}\,[R, r, j]\,A^{(r,a)}(0) \qquad (37)$$

where

$$S_r = \{a \mid A^{(r,a)} \text{ has order } r\}$$

$$S_{Ra} = \{j \mid W^{(k)}\,[R, r, j] \text{ has rank R, order r, and corresponds to } A^{(r,a)} \text{ of order r}\}.$$

If, in equation (15), we now replace the derivatives by their expansions into the differentials A(0), the following equivalent expansions are obtained.

$$\xi^{(k)}(\theta_i) = u^{(k)}(\theta_i) + \sum_{r=\ell+1}^{\infty} \sum_{a\in S_r} \beta_{ra}\,\frac{\theta_i^{r+k}}{(r+k)!}\,A^{(r,a)}(0)$$

$$\zeta_i^{(k)}(1) = u_i^{(k)}(1) + \sum_{r=\ell+1}^{\infty} \sum_{a\in S_r} \Big\{ \sum_{R=r}^{\infty} \sum_{j\in S_{Ra}} \alpha^{k}_{Rrj}\,\Phi_i^{(k)}[R,r,j] \Big\}\,A^{(r,a)}(0) \qquad (38)$$

where the order of summation of R and r has been interchanged.

Equation (38) gives us, essentially, the desired expansion into the basis A. However, it is possible to factor Φ into a numerical coefficient

γ and an algebraic term Γ allowing us to write $\alpha\Phi = \alpha\gamma\Gamma$. We can, if we wish, collect the product $\alpha\gamma = \pi$ and write instead $\pi\Gamma$ where there are only two types of quantities in the summation; the numerical coefficient π and the algebraic coefficient Γ. We can also go the other way and collect together the factor to write $H = \alpha\Phi$. These various representations have their advantage depending on what we are interested in and how the quantities are to be obtained. We give below the definition of the coefficients that are necessary to effect these different representations.

Definition 8: Define

$$
\begin{aligned}
&z_j^k [R-1, r-1, 0] \equiv 1, \; z_j^k [R, r, a] \equiv y_j^k [R, r, a], \; a \neq 0 \\
&Y_i^k [R] \circ \equiv \sum_j f_{ijR}^{(k)} \theta_j^{R-1}. \\
&Y_i^k [R, k_o] \circ \equiv \sum_j g_{ik_oj}^{(k)}. \\
&Y_i^k [R, r_o, k_1, \ldots, k_s] \circ \equiv \sum_j f_{ijR}^{(k)} \theta_j^{r_o}.
\end{aligned}
\tag{39}
$$

then the generic $y_i^k [R, r, a]$ has as its realization the elementary polynomial $\Gamma_i^k [R, r, a]$ of rank R, order r, position a, degree s.

Definition 9: Define

$$
\begin{aligned}
&z_j^k [R-1, r-1, 0] \equiv 1, \; z_j^k [R, r, a] \equiv y_j^k [R, r, a], \; a \neq 0 \\
&Y_i^k [R] \circ \equiv \frac{(R+k)!}{(R-1)!}. \\
&Y_i^k [R, k_o] \circ \equiv \frac{(R+k)!}{(R-1)!}. \\
&Y_i^k [R, r_o, k_1, \ldots k_s] \circ \equiv \frac{(R+k)!}{(R-1)!}.
\end{aligned}
\tag{40}
$$

then the generic $y_i^k [R, r, a]$ has as its realization the polynomial weight $\gamma^k [R, r, a]$ of rank r, order r, position a, degree s, independent of i.

Definition 10: Define

$$z_i^k [R - 1, r - 1, 0] \equiv 1,\ z_j^k [R, r, a] \equiv y_j^k [R, r, a],\ a \neq 0$$

$$Y_i^k [R] \circ \equiv \frac{1}{(R - 1)!} \cdot$$

$$Y_i^k [R, k_o] \circ \equiv 1 \cdot \tag{41}$$

$$Y_i^k [R, r_o, k_1, \ldots, k_s] \circ \equiv \frac{1}{r_o!\ (\omega_1)! \ldots (\omega_\delta)!} \cdot$$

where ω_i is the number of times that (k_i, R_i, r_i, a_i) appears as a factor in $y_i^k [R, r, a]$. Then the generic y has as its realization the product coefficient $\pi^k [R, r, a]$ of rank R, order r, position a, degree s, independent of i and k.

Definition 11: Define

$$z_i^k [R - 1, r - 1, 0] \equiv 1,\ z_j^k [R, r, a] \equiv y_j^k [R, r, a],\ a \neq 0$$

$$Y_i^k [R] \circ \equiv \frac{1}{(R - 1)!} \sum_j f_{ijR}^{(k)}\ \theta_j^{R-1} \cdot$$

$$Y_i^k [R, k_o] \circ \equiv \sum_j g_{ik_oj}^{(k)} \cdot \tag{42}$$

$$Y_i^k [R, r_o, k_1, \ldots, k_s] \circ \equiv \frac{1}{r_o!\ (\omega_1)! \ldots (\omega_\delta)!} \sum_j f_{ijR}^{(k)}\ \theta_j^{r_o} \cdot$$

where ω_i is the number of times that (k_i, R_i, r_i, r_i) appears as a factor in $y_i^k [R, r, a]$. Then, the generic y has as its realization the generalized R.K. harmonics $H_i^{(k)}[R, r, a]$ of rank R, order r, position a, degree s.

In order to obtain the various representations of (37) using these definitions, we appeal to the following result.

Theorem 4:

Let the generic y be generated as $y = Yz_1 \ldots z_s$ where $Y = UV$ and $z_i = u_i v_i$ are permissible factorizations of Y and z_i, respectively. Then $y = y_1 y_2$ is a permissible factorization of y where $y_1 = Uu_1 \ldots u_s$,

$y_2 = Vv_1 \ldots v_s$ are the generations of the factors y_1 and y_2 provided the necessary commutation of the terms in $UVu_1v_1 \ldots u_sv_s$ can be carried out. The converse is also true. That is, given the factors y_1 and y_2 generated using the operators U and V, then $y = y_1y_2$ can be generated using the operator $Y = UV$ provided the necessary commutations can be carried out.

Proof: The proof is inductive. Starting with y [R, r, 1] of lowest rank, order, and degree, we show that the inductive process can be started and is, in fact, true for all degree zero terms. Equation (24) shows that if it is true for a given rank R, order r, degree s, then it is true for rank R + 1. Equation (25) will be true provided it is true for quantities of lower rank and order.

Using this result, we can write the coefficient of A in (37) as follows:

Theorem 5:

Let α and Φ be respectively a derivative harmonic and weighted polynomial of rank R, order r, degree s, position j; then we have that

$$\begin{aligned} \alpha^k_{Rrj} \ \Phi^k_i [R, r, j] &= \alpha^k_{Rrj} \ \gamma^k_{Rrj} \ \Gamma^{(k)}_i [R, r, j] \\ &= \pi^k_{Rrj} \ \Gamma^{(k)}_i [R, r, j] \\ &= H^{(k)}_i [R, r, j] \end{aligned} \tag{43}$$

where γ, Γ, π, and H are respectively the polynomial weights, elementary polynomials, product coefficients, generalized R.K. harmonics.

We now make use of our results to restate the definition of the generalized R.K. scheme, but now in terms of the harmonics of the basis A.

To define a generalized R.K. scheme for the solution of

$$D^p x = X \circ \xi, \ x \in R \to R^N, \ X \in R^{n \times p} \to R^n, \ \xi \in R \to R^{r \times p},$$

specify the rank q, the extent e, and the scheme definition

$$\xi_i^{(k)} = \sum_{j_1} \sum_{k_o} g_{ik_oj_1}^{(k)} \xi_{j_1}^{(k_o)} + \sum_{j_2} a_{ij_2}^{(k)} X(\xi_j) \tag{44}$$

where

$$j_1 \in S_{j_1},\ j_2 \in S_{j_2},\ i \in S_i$$

with

$$S_i \cup S_{j_1} \cup S_{j_2} \subset S = \{j \mid 0 \leq j \leq exq\}$$

$$k_o,\ k \in P = \{0, \ldots, p-1\}.$$

To determine the parameters of (44), choose a point within the extent and require that the corresponding ξ_i shall match the true solution $\xi(\theta_i)$ to a certain order $\bar{r}$. This leads to the parameter defining equations that can be written in the following equivalent forms.

$$\begin{aligned}
&\sum_{R=r}^{\infty} \sum_{j \in S_{Ra}} H_i^{(k)} [R, r, j] = \beta_{ra} \frac{\theta_i^{r+k}}{(r+k)!} \\
&\sum_{R=r}^{\infty} \sum_{j \in S_{Ra}} \pi_{Rrj}^{k} \Gamma_i^{(k)} [R, r, j] = \beta_{ra} \frac{\theta_i^{r+k}}{(r+k)!} \\
&\sum_{R=r}^{\infty} \sum_{j \in S_{Ra}} \alpha_{Rrj}^{k} \gamma_{Rrj}^{k} \Gamma_i^{(k)} [R, r, j] = \beta_{ra} \frac{\theta_i^{r+k}}{(r+k)!} \\
&\sum_{R=r}^{\infty} \sum_{j \in S_{Ra}} \alpha_{Rrj}^{k} \Phi_i^{(k)} [R, r, j] = \beta_{ra} \frac{\theta_i^{r+k}}{(r+k)!}
\end{aligned} \tag{45}$$

where $\ell + 1 \leq r \leq \bar{r}$

$$k \in P$$

$$a \in S_r = \{a \mid A^{(r,a)} \text{ has order } r\}$$

$$S_{Ra} = \{j \mid y_i^k [R, r, j] \text{ has rank } R, \text{ order } r \text{ and corresponds to } A^{(r,a)}\}.$$

In the layout of the interval we have assumed that we proceed in a fashion identical to that of Chapter III using major points and minor points,

the meaning of which are well defined there. Here, all the θ_i are measured from the origin. It will be recalled that the expansions of the major points were assumed "known". For our present development, we use property 1 which, when suitable attention is paid to the re-indexing that has taken place, becomes

$$\begin{aligned} f^{(k)}_{ijR} &= 0 \quad , \; i \neq j \\ f^{(k)}_{iiR} &= \frac{(R-1)!}{(R+k)!}\, \theta_i^{k+1} \end{aligned} \tag{46}$$

when $\zeta_i(1) = \xi(\theta_i)$.

Along with these parameter defining equations, we have conditions equivalent to those of Theorem 10, Chapter III, which arrise when a minimum order $\xi_i^{(k)} - \xi^{(k)}(\theta_i) = O(h^{\ell+1+k})$ is required. These are given as

Conditions A: Let $\theta_i = \theta_i(t)$ and define the polynomials

$$P_i^k(t) = \sum_{j_1} \sum_{k_o} g^{(k)}_{ik_oj_1} \frac{\theta_{j1}^{\ell+k_o}}{(\ell+k_o)!} + \sum_{j_2} a^{(k)}_{ij_2} \frac{\theta_{j_2}^{\ell-1}}{(\ell-1)!} - \frac{\theta_i^{\ell+k}}{(\ell+k)!}$$

then in order that $\xi_i^{(k)} - \xi^{(k)}(\theta_i) = O(h^{\ell+1+k})$, it is necessary and sufficient that

$$D^r P_i^k(t) \equiv 0 \tag{47}$$

for

$$r = 0, 1, \ldots, \ell + p - 1.$$

$$k \in P = \{0, \ldots, p-1\}, \; i \in S = \{0, \ldots, exq\}.$$

Given that equations (47) and (45) are satisfied, then the local truncation error is

$$\xi_i^{(k)} - \xi^{(k)}(\theta_i) = \sum_{r=\bar{r}}^{\infty} \sum_{a \in S_r} \Big\{ \sum_{R=r}^{\infty} \sum_{j \in S_{Ra}} H_i^{(k)}[R, r, j] - \beta_{ra} \frac{\theta_i^{r+k}}{(r+k)!} \Big\} A^{(r,a)}(0) \tag{48}$$

or any of the equivalent representations obtained by replacing H by its factorizations.

In order that these constructions can easily be carried out, we have tabulated in Tables VIII-XII of Appendix I the various realizations of the generic y that have been used here. Since the generalized R.K. harmonics are easily derivable from the quantities, these have been omitted. A detailed explanation of these tables is given in that Appendix. Their use is illustrated later in Chapter VI where various examples are treated.

At the beginning of this chapter, we indicated that there is a complete equivalence between the present global approach and the previously developed substitutive approach to the non-linear parameter defining equations. At first sight this is not apparent; however, the connection can be obtained quite easily. In Chapter III, each individual approximation ξ_i has harmonics, which we shall call approximation harmonics that are obtained by carrying the indicated substitutions and linear combinations. The substitutions are, in practice, effected by means of a substitution table the elements of which can be generated by a realization of a formally defined generation scheme. We note that this scheme is not used in obtaining the harmonics, only the elements of the substitution table. In our present work, we have obtained quantities, the generalized R.K. harmonics or any of their factorizations, using a formally defined generation scheme (a scheme which can also generate the substitution table as one of its realizations) and a linear combination of the appropriate R.K. harmonics yields the approximation harmonics. Now, it is possible to obtain a realization of this latter scheme for which the generated elements are the approximation harmonics. In the process of arriving at the appropriate definition for the quantities in Definition 3, we shall

see the connection between the two different approaches that have been used to obtain the parameter equations.

We know that we can write any approximation ξ_i as

$$\xi_i^{(k)} - u(\theta_i) = \sum_{j,k_o} g^k_{ik_oj} [\xi_j^{(k)} - u_j^{(k)}(1)] + \sum_j a^k_{ij} [X(\xi_j) - R_j] \quad (49)$$

and further that

$$\xi_i^{(k)} - u(\theta_i) = \sum_{r=\ell+1}^{\infty} \sum_{a\in S_r} \alpha^k_{jra} A^{(r,a)}(0). \quad (50)$$

If we substitute (50) into (49) and use the substitution harmonics of Definition 5, Chapter III, then we obtain

$$\xi_i^{(k)} - u(\theta_i) = \sum_{r=\ell+1}^{\infty} \left[\sum_{k_o,j} g^k_{ik_oj} \alpha^{k_o}_{jr1} + \sum_j a^k_{ij} \frac{\theta_j^{r-1}}{(r-1)!} \right] A^{(r,1)}(0)$$

$$+ \sum_{r=\ell+2}^{\infty} \sum_{a\in S_r} \left[\sum_{k_o,j} g^k_{ik_oj} \alpha^{k_o}_{jra} + \sum_j a^k_{ij} \frac{\theta_j^{r_o}}{r_o!} \frac{(\alpha^{k_1}_{jr_1a_1})^{\omega_1}}{\omega_1!} \cdots \right.$$

$$\left. \frac{(\alpha^{k_\delta}_{jr_\delta a_\delta})^{\omega_\delta}}{\omega_\delta!} \right] A^{(r,a)}(0) \quad (51)$$

where ω_i and δ have the meaning assigned to them in the definition of the derivative harmonics.

Now, if we substitute for α^k_{jra}, its representation as

$$\alpha^k_{jra} = \sum_{R=r}^{\infty} \sum_{j\in S_{Ra}} H_i^{(k)} [R, r, j] \quad (52)$$

and collect the coefficients of $A^{(r,a)}$ to obtain the approximation harmonics α^k_{ira} corresponding to the approximation ξ_i^k, we shall find that α^k_{ira} also has the representation (52). This is, indeed, the obvious result that must be true if the derivations are correct; however, this allows us to see that for each approximation that we construct, the constructed harmonics are identical to those obtained using the generalized R.K. harmonics and that

by a rather simple extension of the definition of the substitution harmonics, we can obtain from the generic generator a realization which will give us these harmonics. This is done in

Definition 12: Define

$$z_i^k [R - 1, r - 1, 0] \equiv 1,$$

$$z_i^k [R, r, a] \equiv \sum_{k_o \in P} y_i^k [1 + R + k_o, r, a] + y_i^k [R, r, a], \; R = r, \; a \neq 0$$

$$z_i^k [R, r, a] \equiv 0, \; R > r, \; a \neq 0$$

$$Y_i^k [R] \circ \equiv \sum_j f_{ij}^k \frac{\theta_j^{R-1}}{(R-1)!} . \tag{53}$$

$$Y_i^k [R, k_o] \circ \equiv \sum_j g_{ik_oj}^k .$$

$$Y_i^k [R, r_o, k_1, \ldots, k_s] \circ \equiv \frac{1}{r_o!} \frac{1}{(\omega_1)! \ldots (\omega_\delta)!} \sum_j f_{ij}^k \theta_j^{r_o} .$$

where ω_i is the number of times that (k_i, R_i, r_i, a_i) appears as a factor in $y_i^k [R, r, a]$. Then the generic z of rank R = order r has as its realization the approximation harmonic α_{jra}^k.

Associated with this definition, we have

Theorem 6

If ξ_i is an approximation defined by a generalized R. K. scheme, then

$$\xi_i^{(k)} = \sum_{r=\ell+1}^{\infty} \sum_{a \in S_r} \alpha_{ira}^k A^{(r,a)}(0)$$

where the α are approximation harmonics. This, in particular, includes the solution values $\xi(\theta_i)$.

Proof: The proof of this result follows by induction from (51) where we proceed as we have done many times before to substitute the

expansion for $\xi_i^{(k)} - u(\theta_i)$ into the right side and isolate the general term. This will lead to coefficients which are those of Definition (12).

We note that there is little difference between the definition of the generalized R.K harmonics and those we have just defined. There the generic y turned out to be the quantity of interest, while here the generic z are the items we want.

The definition of these quantities is rather abstract and is given in the foregoing fashion to enable us to obtain all the harmonics using the same formal pattern of generation. However, they have a simple pattern that can be obtained directly from the approximation and it is convenient, in practice, to have this explicitly displayed. Thus, given the approximation (49), we simply note that the harmonics $\alpha^k_{ira} = z^k_i\,[r,r,a]$ are the coefficients of (51). We may now, if we wish, use (52) in (45) and (48).

We thus see that we can, in general, tabulate the equations with no specific reference to the index sets and then the particular realization of anyone scheme arrises by specifying the appropriate index sets. How this actually is carried out will become clear when we treat various examples in Chapter VII. For the present, we restrict ourselves to mentioning a few rather obvious facts. It is evident that the choice of origin is immaterial; conditions A are origin independent and the results up through the principal error term should be origin independent. We lack a general theorem concerning the latter fact; however, the results can be directly verified since the equations are polynomials in θ. The choice of ℓ depends on the desired method; for self starting methods $\ell = 1$ while for finite difference methods of order r, $\ell = r - p$. For mixed methods of a given order r, then $m = r - p - \ell$ is a measure of the complexity of the set of

equations (45) that must be solved. We are rather vague here about the precise definition of order, but, for the present, it suffices to take p = 1 and for higher p we shall be more precise later on when dealing with examples.

A further remark is that when comparing our present results with those obtained by the constructive substitution approach, we must remember that we are essentially dealing with what was then called the backward translation case. We know, however, that if our work is carefully arranged, we should be able to make a direct comparison of the results.

We should also say a few words about the infinite sum on rank R. If the R.K. harmonics H are generated (we have seen that we can instead, if we wish, generate the approximation harmonics), then while it is true that we cannot a-priori state exactly when the infinite sum or the rank R will terminate, the sum can be drastically limited provided we limit slightly the class of methods we wish to investigate. To be more precise, note that for any given scheme the appearance of a term of rank $R > r$ implies that the coefficient g appearing in the scheme are non-zero. We know that for purely R.K. schemes all g can be considered as identically zero. Hence, we need never consider $R > r$ for these cases. Incidentally, for $R = r$ and purely R.K. schemes our results should agree identically, when $\ell = 0$, $p = 1$, with those of Butcher[1] and with $\ell = 1$, $p = 1$ with those of Ceschino - Kuntzmann[2]. For Runge-Kutta schemes with memory where only function evaluations are used, we note that again we need never consider cases with $R > r$. However, this is a severe restriction and we do not limit ourselves to that extent. Instead, for convenience in presenting some complete tables, we limit ourselves to scheme whose parameters are to be determined with the assumption that the major points have "known" expansions whose harmonics are the Taylor harmonics. That is, we consider

major points to have approximations $\xi_i = \xi(\theta_i)$. This then implies that $g^k_{ik_0j} \equiv 0$. This, in effect, simply says that in determining the approximation ξ_i, we shall consider ξ_j to be a true solution value if $j = n \times q$ where q is the rank of the method. Since, in practice, the major points ξ_j all have an order equal to that of ξ_i, the determination of the parameters will not be affected. In contrast to this, we consider $\xi_j = \xi(\theta_i) + \varepsilon_i$ where ε_i is an error term. We know that this case can be easily treated provided we "know" the appropriate expansion of ε_i and therein lies the difficulty. We have indicated in the substitution case how this difficulty can be treated using undetermined parameter expansions. This approach is still valid if we generate the approximation harmonics using Definition 12, but it becomes more difficult to apply directly using the R. K. harmonics H. Since it will not, in general, change the parameter defining equations, we consider $\xi_j = \xi(\theta_j)$. The limitation imposed here does, indeed, become a restriction if we are concerned with global error terms and correct starting values of the scheme, but we shall say more about that later.

With this restriction we are able to forget about all functions for those cases where there appears two or more g coefficients. Referring to Table VIII of Appendix I, we have eliminated all cases where there are two or more E factors. This allows the tables to be reduced to a reasonable working size.

To illustrate with an example, consider

$$\xi_3 = \xi_1 + \xi_2 + \cdots$$

where $g_{31} = g_{32} = 1$ and all other g are zero. An examination of the meaning of the higher powers of E shows us that we will have terms such as $g_{31}\, \varepsilon_1 \cdots$, but since ξ_1 is an exact value $\xi(\theta_1)$ we have that $g_1 \cdots = 0$.

These ideas become clearer when specific examples are treated. In short, it turns out that we need only consider terms that contain one or less factor E when treating generalized R.K. methods in which we determine the parameters using one major step. If we use two major steps, consider the case of the truncation error after two steps, then we need terms with two or less factors and, in general, the pattern persists. Should one wish to utilize the tables presented for these more complicated schemes, then these tables must be extended. We comment that if these quantities are generated, then there is no problem; we get all the quantities we need simply by specifying the index sets and the "known" expansion at major points.

Having limited the table of functions needed, it turns out that we can further limit the sum on R. The rank R can be written as $R = r + 1 + k_1$ when $k_1 \in \overline{P} \subset P = \{0, \ldots, p - 1\}$. Thus, we need only consider those k_1 that appear explicitly in $X \circ \xi$.

V. UTILIZATION OF ERROR HARMONICS IN GENERALIZED RKF SCHEMES

In Chapter V, we present results that are, to a large extent, simply a re-interpretation of the earlier work of Chapter III. It is shown that if we evaluate the differentials not at $u(0)$, but, instead, at $\xi(0)$ where ξ is an exact solution to $D^p x = X \circ \xi$, then many of the harmonics that were previously non-zero become zero and no new non-zero harmonics are introduced. We thus, in a sense, have found a canonical set for a basis. We are lead to this set by considering the errors $\varepsilon_i = \xi_i - \xi(\theta_i)$ of the generalized scheme. These errors are defined in Definition 1. In order to carry out the required expansions, we define, Definition 2, error differentials E (the canonical basis) and then proceed to show that most of the work of Chapter III has a parallel development in terms of these functions. We can calculate the derivatives of E, Theorem 1. The derivatives of x can be expanded into the set E, Theorem 2, and, in fact, those harmonics are mostly zero. Again, it is possible to define derivative, substitution, multiplication, and translation harmonics. In fact, it turns out that the substitution and multiplication harmonics are identical to those previously defined.

Once again, we are able to extend the definitions to the generalized RKF schemes. This is given in Definition 5 which is merely a restatement of that given in Chapter III, but now in terms of the errors ε and the error differentials E. This allows us to show that there is relatively little to be done before the program RKMI can construct schemes in the coefficient space of the error differentials. Finally, we show, Theorem 6, that there is essentially no distinction to be made between the approximation harmonics α given in Chapter IV and the error

harmonics e developed here. That is, a table of approximation harmonics is also a table of error harmonics.

In the work that has been presented in Chapters II - IV, extensive use has been made of the fact that any solution ξ of the differential equation $D^p x = X \circ \xi$ can be represented as $\xi = u + v$ where the first term is a finite number of terms of the Taylor's series expansion of ξ and v is the remaining part of that series. This has caused all of our functions, in particular, the differentials A and weighted differentials W, to be evaluated at u. For example, we have $D(X \circ u)(0)$. This has been a fortunate choice in some respects. In particular, it has forced us to find an expansion formula for the derivatives $D^r(X \circ \xi) = D^r(X(u + v))$ which has been the prime source of inspiration in obtaining the definitions for the differentials which in turn has lead to a definition of the generic y. However, it also leads to results that do not necessarily compare directly with other work. Our previous work can be directly compared with that presented by Ceschino - Kuntzmann[2] and for the $p = 1$, $\ell = 1$ case does agree with their results; but the work presented by Butcher[4,8] has caused us to ask, what is the direct connection between his results and those derived here? The overly simple answer is that the point of evaluation of the derivatives is different from our choice, so there must be a difference in our error terms. However, what about the equations that determine the undetermined parameters? If we look closely, we can identify all the derivatives that he uses and find the correspondence between these functions and ours, but the number of equations that he sets forth to solve are a subset of those that we would obtain. Also, his equations contain error terms of the approximations; ours never do. In short, his results look simpler and the question

arises: what caused the simplification? This is an important point because:

1) For higher order RKF schemes in which the difference between the order of accuracy of the minor and major points is more than three or four, the nonlinear parameter equations soon become quite complicated and, thus, any simplification in the results will aid in understanding and solving these equations.

2) The storage requirement of RKMI can easily become excessive and it is necessary that the problem be simplified or the program modified to take advantage of certain properties of the parameter equations.

We shall present below a development that connects our work to that of Butcher, simplifies the equations, and, at the same time, raises some interesting questions. To do this, we shall use the results of Chapters III and IV We shall, however, use the notation of Chapter IV.

In Chapters II - IV, we have worked with approximations ξ_i which were constructed in a well-defined manner and we obtained our parameter equations by requiring that these approximations agree to a certain order of accuracy with the true solution $\xi(\theta_i)$. We shall, in this chapter, work with the errors ε_i which are a measure of how well the approximations and the true solution agree. These are defined as:

Definition 1: The error ε_i is said to be a generalized R.K scheme error if, and only if

$$\varepsilon_i^{(k)} + v^{(k)}(\theta_i) = \sum_{j,k_o} g_{ik_oj}^{(k)} [\varepsilon_j^{(k_o)} + v^{(k_o)}(\theta_j)] + \sum_j a_{ij}^{(k)} [X(\xi(\theta_j) + \varepsilon_j) - R_j] \tag{1}$$

where $\xi(\theta_j)$ is an exact solution of $D^p x = X \circ \xi$

$$v^{(k)}(\theta_i) = \sum_{r=k+\ell+1} D^{p-1-k+r}x(0) \ \frac{\theta_i}{r!}$$

$$R_j = \sum_{r=o}^{\ell-1} D^{p+r}x(0) \ \frac{\theta_j^r}{r!} \tag{2}$$

and ε_j is an error obtained in the same fashion. We immediately have that

$$\xi_i = \varepsilon_i + \xi(\theta_i) \tag{3}$$

which follows directly from (1) upon substituting (3) into (1), recalling that $\xi = u + v$, and referring to equations (11), Chapter III, or equation (9), Chapter IV. We shall assume that equation (11') of Chapter III always holds. That is, condition A is in effect for all valid choices of the parameters in (1).

We now define differentials exactly as was done in Definition 2 of Chapter III <u>except</u> we replace u by ξ. We give that definition as

<u>Definition 2</u>: Define

$$z\,[r-1,\,0] \equiv 1,\ z\,[r,\,a] \equiv y\,[r,\,a],\ a \neq 0$$

$$Y\,[r]_\circ \equiv C_{N_o}^{(r-1)} \tag{4}$$

$$Y\,[r_o,\,k_1,\,\ldots,\,k_s]_\circ \equiv C_{N_oN_1\ldots N_s}^{(r_o,k_1,\ldots,k_s)}.$$

where $$C_{N_o}^{(r_o)} = D^{r_o}(X \circ \xi)_{N_o}$$

$$C_{N_oN_1\ldots N_s}^{(r_o,k_1,\ldots,k_s)} = D^{r_o}(D_{L_1\ldots L_s} X \circ \xi)\, I_{L_1N_1}^{(k_1)}\ldots I_{L_sN_s}^{(k_s)}$$

and $\xi = (D^{p-1}x, \ldots, D^o x)$ is an exact solution to $D^p x = X \circ \xi$.

Then the generic y [r, a] has as its realization the error differentials E [r, a].

We have given these differentials a different name, E. instead of A, because they really are different functions since they contain a composition with the function ξ, instead of u as was the case with the differentials A. However, the results proven using the differentials A carry over directly to the E and the proof of these results is carried out in the same manner. We shall, therefore, only give the results that we wish to use and the reader can refer to the appropriate previous work if he wishes to carry out the proofs.

Theorem 1

The j-th derivative of the differential E of order r, degree s is a linear combination with non-negative integral coefficients of the differentials E of order r + j, degree s.

Proof: See Theorem 2, Chapter III.

Theorem 2

$D^{p+r}x(0) = B^r$ of order r + 1 is a linear combination of differentials E of order r + 1 evaluated at 0. In fact, the only non-zero coefficients are those of the degree 0 terms.

Proof: Notice that $D^{p+r}x(0) = D^r(X \circ \xi)(0)$ which is the differential of degree zero, order r + 1.

If we compare this with Theorem 3 of Chapter III, we immediately see that we have a much simpler case here. Previously, the coefficients were greater than zero for all the order r + 1 terms We went to a considerable amount of trouble to calculate the derivative harmonics in Chapter III. They are trivial to obtain here; they are all zero except the degree 0 terms which have coefficient 1.

Definition 3: Define

$$z\,[r-1,\,0] \equiv 1,\quad z\,[r,\,a] = y\,[r,\,a],\quad a \neq 0$$

$$Y\,[r]_\circ \equiv 1$$

$$Y\,[r_o,\,k_1,\,\ldots,\,k_s]_\circ = 0.$$

then the generic $y\,[r,\,a]$ becomes the derivative harmonic β_{ra}.

These results allow us to expand $v(\theta_i)$ into the error differentials as

$$v^{(k)}(\theta_i) = \sum_{r=\ell+1}^{\infty} D^{p+r-1}x(0)\ \frac{\theta_i^{k+r}}{(k+r)!} = \sum_{r=\ell+1}^{\infty} \frac{\theta_i^{k+r}}{(k+r)!}\ E^{(r,0)}(0). \quad (5)$$

It turns out that the operations of substitution and multiplication have the same form in the coefficient space of the error differentials E as they did in the coefficient space of the differentials. We, thus have

Theorem 3 (Substitution)

Let

$$z_N = X(\xi(\theta) + T_L) - R_N$$

where

$$R_N = \sum_{r=o}^{\ell-1} D^r(X \circ \xi)(0)$$

$$T_L = \sum_{r\in S}\ \sum_{a\in S_r}\ \sum_{k\in \overline{P}}\ \alpha_{rka}\ I_{LN}^{(k)}\ E_N^{(r,a)}.$$

Then

$$z_N = \sum_{r\in S}\ \sum_{a\in S_r}\ \beta_{ra}\ E^{(r,a)}$$

where β_{ra} are the substitution harmonics corresponding to T_L. These harmonics are defined exactly as in Definition 5, Chapter III.

Proof: The proof is carried out exactly as was done for Theorem 6, Chapter III, except we start with $z = X(\xi(\theta) + T)$ and we note that

$$D^r(X \circ \xi)(0) = D^r(X \circ u)(0) \text{ for } 0 \leq r \leq \ell - 1.$$

In a similar fashion, we have

Theorem 4 (Multiplication)

Let T_L and S_L be defined as

$$T_L = \sum_{r \in S} \sum_{a \in S_r} \sum_{k \in \overline{P}} \gamma_{rka} \; I_{LN}^{(k)} \; E_N^{(r,a)}$$

$$S_L = \sum_{r \in S} \sum_{a \in S_r} \sum_{k \in P} \alpha_{rka} \; I_{LN}^{(k)} \; E_N^{(r,a)}.$$

Let

$$J_{NL} = D_L X(\xi(\theta) + T_L) \text{ and } W_N = J_{NL} \; S_L.$$

Then

$$W_N = \sum_{r \in S} \sum_{a \in S_r} \beta_{ra} \; E_N^{(r,a)}$$

where β_{ra} are the multiplication harmonics corresponding to T_L and S_L as defined in Definition 7, Chapter III.

Proof: See Theorem 7, Chapter III The starting point is, however, now $D_L \; X(\xi(\theta) + T_L)$.

It will be recalled that one of our big difficulties has been to obtain the translation harmonics that allowed us to change the origin of evaluation of the differentials. These can now easily be obtained using Theorem 5, Chapter III, which gives the derivatives of A. In that theorem, we need simple replace A by E to obtain the derivatives of E. However, it is easier, in practice, to obtain these quantities by actually carrying out the differentiation. This was indicated in Chapter III where we gave a few of the derivatives of A; again, we need simply replace A by E.

Once we know the derivatives of E, we can obtain the corresponding translation harmonics as follows:

Let

$$\overline{E} = E^{(\overline{r},\overline{a})} = \{\overline{r}_o, \overline{k}_1, \ldots, \overline{k}_s, E^{(\overline{r}_1,\overline{a}_1)} \ldots E^{(\overline{r}_{\overline{s}},\overline{a}_{\overline{s}})}\}$$

$$E = E^{(r,a)} = \{r_o, k_1, \ldots, k_s, E^{(r_1,a_1)} \ldots E^{(r_s,a_s)}\} .$$

Then, we have

Definition 4: The translation harmonics $\gamma_{ra}^{(\overline{r},\overline{a})}$ corresponding to $\overline{E}$, E are defined as

i) $s < \overline{s}$ $\qquad \gamma(t) = 0$

ii) $s = \overline{s}$

a) $(\overline{k}_1, \ldots, \overline{k}_s) \neq (k_1, \ldots, k_s)$ $\qquad \gamma(t) = 0$

b) $(\overline{k}_1, \ldots, \overline{k}_s) = (k_1, \ldots, k_s)$

1) $r_o < \overline{r}_o$ $\qquad \gamma(t) = 0$

2) $\overline{r}_o \leq r_o \leq \overline{r}_o + j$

a) $r_i \notin [\overline{r}_i, \overline{r}_i + j]$ $\qquad \gamma(t) = 0$

b) $r_i \in [\overline{r}_i, \overline{r}_i + j]$

$$\gamma(t) = \frac{(\omega_1)! \ldots (\omega_g)!}{(\delta_1^{(1)})! \ldots (\delta_{\sigma_1}^{(1)})! \ldots (\delta_1^{(g)})! \ldots (\delta_g^{(g)})!} \cdot \frac{t^{r_o-\overline{r}_o}}{(r_o-\overline{r}_o)!} \cdot \overline{\gamma}_{r_1,a_1}^{(\overline{r}_1,\overline{a}_1)} \ldots \overline{\gamma}_{r_s,a_s}^{(\overline{r}_{\overline{s}},\overline{a}_{\overline{s}})}$$

where

$$\overline{\gamma}_{r_i,a_i}^{(\overline{r}_i,\overline{a}_i)} = \gamma_{r_i,a_i}^{(\overline{r}_i,\overline{a}_i)}(t)/t^{r_i-\overline{r}_o} \quad \text{is independent of } t$$

and

$$S_i = \{(r_j, a_j) \mid k_j = k_i,\ 1 \leq j \leq s = \overline{s}\} =$$

$$\{(r_{i_j}, a_{i_j}) \mid j = 1, \ldots, \omega_i\}$$

$$\delta_j^{(i)} = \text{the repetition factor of } (r_{i_j}, a_{i_j}) \in S_i$$

$$\omega_i = \sum_{i=1}^{\sigma_i} \delta_j^{(i)}$$

$$S_o = \{(r_i, a_i) \mid i = 1, \ldots, s\} = \bigcup_{i=1}^{g} S_i$$

iii) $s > \bar{s}$ $\qquad\qquad \gamma(t) = 0$

Once again we can state the translation results as

Theorem 5

Let the differentials E be sequentially indexed as $E^{(j)}$. Let the differentials $E^{(i)}(\delta, 0)$ and $E^{(j)}(\delta, 0)$ be given along with their associated translation harmonics γ_j^i. Then

$$E^{(i)}(\delta, 0) = \sum_{j \in S_j} \gamma_j^i \; (\delta - \zeta)\, E^{(j)}(\zeta, 0), \; i \in S_i = S_j \qquad (6)$$

when $\delta - \zeta$ corresponds to the interval of translation and S_j = {set of all error differentials}.

This is the same as Theorem 9, Chapter III, and the corollary stated there giving the orthogonality conditions on γ is also applicable here.

Proof: We write $E^{(\bar{r},\bar{a})}(t) = \sum_{j=o}^{\infty} D^j(E^{(\bar{r},\bar{a})})(0) \frac{t^j}{j!}$ and then use Theorem 5 of Chapter III, as applied to error differential E, to obtain the expansion of $D^j(E^{(\bar{r},\bar{a})})(0)$. The reader will see that this leads directly to the harmonics given in Definition 4 above.

We have purposefully stated Definition 4 in the same format as was used to define the translation harmonics in Definition 9, Chapter III.

It is seen that there is a tremendous simplification when we use the error differentials E. We also note that Properties 1 and 2 are still true.

We, thus, can carry out the construction of the scheme using the error differentials E and we will have the non-linear parameter defining equations as error harmonics. In Chapter III, we devoted considerable space to explaining how that process was to be effected. There is no need to repeat it here; the overall approach is the same when using the functions E as a basis. However, it is important that Definition 10, Chapter III, be correctly interpreted and we restate it here. Remember, this definition was given in such a manner that it could be interpreted either in terms of points in $R^{n\times p}$ or as coefficients in the coefficient space of the differentials. It, therefore, is this definition that is reflected in the program RKMI. Such will also be the case here and it will, thus, be easy to see what has to be done to utilize error harmonics in RKMI.

Definition 5: The error ε_i is said to be obtained by means of a generalized Runge-Kutta-Frey type integration method (RKF) if, and only if

$$\varepsilon_i^{(k)} + v^{(k)}(\theta_i) = \sum_{j,k_o} g_{ik_oj}^{(k)} \left[\varepsilon_j^{(k_o)} + v^{(k_o)}(\theta_j)\right] + \sum_j a_{ij}^{(k)} \eta_j$$

where the approximators η_N are defined to be either

$$\eta_i = X(\xi(\theta_i) + \varepsilon_i)$$

or as $\eta_i = J_j S_i$

with $J_i = D\, X(\xi(\theta_i) + \varepsilon_i)$

and the sums S are defined to be

$$S_i^{(k)} = \sum_j \bar{g}_{ik_oj}^{(k)} [\varepsilon_j^{(k_o)} + v^{(k_o)}(\theta_j)] + \sum_j \bar{a}_{ij}^{(k)} \eta_j$$

constructed using the errors ε and approximators η. The index sets are such that any element lies in the set $S_o = \{0, \ldots, e \times q\}$ where e is the extent and q the rank of the scheme.

This is nothing more than the restatement of the previous definition by using $\xi_i = \varepsilon_i + \xi(\theta_i)$. There is a complete equivalence and, thus, the discussion of Chapter III pertains also to this work. In particular, see the summary of that chapter.

We point out that the great saving with respect to storage in the use of RKMI comes from working with ε_i, or $\varepsilon_i + v(\theta_i) = \xi_i - u(\theta_i)$; we have lots of zero harmonics. For any major point, we have that $\varepsilon_i = 0$ for all harmonics, and $v(\theta_i)$ has non-zero harmonics for only the degree zero terms. We also note that in the coefficient space, $u(\theta_i)$ has all zero harmonics, so, in reality, $\varepsilon_i + v(\theta_i) = \xi_i$ and we are able to use Definition 10, Chapter III, as it stands provided we:

1) Think of this as being in the coefficient space of error differentials, and
2) Remember to subtract off $v(\theta_i)$ from ξ_i before performing a substitution.

To state it quite plainly, if we use the translation and derivative harmonics presented in this chapter, and if we always substitute $\xi_i - v(\theta_i)$ instead of ξ_i, then the work of Chapter III and the program RKMI which arises from Chapter III can be interpreted directly in terms of error differentials and error harmonics as presented here. We see that the only modification necessary to RKMI is the ability to perform the subtraction $\xi_i - v(\theta_i)$ before substituting. The implementation of this is, however, relatively easy; although not completely trivial since

we probably would not want to store separate v's for each θ_i. With regard to this, the reader should refer to how this problem was solved for the translation tables (remember, they depend on the translation interval). The description of procedure translate will be found in Appendix II.

We now turn to obtaining the harmonics directly. It will be recalled that using the differentials A as a basis, the approximation harmonics could, with the aid of the substitution table, be directly obtained for generalized RK schemes and that this was done at the end of Chapter IV. This is also true when using the error differentials E. To distinguish these harmonics from those previously defined, we shall refer to them as error harmonics. This seems natural since the local error can be expressed as $\varepsilon = \xi - v$ where ξ has as harmonics those which we call error harmonics and v has mostly zero harmonics. Thus, in reality, most of the error harmonics are, indeed, the harmonics of the local error when expanded into the space of error differentials.

In fact, referring to Equations (49), (50), and (51) of Chapter IV, we see that if we interpret the results using the error differential E, the equations are still true. We, thus, arrive at

Theorem 6

The approximation harmonics defined in Definition 12, Chapter IV, can also be interpreted as error harmonics e provided we change the previous definition (Definition 12, Chapter IV) to read

$$Y_i^k [R, r_o, k_1, \ldots, k_s]_\circ \equiv \left(\frac{1}{r_o!} \frac{1}{(\omega_1)! \ldots (\omega_\delta)!} \sum_j f_{ij}^k \theta_j^{k_o} \right) \cdot \left[-\beta_{jr_1 a_1}^{k_1}, \ldots, -\beta_{jr_s a_s}^{k_s} \right]_\circ$$

where

$$\left[-\beta^{k_1}_{jr_1a_1}, \ldots, -\beta^{k_s}_{jr_sa_s}\right] \circ \left(z^{k_1}[r_1, a_1], \ldots, z^{k_s}[r_s, a_s]\right) \equiv$$

$$\prod_{i=1}^{s} \left(z^{k_i}[r_i, a_i] - \beta^{k_i}_{jr_1a_1}\right)$$

and $$v^k(\theta_j) = \sum_{r=\ell+1}^{\infty} \sum_{a \in S_r} \beta^k_{jra} E^{(r,a)}(0).$$

If ξ_i is an approximation defined by a generalized R.K. scheme, then

$$\xi_i = \sum_{r=\ell+1}^{\infty} \sum_{a \in S_r} e^{(k)}_{ira} E^{(r,a)}(0)$$

when the e are error harmonics. In particular, this trivially includes the solution values $\xi(\theta_i)$ for which all e_i of degree greater than zero are identically zero.

Thus, we see that much of our work carries over directly to expressing the problem in terms of the local error between the true solution and the constructed approximation. There is an advantage in such a presentation, and it is this set of functions that leads to the same results as Butcher. All of his derivatives are evaluated at $\xi(\theta_i)$ an exact solution point. However, we should note that while it seems natural to define and use the set of function {A} as a basis, it is not at all obvious that the set of functions {E} are the appropriate basis. There are just too many zero harmonics for the derivatives in the latter set of functions. Thus, the evaluation of X at $u(\theta)$ has, through the use of formula (21)(Chapter III), lead us to a definition for the set of differentials {A} that have in turn suggested the correct definition for the set of error differentials {E}. We have, in a sense, found a canonical set of functions to use for a basis. The set is certainly

large enough; however, while the set {A} is a minimal set (we need all the functions) and we made it that way by restricting all indices to the normal index set, it is in no way obvious that the set {E} is minimal. It probably is; it would be nice to show this.

Also, there is the question of whether the work of Chapter IV applies to this set of functions. How to formulate the definition of the approximation ζ so that we are in the coefficient space of the error differentials is an open questions; those definitions that the author has tried so far lead to difficulties that will not be discussed here.

VI. GENERATION OF GENERALIZED RKF SCHEMES BY MEANS OF AN ALGOL 60 PROCEDURE

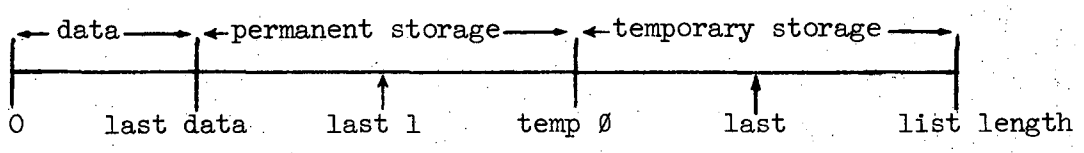
Chapter VI is devoted to a description of various parts of the program RKMI that carries out the successive substitutions. The author is of the opinion that it is impossible to "describe" a program in detail; the user must do this for himself by means of the source listings. On the other hand, it seems to be just as impossible to read a source listing if there are no hints as to why things are done the way they are and what certain quantities mean. We try to give here, in this chapter, a guide to the rationale behind the construction of RKMI. We present a rather detailed, and yet, at the same time, rather sketchy description of how the program goes about performing the necessary operations. The list structure is explained; the procedures that create, store, and fetch from the list are described; and an attempt is made to describe how the operations of addition and multiplication are performed using lists and how the ability to perform these elementary operations on the lists enables us to carry out the successive substitutions and arrive at a scheme and its associated parameter defining equations.

This chapter is meant to be read in conjunction with Appendix II. Ideally, the reader has before him a complete listing of whatever procedure is being discussed along with the appropriate variable declarations and procedure descriptions. Since this is, indeed, rather ideal, it is hoped that those who want, or need, to know more about the how and why of RKMI will find in this chapter and in Appendix II enough material for them to answer their own questions. There is no attempt made here to be exhaustive; as we have said, we believe it to be impossible and actually unprofitable.

The procedures that follow below will not necessarily appear in the order that they appear in the program. Instead, we shall present them in an order which illustrates how the program carries out the successive substitutions.

In order to carry out the algebraic manipulations that are required, a list structure is developed and procedures are built that perform certain elementary operations on the list. The list itself is stored in the array V [0: list length]. This array is divided into two parts; the first being a permanent storage section from [0: temp ø - 1], and the second being a temporary storage part from [temp ø: list length]. Temporary storage is essentially a scratch area and is continually overwritten starting at the temporary storage origin temp ø, whereas the permanent part is never overwritten except to restart a new problem. With respect to the array V, there are three global variables that are consistently used as pointers; they are last, last 1, last data, list length. The description of these variables appears in the Variable list given in Appendix II. Figure 1 shows the storage layout:

Figure 1



The pointers last and last 1 may, of course, point to anywhere in the array; their particular position depends on the procedure being executed. There are other pointers used to keep track of the status of the array V, but they are of a temporary nature and are local to the procedure in which they are used.

The basic procedures used to create and manipulate the lists stored in the array V are: father, son, get atom, atom, collection, Bncollection, and sum. The first five of these procedures were made available to the author by Professor R. DeVogelaere and have been changed hardly at all, except that a packed version of atom and get atom were constructed. The other procedures have been constructed by the author as needed. It is, of course, obvious to those familiar with Professor DeVogelaere's work that these procedures are strongly influenced by his work which is only natural since the basic list structure that he proposed has not been modified.

Before describing these procedures and how they are used, we first describe the list structure and how it enables us to carry out the substitutions. We recall from Chapter III that the actual creation of the parameter defining equations by means of successive substitutions was to be carried out by means of Theorems 6, 7, 8, and 9. We have pointed out that all the necessary operations can be performed in the coefficient space in the sense that if we know the coefficients of one element, we know how to obtain the coefficients of the derived element in terms of the first element. See, for example, the substitution theorem of Chapter III, Theorem 6.

The coefficients are, however, never known numerically, only algebraically. We must, therefore, be able to identify and manipulate the algebraic quantities that we have previously been writing as β_{ra}, θ_i, etc. Any such coefficient C will have the following structure:

$$C = [\{\frac{a}{b} \times B_1^{e_1} \times \ldots \times B_n^{e_n}\} + \{\frac{c}{d} \times B_{n+1}^{e_{n+1}} \times \ldots \times B_m^{e_m}\} + \ldots] \qquad (1)$$

where a, b, c, d, ... are integers; e_1, e_2, ... are integers;

B_1, B_2, ... are the various parameters that appear in the parameter defining equations. For example, the expansion of an approximation $\xi(\theta_i)$ into a Taylor's series gives rise to

$$\xi(\theta_i) = \xi(0) + D\xi(0)\ \frac{\theta_i^1}{1} + D^2\xi(0)\ \frac{\theta_i^2}{2!} + \ldots \qquad (2)$$

which has the coefficients

$$C_1 = \frac{1}{1}$$

$$C_2 = \frac{1}{1}\ \theta_i^1$$

$$C_3 = \frac{1}{2}\ \theta_i^2$$

.
.
.

Thus, our basic lists, identified as a list of level 0 (list ∅), is composed of the individual terms of the sum of terms that make up the coefficient C. Our secondary list, identified as a list of level 1 (list 1), is the coefficient C, that is the sum of the terms. We visualize C as

$$C = \left.\begin{array}{l} \left.\begin{array}{l} x \\ x \\ x \end{array}\right\} \text{ list } \emptyset \\ \left.\begin{array}{l} x \\ x \\ x \end{array}\right\} \text{ list } \emptyset \\ \quad . \\ \quad . \\ \quad . \\ \left.\begin{array}{l} x \\ x \\ x \end{array}\right\} \text{ list } \emptyset \end{array}\right\} \text{ list 1} \qquad (3)$$

in which list ∅ has as its elements the atoms consisting of (a, b, 1,

e_1, 2, e_2, ..., n, e_n) where we identify B_i by means of the index i, while list 1 has as its elements the lists (list ∅, list ∅, ..., list ∅). It is quite obvious that any approximation $\xi_i = \Sigma\ \alpha_i\ A_i$ can be represented as a list of level 2 (list 2) with the elements (list 1, list 1, ..., list 1) and that the totality of all such approximations would be a list (list 3) comprised of (list 2, list 2, ..., list 2). It has, however, proved convenient to give the second lists, list 2, names by which they can be identified. We shall be more explicit about this shortly.

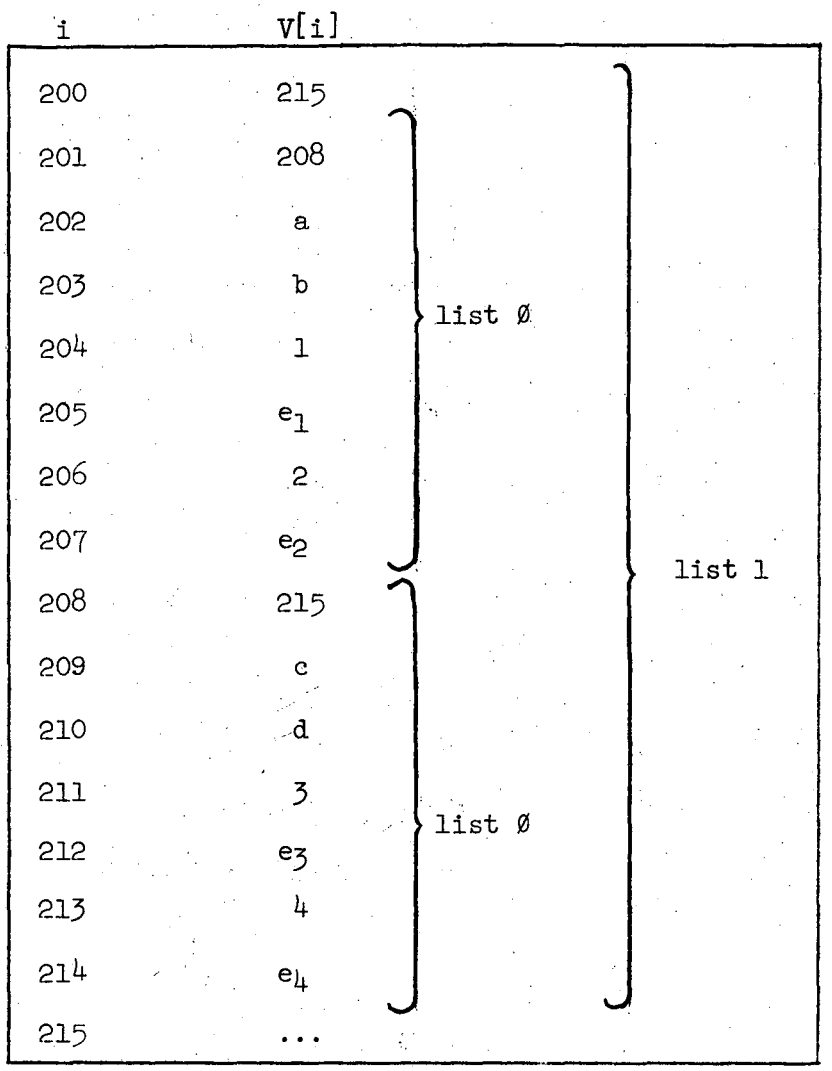
Each list, list i, has an identical structure. Associated with the array V are two quantities; the name (position of) the array element and the contents of that element. We shall, for the present, denote the name of the element by i where $0 \leq i \leq$ list length and we shall denote the contents as V[i]. Each list can be visualized as shown below in Figure 2 where we see that the first entry is the last list item + 1 which, if there are succeeding lists, is the name of the next list which may or may not be of the same level as list i. This next list will, however, not be of a lower order than list i.

Figure 2: General List Structure

i	V[i]
entry to list i = j	last list item + 1
j + 1	first list item
j + 2	next list item
.	.
.	.
.	.
j + n	last list item
entry to next list k = j + n + 1	
j + n + 2	
.	
.	
.	

We can easily illustrate these ideas using the general term C of (1) which, if we assume that the current position of last was 200 at its time of creation, would look like Figure 3.

Figure 3: Equation (1) in List Form



Note: $C = \frac{a}{b} B_1^{e_1} B_2^{e_2} + \frac{c}{d} B_3^{e_3} B_4^{e_4}$.

The order that we have used in (1) is what we shall call the normal form of the list and all lists will be assumed to exist as normal form lists if these elements are atoms. That is, a normal form list is a list of atoms arranged in the order (numerator, denominator, index 1, exponent 1, ..., index i, exponent i, ..., index n, exponent n) where i

appears to the left of index j if $i < j$. Repeated indices are stored in the exponent as powers so there are no repeated indices in a normal form list.

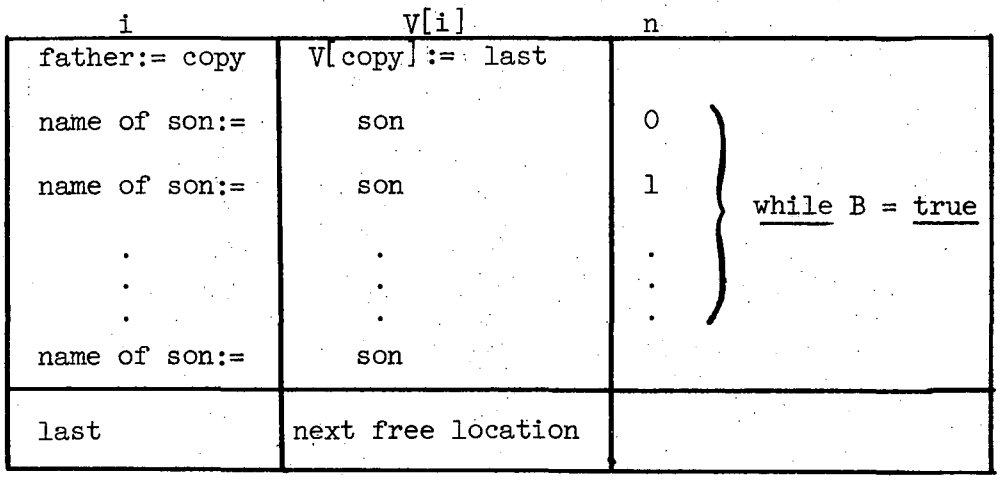
In the procedure RKMI, the subscripted variable B is consistently used when printing the parameters of the scheme. Internally, the parameters are identified by their index value. There is a well-defined ordering of these parameters; some are reserved for interval parameters and then free parameters are simply added as needed in constructing schemes. We shall describe that ordering in detail shortly, but for the present discussion, it is sufficient to state that any approximation $\xi_i = \Sigma\ \alpha_i\ A_i$ has as its coefficients $\alpha_i = c_i$ where c_i has the form given in (1).

If we reflect for a moment, it becomes evident that we shall be able to manipulate the quantities ξ_i in an algebraic fashion provided we can add and multiply their harmonics. The addition is necessary since we wish to form linear combinations $\xi_i = \Sigma\ \gamma_j\ \xi_j$ and the multiplication is necessary because operations such as substitution require the multiplication of harmonics. These harmonics are represented as lists so we need to be able to add and multiply lists. These operations can themselves be reduced to operations that store and fetch from existing lists and that create new lists.

We shall describe first that procedure which creates lists, actually any list i since all lists have the same structure. This is the procedure father. The reader should consult Appendix II for a concise description of father and also to obtain the parameter declarations.

Schematically, father works as pictured below in Figure 4.

Figure 4: The Creation of a List by procedure father.



We see that the actual parameter son should furnish a name ⟨name of son⟩ and a list item ⟨son⟩. We note that father is, itself, a suitable candidate for the actual parameter son. If no list items are created, then (copy = last - 1) has value true and last is returned to copy, the list with name father is nil; that is, empty. The global variable-nil-is consistently used to indicate a non-existent item. The counter n indicates the sequential number of the next list item to be created.

The procedure father, thus, furnishes the basic tool needed to create lists. The creation of the list given as an example in Figure 3 would entail the use of

father(no1,B01,no1<1,father(no,B0,no<6,A,son))

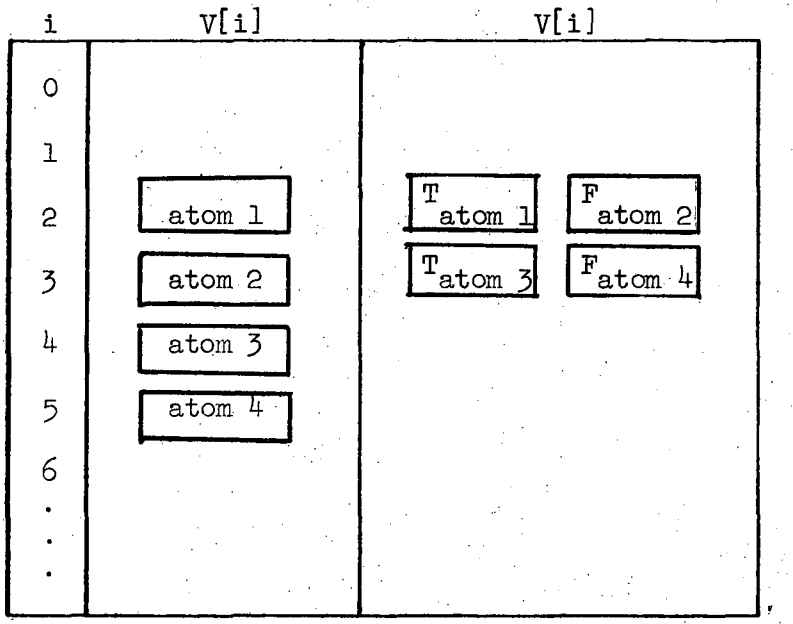
where we have yet to specify the actual parameter son. The above call illustrates the typical use of the global variables that appear as actual parameters. It is, of course, immediately evident that the list i can be created by a call father(...,father(...,...)) where there are i + 1 calls to father.

To complete the creation of a list, it is necessary to store the atoms. This is accomplished using the procedure atom. We shall always assume that the reader is familiar with the procedure descriptions given in Appendix II and that he will also refer there for variable declarations and descriptions that are not given here. Thus, the reader will see from the listing of RKMI that atom is, itself, a suitable candidate for the parameter son in the procedure father. When atom is used as an actual parameter in this position, the elements of the lists are atoms stored in the array elements of V and their names are the current value of last when they were stored.

There are presently two versions of atom; one which stores one atomic value of i in each array element, and another which stores two atomic values of i in each array element. The fundamental purpose of each procedure is identical; however, the packed version is necessarily more complicated. As presently constructed, both routines are completely interchangeable in RKMI. Note, however, that if atom is packed (unpacked) then get atom must also be packed (unpacked).

The operation of the packed version of atom can easily be understood in terms of the unpacked version by means of Figure 5 given below. In the packed case, the first atom is left adjusted, the second atom right adjusted; the global Boolean BA2 is <u>true</u> if we are in the left half of the word, <u>false</u> if we are in the second half.

Figure 5: Packed and Unpacked Version of atom.



The actual word structure is quite simple and we illustrate it on a six-bit word, the generalization to n bits being obvious.

W_1 W_2

W = | a | b | c | d | e | f |

The bits a and d are sign bits. The bit d is on for negative W_2, the sign of W is set to the sign of W_1. A check is made to see that W_1 and W_2 will fit, in this case, less than $2^2 - 1$. We also note that -0 is stored in the second half of the word to indicate an empty second half. It will be seen later that the procedure get atom needs to know when it has run out of atoms. The procedure atom also performs a check before storing to see that there is room in the list to store the atom. This is done using procedure check.

The list of Figure 3 could now be built by calling

```
father(no1,BO1,no1<1,A1,father(no,BO,no<6,A,atom(if
        no1 = 0 then (if no = 0 then a else
                      if no = 1 then b else ...))))
```

where this is the "brute force" application of atom. We have quite often built small local procedures named store to accomplish this storage.

We, of course, implicitly assume all the atoms are numbers and, as will become evident later on, we also assume that if a list of level Ø is non-empty, then there exists at least two atoms (a, b) -- the numerator and the denominator. This, in actuality, implies that zeros are represented as empty lists and, in turn, prevents the formation of coefficients that are identically zero. This, in effect, eliminates identically zero terms since the coefficients are algebraic quantities composed of sums of products.

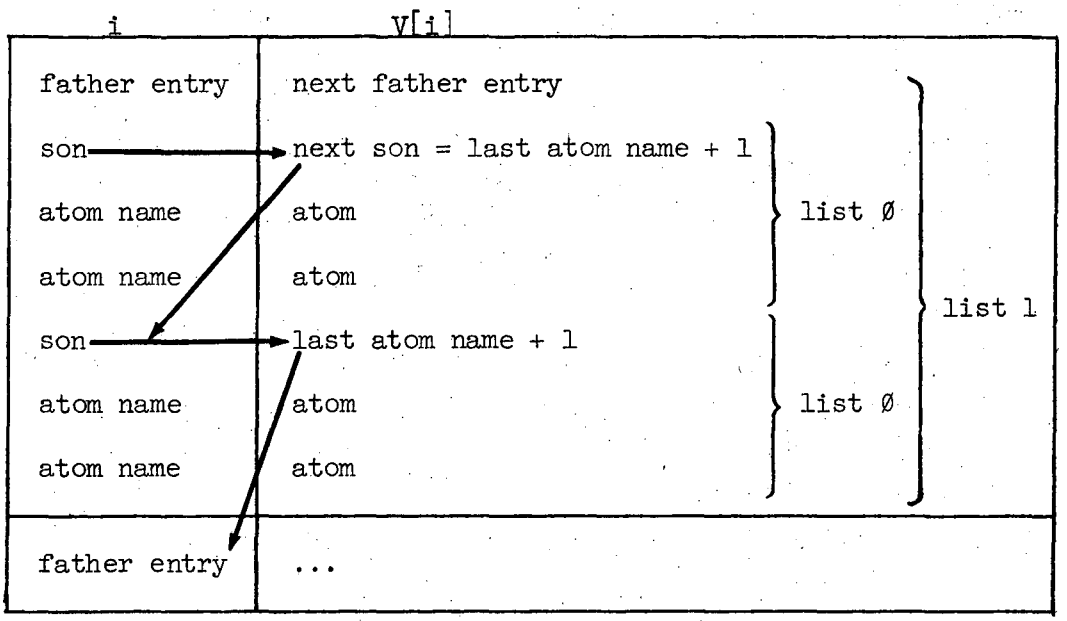
Having built the lists, it is now necessary to be able to extract atoms from these lists. The next procedures described will enable us to do just that. The first of these described is procedure son.

The action of this procedure is easily understood from Figure 6 where we have shown a list of level 1 whose elements are lists of level Ø. We note that the procedure son is, itself, a suitable candidate for the parameter father entry so we have the ability to find sons in lists of level i by calling son(son(son(...))) where the innermost son is the father entry of the list of highest level.

Once we have located the son whose items are atoms, it becomes necessary to pick up these atoms. This is accomplished using procedure get atom.

The manner in which atom works can be easily understood by consulting Figure 6 and the source listing for RKMI in Appendix II. We note that get atom effectively considers all lists to be of level Ø. Thus, a list of level i can be dumped completely using get atom. This is, however, not true for the packed version since only the atoms are packed.

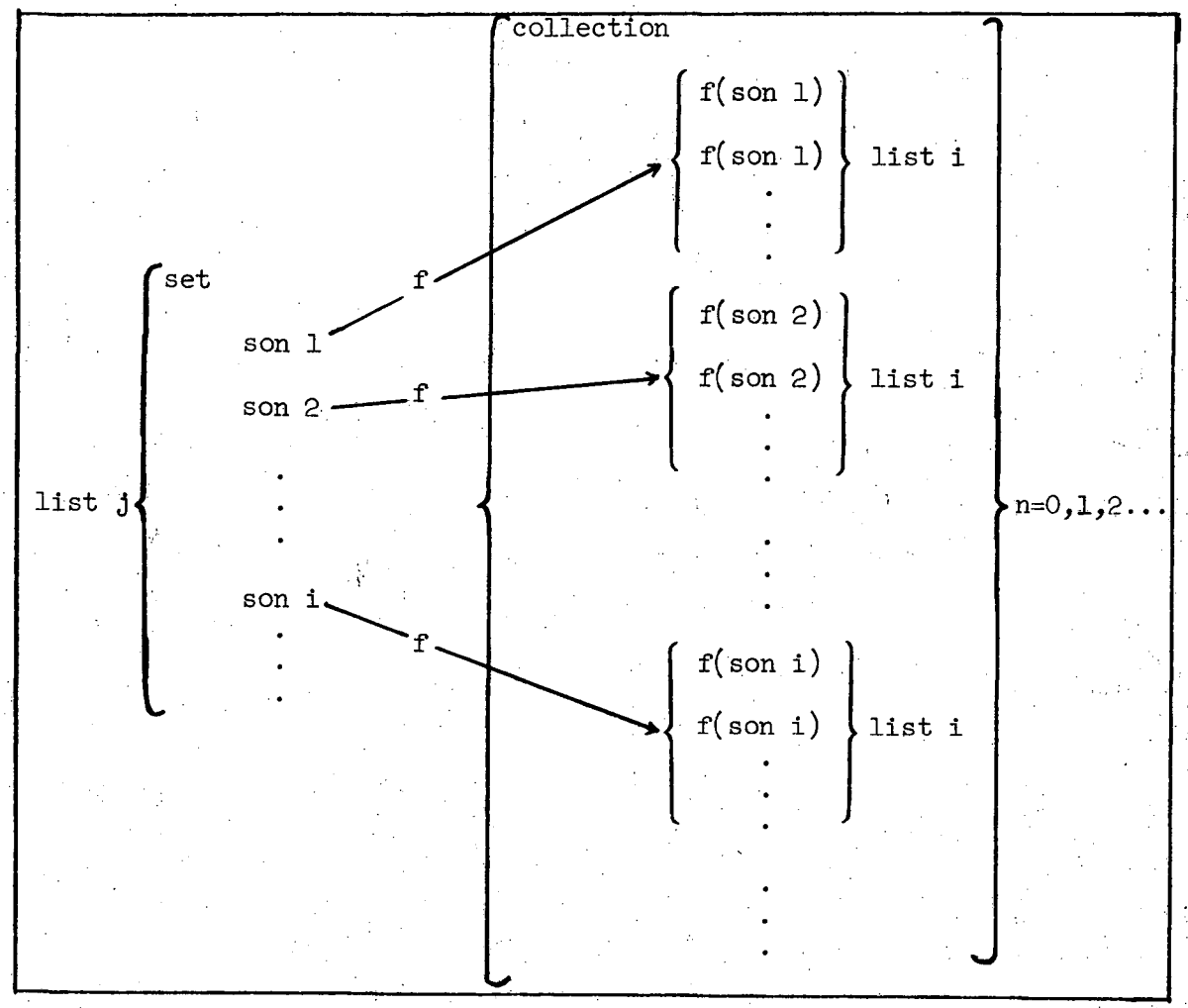
Figure 6: A Pictorial Representation of procedure son.



With respect to the packed version of get atom, we note that its purpose is identical to the unpacked version, and both versions are completely interchangeable in RKMI provided the corresponding version of atom is also used.

We now have at our disposal the ability to create lists, to store them, and to fetch them. There is, however, one more basic procedure that will be needed. This procedure will, as does father, create a list; however, the elements of the list it is creating will depend on elements of another list. This is the procedure collection. The manner in which collection operates can be easily understood with the help of Figure 7. We see from Figure 7 that collection is a mapping whose domain of definition consists of the sons of set j and whose range of values consists of the elements of the lists of level i that are the elements of collection.

Figure 7: The Creation of a List by procedure collection.



It will be seen in the source listing that we have implied that list i is list ∅ for which the elements are atoms. This is, however, not inherent in the construction of collection. The only requirement is that list j be of at least level 1 since it is assumed that there exists sons. Note also that set may be a nil list.

The first operation that we shall treat is that of summation. It is desired to form in the coefficient space the vector sum $S \doteq \Sigma \alpha_i A_i = \Sigma B_j \eta_j$ where $\eta_j = \Sigma \beta_i A_i$. That is, we wish to take a linear combination of quantities whose components are represented by lists the general form of which is given by (1). The reader should refer to Chapter III,

Definition 10, Equations (91), (94), and also (97). It is seen there that it is sufficient to carry out the sum for each individual component and then repeat this for all values of $k \in P = \{0, \ldots, p - 1\}$. In the Program RKMI, we have consistently that

order = p = order of the differential equation

im[0] = the number of basis functions A_i

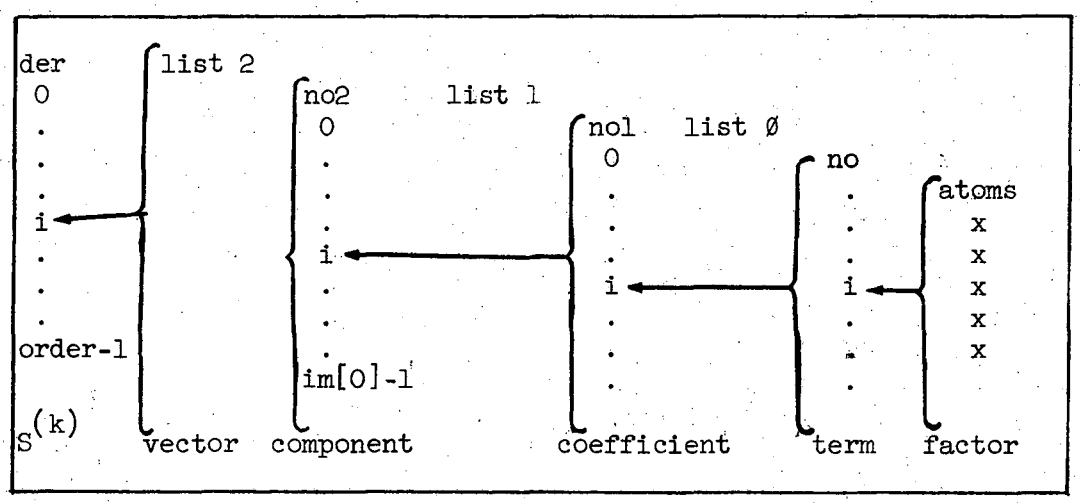
and we usually have that

der = k = $\in \{0, \ldots, p - 1\}$ is the derivative counter

no2 $\in \{0, \ldots, im[0] - 1\}$ is the component counter and also the list 2 counter since a list of level 2 represents the vector $\xi^{(k)}$.

The quantities that are to be summed are assumed to be represented as lists that are ordered as shown below in Figure 8:

Figure 8: List Structure Used in RKMI.



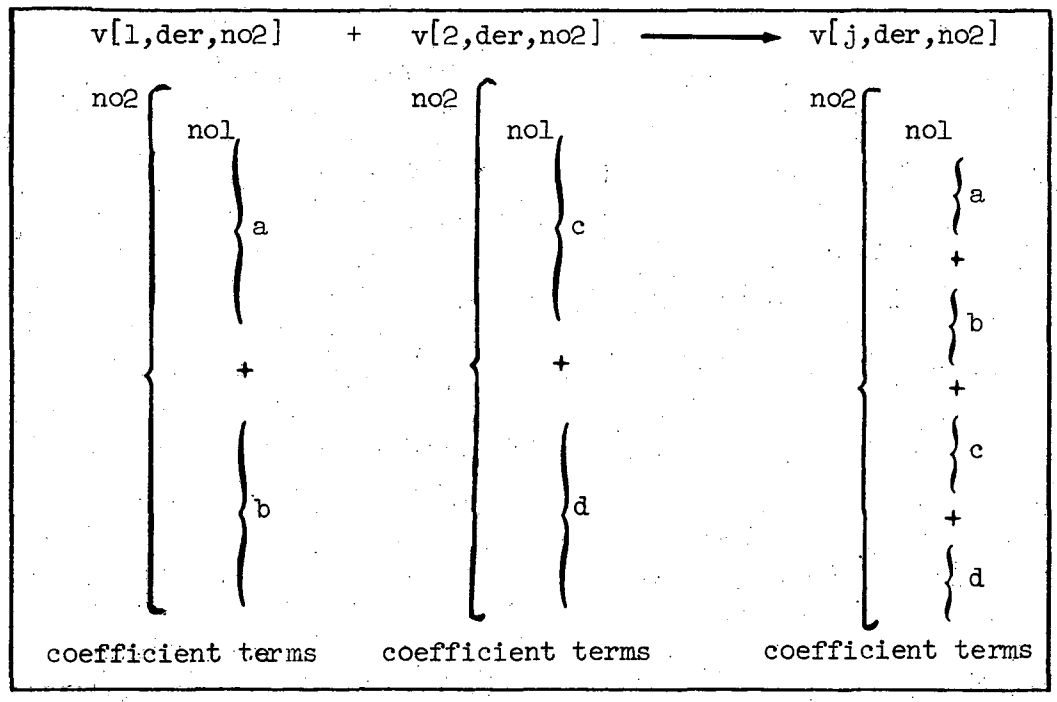
It is immediately evident that it would be natural to include the lists, list 2, as elements of list 3. However, the elements of list 2 are directly accessible by means of names rather than the list structure and the procedure sum has been constructed without this extra level.

The resultant sum is, itself, of the same form since, in reality, all we have done is sum component by component.

The actual summation is carried out by procedure sum. In creating the linear combination, it is necessary to introduce parameters which we call B[ℓ] where ℓ is the global variable of RKMI that is consistently used to indicate the next free parameter.

The actual manner in which the summation is accomplished by procedure sum can be easily understood by referring to Figure 9 below where the components of two vectors are added together to form the corresponding component of the resultant sum.

Figure 9: Creation of a Sum by procedure sum.



We see that in this particular case the procedure collection has been used to collect the separate terms a, b, c, d and write them as one level 2 list. The addition of these two vectors is accomplished by writing sum(v[j,der,no2], i, 2, v[i+1,der,no2]). If a linear combination has been requested, then each term has the extra factor B[ℓ] attached.

Since ℓ is continually increased, the normal form lists will remain in normal form. We would then have that (n, d, B[...], e[...], ..., B[...], e[...]) becomes (n, d, B[...], e[...], ..., B[...], e[...], B[ℓ], 1). If temporary storage is desired, then the resultant sum is written into V starting at temp $\emptyset$.

We now turn to a description of how RKMI effects a substitution $\eta_j = X(\xi_i)$. We know from Chapter III that this operation can be carried out in the coefficient space by means of a substitution table. The manner in which the multiplication $DX(\xi_i) \cdot S$ is carried out is quite similar and will be easy to understand when we understand how to substitute.

Basically, we wish to construct an element $\eta_j = \Sigma\, \alpha_i\, A_i$ that has the list structure indicated in Figure 8. We note that der does not enter into the development since the vector constructed has components α_i that are independent of der = $k \in P = \{0, \ldots, p - 1\}$. The results are based on Definition 5 and Theorem 6 of Chapter III. We start out with the following call to father

father(no2,B02,no2 < im[0],name,Ze)

which is used to construct a level 2 list, list 2, for each component $0 \leq$ no2 $\leq$ im[0] - 1. In RKMI, the name of these lists will always be denoted by Zn or Zp depending on whether the created quantities are stored in permanent storage or in temporary storage. Thus, the quantities ξ, η, S of Definition 10, Chapter III, are identified in RKMI as Zn or Zp. It is immediately evident that Ze must be an integer procedure that completes the construction of list 1 and list $\emptyset$.

We see from the source listing of Ze in Appendix II that W[type,no2] is used as the name of a list of level 1 the entries of which are sons which in turn have elements composed of atoms. If we examine a typical term of Definition 5 of Chapter III, we see that these

terms can be conveniently represented as

(numerator, denominator, θ_i, exponent, der_1, n_1, ..., der_j, n_j, ..., der_{max}, n_{max}) + ...

which almost looks like a normal form list. What we really do is to break up the information into a number of sons that appear as

(numerator, denominator), (exponent), (der_1, n_1, ..., der_{max1},n_{max1}), ... (der_j, n_j, ..., der_{maxj}, n_{maxj}), ... (der, n, ..., der_{max}, n_{max}).

In this form, the first son is a normal form list giving the fraction multiplier, the second son gives the exponent of θ_i, and the remaining sons give the k value and the sequential position of the harmonic appearing in the substitution, or multiplication. We see from Definition 5 that there are needed only three sons to represent a substitution while for a multiplication more sons may be needed. As a concrete example, consider from Table VI of Appendix I the table entry $\frac{1}{1}\theta^1\alpha_{00}$ which is represented as (1,1),(1),(0,0).

RKMI consistently denotes the name of the list specifying θ_i as T[...,i]. The actual substitution, or multiplication, is carried out by list multiplication. If we desire to obtain the coefficient corresponding to a particular component no2, then we look up the first son of the list W[type,no2], called here son ∅. We obtain from the second son, son 1, the exponent for the list T[...,j]. We locate the lists corresponding to the factors specified by son i, $2 \leq i \leq \ldots$. These are lists with name j[der,no2]; remember that, in reality, all such lists have name Z or Zp. Then we perform the list multiplication

(son ∅) × (T[0,j]) × ... × (T[0,j]) × name j[der,no2] × ...

as often as required to construct the resultant coefficients

$$C = \left(\frac{n}{d}\right) \times \theta_j \uparrow (\text{exponent}) \times \ldots \times \ldots$$
$$+ \left(\frac{n}{d}\right) \times \theta_j \uparrow (\text{exponent}) \times \ldots \times \ldots + \ldots$$

which is the internal representation of Definitions 5 or 7 depending on whether we are substituting or multiplying. We note that j = num[0] is the name of ξ_j at which either X or DX is evaluated.

In the procedure Ze, these lists are denoted by B2[...]. After collecting together all factors B2[0],...,B2[n] of the product, the product is formed by setting

Ze:= father(nol,...,father(no,...)).

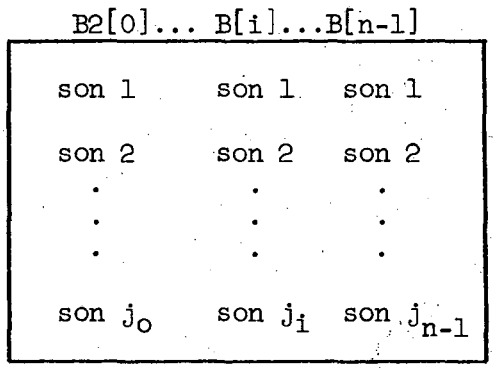
The source listing for RKMI shows that to complete this discussion we must understand the actions of two procedures that appear as actual parameters in the above call.

Each list B2[i] that appears as a factor in the product is composed of a number of sons. To form a product of these lists means that we must form the individual products of all n-tuples (son 1, ..., son i, ..., son n) where son i is taken from B2[i] and we must be sure that all sons are included. That is, we must form all cross products that appear when we take the product of sums of items. The procedure Bncollection obtains these individual n-tuples. The procedure JPn performs the actual multiplication of lists in each n-tuple; each item being a normal form list.

The operation of Bncollection is very straightforward. The lists are laid out like a counter as is shown below in Figure 10. The first row is taken as the first n-tuple. Thereafter, the columns are indexed separately through each son in the manner of a digital sequential counter, starting with the units in the left column, until the n-tuple containing the last row is obtained. A suitable check is made on k to

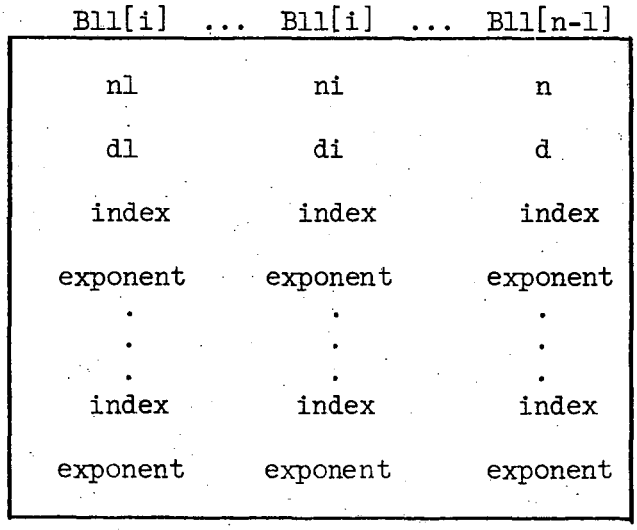
"freeze" the first k columns at the first row. We note that a nil list corresponds to a zero factor which in turn corresponds to a zero term and that setting Bncollection to false causes father to skip this factor; in effect, a nil list is stored. Thus, zero factors never appear in the results.

Figure 10: List Layout for procedure Bncollection.



JPn which carries out the actual multiplication of the elements of the n-tuple operates in a manner similar to Bncollection in that the lists, which are now lists of atoms, are laid out in the same way. See Figure 11 below:

Figure 11: List Layout for procedure JPn.



JPn first obtains the fraction multipliers $c/d = \pi n_i/\pi d_i$. Then, each list is read until all lists are exhausted. The local counter n2 keeps track of the number of lists that have been read completely. When following the operations of JPn, it should always be kept in mind that in each lists the indices, index, increase in sequential order, since the list is in normal form, and that we always have to fetch at least two items, the index and the exponent. We desire to have the resultant product list in formal form; thus, we must search the lists Bl[i] for the smallest index and be careful to have only one index for each parameter. The exponent is then increased to take care of the product of like factors.

To accomplish this, JPn stores the index from each list in v:= (index 1, ..., index i, ..., index n-1) and then uses a small procedure minimum to furnish some needed information about these indices. We see from Appendix II that the call m:= minimum(v,n,w) furnishes the smallest index from the lists Bl[i] and also furnishes the number of factors with that index, thus, allowing us to perform the multiplication

$$B_j^{e_{j_1}} \times \ldots \times B_j^{e_{j_m}} = B_j^{e_{j_1}+\ldots+e_{j_m}}.$$

After this multiplication, each list is positioned at the next index provided its current index is the current minimum index. When all lists are exhausted, a normal form list has been stored. Note that B_j^0 is not stored which is consistent with the interpretation $B_j^0 = 1$. The Boolean BAl is properly set for the packed version of get atom, two atoms/word. If a different packing density greater than two atoms/word was to be used, then some consideration would need to be given to this variable. The procedure normalize is to be used to create a relative prime fraction multiplier; however, since most of the numerators

are small, this procedure has not presently been implemented.

We have, then, that the multiplication of the lists B2 whose elements are sons B11, the elements of which are in turn normal form lists of atoms, can be accomplished by means of the call

father(no1,B01,Bncollection(B2,B11,n,B01,k),A1,
father(no,B0,B0,A,JPn(1,1,n,B11))) (4)

where $0 \leq k \leq n - 1$ is appropriately selected. In the particular case at hand, since we desire only son Ø from W, we set k = 1 thus causing the n-tuples to be selected from the lists B2[1],...,B2[n-1] with only the first son from B2[1] appearing in all the n-tuples.

Equation (4) represents the operation of multiplication in the coefficient space and we shall use it in other procedures of RKMI. It seems natural that a multiplication procedure should be built, although we have not yet done this.

Up until now, we have simply assumed that there exists an expansion for any item that we choose to use. That is, there exists a list of the form indicated in Figure 8, and that any summation, substitution, etc., is to be performed on these lists. This, of course, is not the case. These lists must be appropriately created before they can be used. The next two procedures that we shall describe accomplish this task. In order to carry out a descripton of these procedures, procedure create E and procedure translate, it is necessary that we first present a detailed description of how we organize the interval subdivision and the parameter indexing. It will be recalled that we defined in Chapter III, Definition 8, the quantities involved. In particular, RKMI uses the variable listed below to represent these quantities.

period:= the <u>period</u> of the scheme

q := the <u>rank</u> of the scheme

e := the <u>extent</u> of the scheme

B[i] := the <u>scheme parameters</u> which may be the interval parameters θ_i or the undetermined parameters β_i of an implicit representation of some approximation, or the scheme parameters previously (Chapter III, Definition 1) denoted by A and B.

The manner in which intervals are subdivided is indicated in Figures 12 and 13 which illustrate the two cases considered by RKMI. It will be recalled that in Chapter III we indicated that there are different ways in which the expansion into undetermined parameters may be handled. We have in RKMI made the following choice:

1) Expand about a local origin and translate the expanded vector to the point where all substitutions, multiplications, etc., are to take place. We consistently indicated this by setting the variable mode:= -1. Determine the undetermined parameters by equating their local origin expansion to a constructed expansion.

2) Expand about the point where all substitutions take place and translate this expansion to the local origin. We consistently indicate this by setting mode = 0. Determine the undetermined parameters by equating the translated (local origin) expansion to a constructed expansion.

These two cases are called a forward translation (mode = -1) and a backward translation (mode = 0) and are illustrated schematically in Figures 2 and 3, Chapter III.

Figure 12: Scheme Parameter Determination for a Forward Translation (mode = -1)

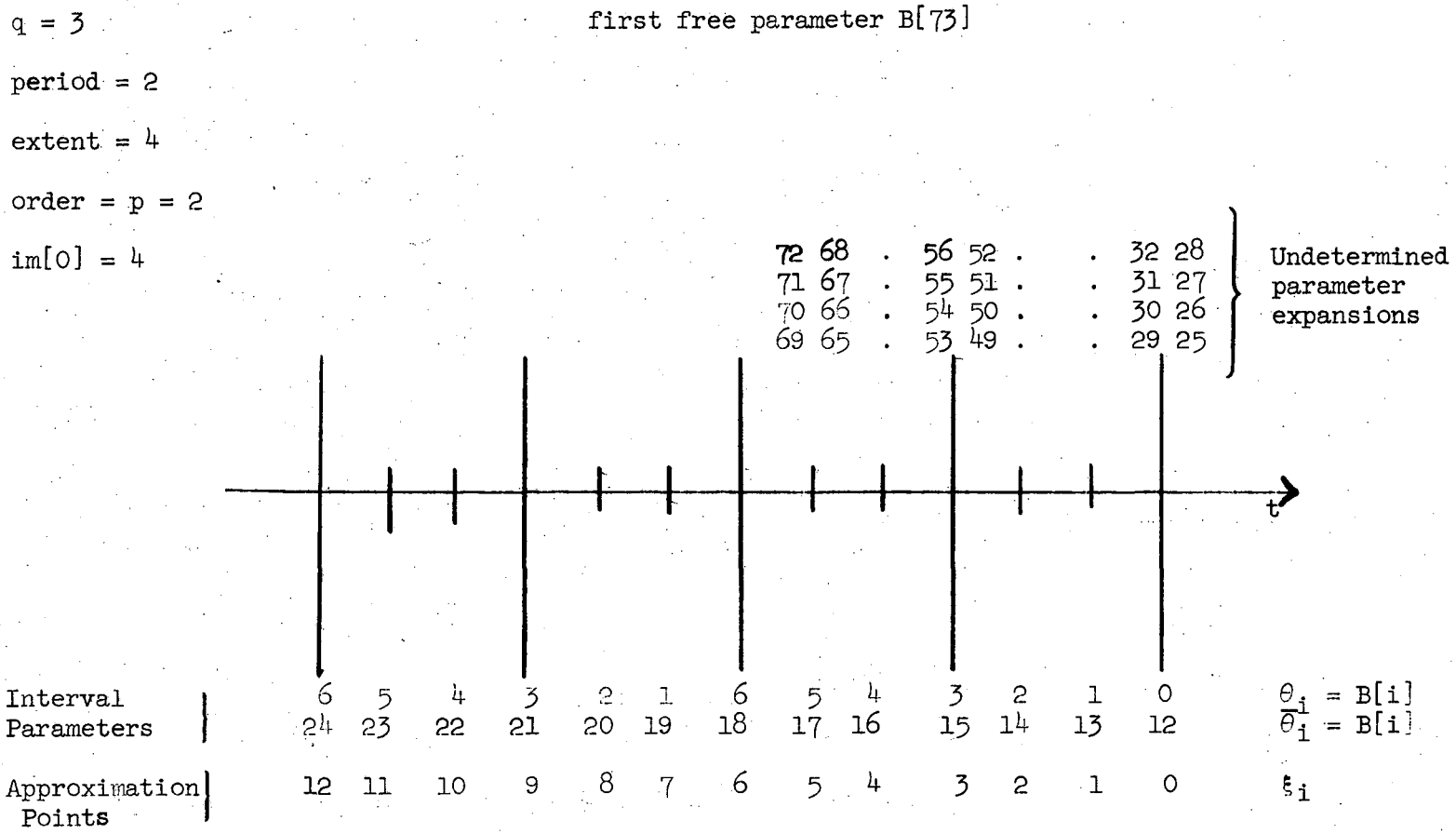


Figure 13. Scheme Parameter Determination for a Backward Translation (mode=0).

q = 3 order = p = 2

period = 2 im[0] = 4

extent = 4 first free parameter B[i] = 129

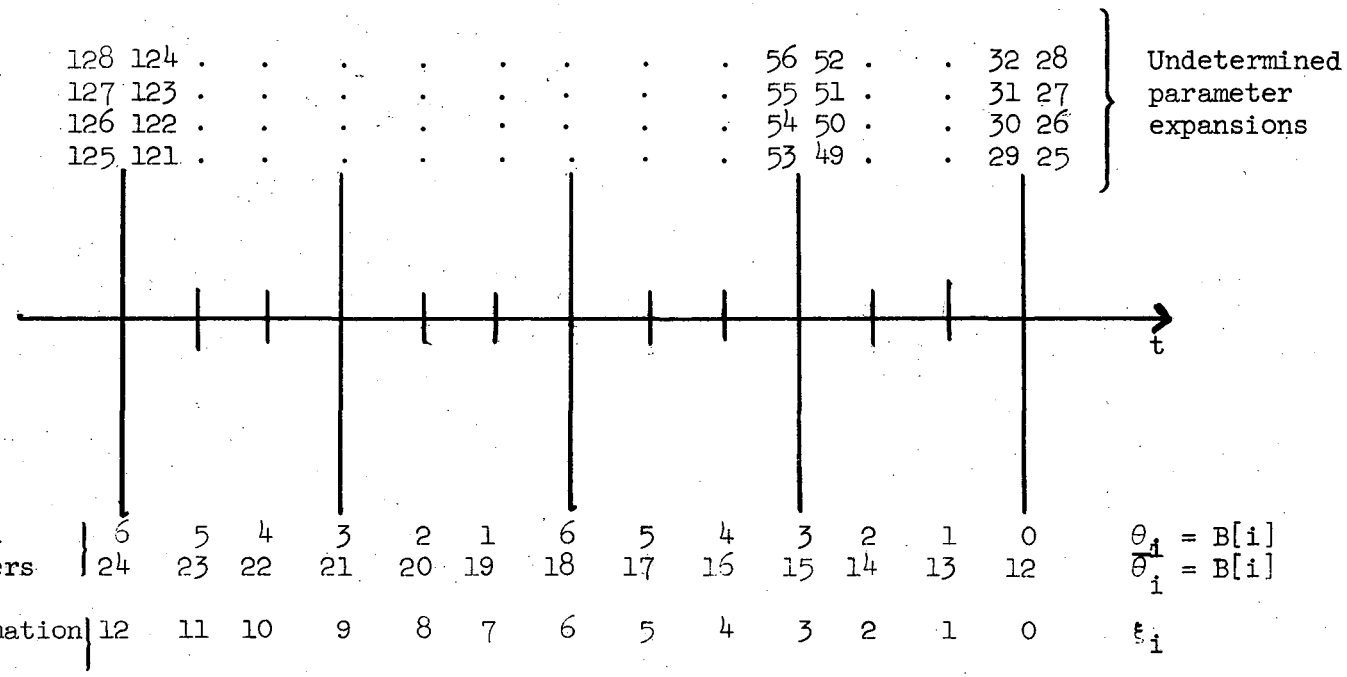
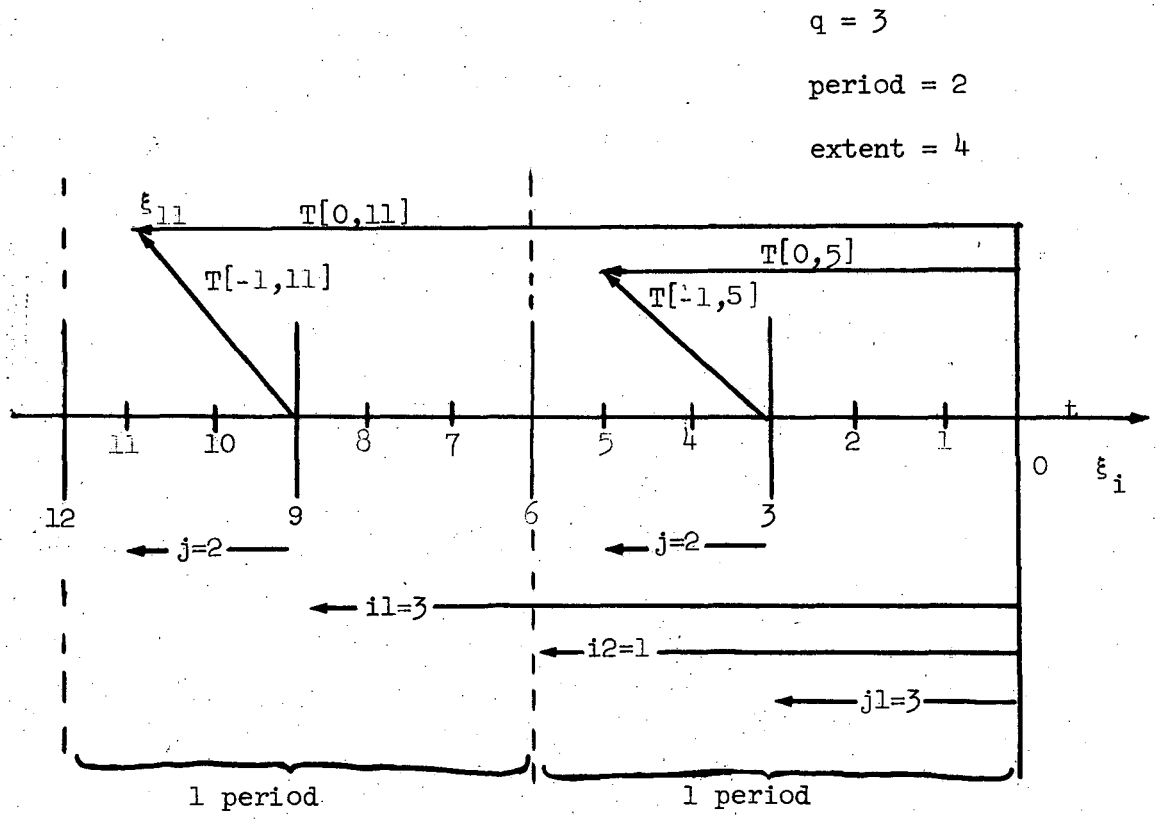


Figure 14: Interval Parameters Returned by procedure index.



In order that we can easily locate any approximation ξ_i, $i \in M = \{0, \ldots, eX_1\}$, there is in RKMI an integer procedure, procedure index, which when called as index (i) returns a number of useful quantities. These are illustrated in Figure 14 above. The most obvious and most frequently used quantity is the value of index which is the corresponding point within the period. For example, referring to either of the Figures 12 or 13, we have that index (5) has value 5 since ξ_5 is within the period while index (11) also has value 5 since ξ_{11} and ξ_5 represent the same approximation. The rest of the values are returned in global variables. These are:

il:= the number of h intervals away from ξ_0

j:= the fraction of the h interval = the distance from the local origin.

i2:= the number of period intervals away from ξ_o

j1:= the local origin within the period that corresponds to the local origin ξ_i.

We see from Figure 13 that the local origin of ξ_i and its corresponding point within the period, ξ_j where ji = index(i), are not dependent on the manner in which the undetermined parameter expansion is carried out. On the other hand, the significance of B[i], the parameters of the method, is directly connected with this choice. If mode = -1 and a forward translation is performed, then the following situation exists for $0 \leq i \leq 2 \times (\text{period} \times q) + ex q$.

We have that

B[0]:= t_o which is the distance from the origin to the point where all substitutions, etc., are carried out.

B[q]:= h which is the distance between major points

B[i]:= if index (i) $\neq$ 0 then T[-1,i] which is the distance of ξ_i from the local origin else

if (index (i) = 0 $\wedge$ i < period $\times$ q) then B[i] is undefined, provided i $\neq$ q, and never appears in any of the equations. However, T[-1,i]:= 0 since the distance from the local origin is zero.

if (period $\times$ q < i < 2 $\times$ period $\times$ q) then B[i] is undefined else

if (2 $\times$ period $\times$ q $\leq$ i $\leq$ 2 $\times$ period $\times$ q + ex q) then T[0,i] which is the distance of t_i from the origin.

The undetermined parameters B[i] are reserved, im[0] $\times$ order of them for each point, starting with i = (2 $\times$ period + e) $\times$ q + 1 and stopping when we reach i = (2 $\times$ period + e + order $\times$ im[0] $\times$ period) $\times$ q. Those points outside of the first period have the same undetermined

parameters as their corresponding point within the period. This latter fact arises because the expansions are carried out about the local origin and the approximations ξ_i and $\xi_{index(i)}$ are the same distance, B[index(i)], from their local origins and because we assume that ξ_i and ξ_j were constructed in the same fashion.

Having reserved the necessary parameters B[i] to represent θ_i, $\overline{\theta}_i$, and the undetermined parameters, the next free parameters available for use as a scheme parameter is B [2 × period × q + exq + 1 + im[0] × order × period × q]. All parameters introduced into the scheme by taking linear combinations of various quantities are sequentially indexed starting with this value.

If mode = 0 and a backward translation is to take place, then the parameter indexing is slightly different. We see from a comparison of Figures 12 and 13 that we have reserved undetermined parameters for all the approximations. We do this because each undetermined parameter expansion is with respect to the origin and since each point is a different distance from the origin, the harmonics are different. This causes the next free parameter index to be

i - 2 × period × q + exq + 1 + order × im[0] × (exq + 1).

This, however, is the only difference in the interpretation of the parameters.

The above discussion shows that each approximation has a well-defined expansion and all we need is to create it and to translate it suitably. We do this using the procedures create E and translate. For each ξ_i referenced, we have two representations, Z[-1,der,i,no2] and Z[0,der,i,no2], which respectively refer to the expansion about the local origin and about the origin, taken to be t_o. With respect to the latter, we note that Z[0,...] references the undetermined parameter

expansion until a constructed representation is obtained at which time it then references that constructed representation. Note that the location of the list representation is forgotten when a constructed representation exists. This is consistent with the fact that in a substitution, etc., we wish to use the undetermined parameter expansion only as long as there is not available a constructed representation. This at least was what was first thought. It has proved advantageous to revise this philosophy and this is a question that is still to be completely resolved. With respect to substitutions, there is no real problem; we can perform all substitutions first and this forces the use of undetermined parameters. However, when performing a multiplication, we use the current representation referenced by the name Z[0, ...] when we evaluate $DX(\xi_i)$ and cannot easily force the program to use the undetermined parameter expansion. A solution is to increase the family of names, but this is at the expense of requiring more storage. The reason for wishing to use undetermined parameters as long as possible is that this simplifies the equations enormously because all distributed products, arising because the components are sums of products, disappear. More thought should be given to the solution of this problem.

The actual manner in which create E carries out this creation of an approximation is easily obtainable from the source listing in Appendix II now that it is known what is to be constructed. We simply note a few details here. The local Boolean B indicates the special case of a forward translation with i a major point or a backward translation with i the first point; that is $i = t_0 = 0$. The quantity E[,] is originally set to nil, then E[0,i]:= 0 implies that their exists T[-1,i] and T[0,i] while E[-1,i]:= 0 implies that an expansion into undetermined

parameters has been made. Note that major points have all zero coefficients when mode = -1 and that we store zero values as nil lists, thus, we presently ignore the parameters of ξ_i when this special case exists. RKMI needs to know which approximations exist and which don't because to save space we only create those approximations that are used and these only once. We also note that these quantities are stored permanently and are always available once created with the exception we noted above about our reference to the undetermined parameter expansion using the name Z[0, ...]. With regard to that, note that we still have the expansion, only its name is forgotten.

Once a full expansion of E is created, it is necessary to translate it appropriately. This is carried out using the procedure translate.

We should point out that RKMI has evolved through many editions and that the formulation of the problem has also evolved. These two processes have gone hand-in-hand as more experience was gained in what can or cannot be done with the program and what the output generated looks like when presented from different viewpoints. This is, of course, natural, but the two processes are separate and when both change considerably, as has happened here, there is bound to be deadwood in the program and procedures are never going to be as neat and elegant as they could be. This is especially true of the procedure translate.

We will not trouble the reader with past history, but do mention that translate originally was to translate a single h interval repeatedly thus enabling it to translate a number of h intervals by doing the same thing over and over again. Also, the interpretation of the interval parameters was different. The remnants of these ideas are still in

translate; therefore, this procedure should be rewritten. With regard to this, we mention that thought should be given to freeing the origin; that is, letting the origin be completely arbitrary. This will allow the full symmetry of the equations to become apparent.

There are a number of details in translate that need to be clarified in order that the source listing given in Appendix II can be easily followed. Before doing this, we illustrate the typical use in RKMI of the procedures create E and translate.

```
if E[0,i] = nil then
begin create E(i,true);
      translate(i,Z[mode,der,i,no2],
                  Z[mode 1,der,i,no2], i ÷ q,mode 1)
end;
```

A check is made to determine whether it is necessary to create the approximation; if it is, then the procedure create E does so, the vector is appropriately translated, and the two representations of ξ_i are now available to be used as required.

Translate can be easily understood once we display the actual manner in which a translation is performed. We have already furnished in Theorem 8, Theorem 9, and Definition 9 of Chapter III the necessary means for carrying out the translation. We must, however, collect these results together into one easily understood process. The reader should refer to Figures 2 and 3 of Chapter III which give, respectively, a schematic representation of the backward and forward translation.

For any approximation, we have the two representations

$$\xi_i^{(k)} = u^{(k)}\begin{pmatrix}\tau & \theta_i \\ 0 & \bar{\theta}_i\end{pmatrix} + \sum_{s=o}^{\bar{s}-1}\begin{pmatrix}\beta_{i,s}^{(k)} \\ \bar{\beta}_{i,s}^{(k)}\end{pmatrix} A_N\begin{pmatrix}\tau \\ 0\end{pmatrix}[s] \qquad \begin{matrix}\text{mode} = -1 \\ \text{mode} = 0\end{matrix} \qquad (4)$$

$$\bar{\theta}_i = \theta_i + \tau$$

where the upper row of arguments and coefficients is to be used when mode = -1, forward translation; and the lower row is to be used when mode = 0, backward translation. Equation (4) represents the untranslated approximation. The corresponding translated approximation can be written as

$$\xi_i^{(k)} = u^{(k)}\begin{pmatrix} 0 & \bar{\theta}_i \\ \tau & \theta \end{pmatrix} + \sum_{j=1}^{\bar{j}-1}\left[\sum_{n=\bar{s}}^{\bar{s}+\bar{n}-1}\left\{\sum_{m=0}^{k+\ell}\binom{\tau}{-\tau}^{a}\cdot\binom{\theta_i}{\bar{\theta}_i}^{m}\frac{1}{a!\,m!}\right\}\right. \tag{5}$$

$$\left.\cdot\, a[\text{type } 2,\ n-\bar{s},\ j] + \sum_{n=0}^{\bar{s}-1}\gamma_n\ a[\text{type } 1,\ n,\ j]\right]\cdot A\binom{0}{\tau}[j]$$

$$a = k + \ell + n + 1 - m - \bar{s}$$

where again the upper argument are for mode = -1 and the lower ones are for mode = 0. The results given here in (5) are equivalent to Theorem (8), Chapter III; however, to arrive at (5), it is more convenient to start with equation (66) of Chapter III, expand the derivatives as

$$D^{p+\ell+n-\bar{s}}x(\zeta) = \sum_{s=0}^{\bar{s}-1} a[\text{type } 2,\ n-\bar{s},\ s]\ A(\zeta)[s],\quad n=\bar{s},\ldots,\bar{s}+\bar{n}-1,$$

and then (5) is easily obtained.

The coefficients a[type 2, i, j] are derivative harmonics of $D^{p+\ell+i}x$ and are readily available from the tables of Appendix I. The coefficients a[type 1, n, j] are the translation harmonics that represent the change of basis

$$A\binom{\tau}{0}[n] = \sum_j a[\text{type } 1,\ n,\ j]\ A\binom{0}{\tau}[j]$$

and these too are available from Appendix I.

The key formula to the understanding of the translation process is (5). The coefficients of interest are

$$\left\{\quad\right\} \cdot a[\text{type } 2,\ n - \bar{s},\ j]$$

and

$$\gamma_n \cdot a[\text{type},\ n,\ j].$$

Each of these is represented as a list of level 1. See Figure 8 with regard to this representation. The array elements of a are stored in the appropriate list structure by data. The harmonics γ_n of the untranslated vector are already in a list structure and the procedure translate creates the lists that represent the coefficient { }. Thus, to obtain C_j, the coefficient of the translated vector, we need only form the sum of lists that are obtained as the products of lists. We know how to do this using the already described procedures sum and JPn.

The first thing done upon entry into translate is to check whether v is far enough away from the origin. If the origin turns out to be the local origin of v, then we perform the identity translation. If the vector requires an actual translation, then storage is reserved for some temporary arrays. The appropriate coefficients are stored in lists with names vs[der,no2], first the coefficients of v and then the bracket coefficients. The translation harmonics and the derivative harmonics are located by storing the names of their lists in al[no2,j2]. We note here that the translation table depends on the rank q and the direction of the translation. Translate takes care of this since at the same time that it is locating the names of the translation table, it is also using the procedure collection to insert in the table the value of q and the appropriate sign of the coefficient. Note that the initial manipulation of k3 and k4 just prior to obtaining vs is to insure that normal form lists will be constructed.

Since the correspondence of the indices used in (5) and in translate

is not necessarily obvious, we give below the correspondence for the bracket coefficient which we denote as α

$$\alpha = \sum_{m=o}^{k+\ell} \begin{pmatrix} 1 \\ -1 \end{pmatrix}^a \frac{1}{a!\,m!} \left(\frac{\theta_i}{\bar{\theta}_i}\right)^m (\tau)^a$$

$$= \sum_{i=o}^{kl} (num2) \uparrow jl \; \frac{1}{(jl)!\,(i)!} \cdot B_j^i \cdot B_t^{jl}$$

where we note that the array b is used to store the list items temporarily for this sum of terms.

To complete the translation, it is necessary to form the products indicated in (5) and carry out the summations. The result has name 2 and is stored permanently.

We have described, in some detail, all the procedures of RKMI that are concerned with actual list manipulation. The rest of the procedures, with the exception of data, are concerned with the output of information. Of these, the only ones that are of interest here are those that help construct the parameter equations. The other procedures can be easily understood with the aid of the listings in Appendix II. Procedure data has a special description given in Appendix IV since it pertains to the use of RKMI and this procedure is an input procedure.

The two output procedures that concern us here are procedure print sum and procedure conditions E. The first of these, print sum, is almost self-explanatory with the aid of Appendix II. However, it helps to remember that

length 1 = the number of approximations E in the sum

length - order × length 1 = the number of approximators N in the sum.

vs[0,i]= the names of the approximations E in the sum.

vs[1,i]= the names of the approximators N in the sum.

Procedure conditions is a little more interesting. It has as its sole purpose the printing of equations (99), (100), (101) of Theorem 10, Chapter III. Each approximation consists of a linear combination of approximations, approximators obtained from a substitution, and approximators obtained from a derivative multiplication. Of these three, only the parameters appearing in the first two have conditions attached to them. Thus, when the sum is created, and when the approximation is created, certain information must be stored that will allow us to now identify which parameters appear in the approximation and which of these parameters have conditions attached to them. For this, we have

E[0,i]:= the minimum value of all B_j appearing in ξ_i

E[1,i]:= the number of approximations in ξ_i

E[2,i]:= the number of approximators $\eta = X(\xi_j)$ in ξ_i

E[3,i]:= the total number of items summed to obtain ξ_i

v[0,i,j]:= the names of ξ_j appearing in ξ_i, thus allowing the identification of θ_j

v[1,i,j]:= the names of ξ_j appearing in $\eta = X(\xi_j)$, thus allowing the identification of ξ_j

Since it knows the first parameter appearing in ξ_i, the length of the sums and the manner in which ξ_i was constructed, conditions E can calculate the appropriate indices for the parameters that appear in the conditions associated with this approximation. These conditions are printed exactly as they appear in Theorem 10 of Chapter III.

VII. EXAMPLES OF GENERATED SCHEMES

This chapter is devoted to the treatment of five examples all of which have been generated using the ALGOL procedure, RKMI, based on the work of Chapter III. We first present a classical 4th order Runge-Kutta scheme. This scheme is verified on four sets of well-known coefficients. We then proceed to present four simple generalized Runge-Kutta schemes that actually make use of data from the past; that is, have a memory. These example, which are the simplest cases of the class of generalized schemes treated in the previous chapters, are believed to be new. They are analogous to the k-step methods presented by Butcher; however, they make use of only one intermediate function evaluation and, therefore, require one less function evaluation per step than do his examples. The results lead to k-step methods of accuracy $O(h^{2k+2})$ that have stable correctors. For each of these schemes, a start is presented for the intermediate point since it requires a correct start. For Examples 1 and 2, a complete starting procedure is also provided, and for Example 2, a complete presentation of the RKMI results is given.

This chapter contains the discussion of these examples and Appendix III contains all the tables and graphs which pertain to these examples. For each example, we give error terms and, where applicable, determine the range of a stable corrector.

As an introduction to the generation of schemes using the ALGOL program, RKMI, we give, in complete detail, the classical fourth order Runge-Kutta process. This process was selected for various reasons. In particular, it is a well-known method that is sufficiently complicated to be interesting while, at the same time, it is not too complicated. We gain nothing by presenting a fifth or higher order process and a third

order process is rather simple. Because this process is well-known and has a variety of solutions, we have at our disposal a number of sets of test coefficients. This is an extremely important point to remember when creating schemes using RKMI. It is of the utmost importance to try and verify the output on known coefficient sets; otherwise, all feeling for the validity of the results is completely lost.

The results of this example are used later in the development of a starting scheme for Example 2 and thus we have some continuity in our presentation and some confidence that the material which we use to generate the starting scheme is accurate.

How one actually organizes the problem to obtain input data for RKMI is of little importance; however, we have presented one approach in Appendix IV where we used RK4 as an example.

If the reader will refer to Example RK4 in that appendix, he will see that we:

1) define the problem,
2) establish a scheme to be generated,
3) layout the interval division,
4) set forth the interval parameters of interest which are the rank, period, and extent,
5) extract the data for RKMI consisting of the order, upper, rank, extent, number of basis functions, and the number of derivatives included in this choice of basis functions.

Note that we have purposefully left out the scheme parameters that multiply the right hand side of 2). This is convenient and quite natural since RKMI will supply us with these coefficients. We see that there are only four points in one major interval for RK4, thus, the rank = 4;

the scheme repeats after one major interval, thus the period = 1; there is a totality of one major interval, thus the extent = 1. Also, since this is a classical RK scheme, we know that the lower bound for the order of accuracy for all points is not greater than 1 and that the greatest lower bound for this accuracy is $\ell = 1$. This setting of the ℓ value and the extent e to 1 is one of the characteristics of classical RK schemes. As we have mentioned earlier, finite difference methods have ℓ equal to the order of the method minus 1 and have an extent equal to the number of points incompassed by the method.

If we were dealing with an unknown scheme, we would, at this point, have to decide how accurate we were going to force the intermediate points to be. For example, in our k-step examples that follow, we do as Butcher has done and say that the intermediate points (minor points) are of $O(h^{2k})$ and the major points have accuracy $O(h^{2k+2})$. We also know to what order of accuracy we can match the derivatives of x for this example. If we let $\tau_k = \xi_o^{(k)} - \xi^{(k)}(t_o)$ be the local error in the p-kth derivative, then we have that $\tau_o = O(h^5)$, $\tau_k = O(h^{5+k-1})$ for k = 1, 2, ..., p - 1. We write this as $\tau_k = O(5, 5, \ldots, p + 3)$.

Since we have $\ell = 1$, we have that DX is the first derivative that we have expanded into the set of differentials A. Referring to Table V, Appendix I, which gives these differentials, we see that we must decide how many orders to use in the expansion or, equivalently, how many derivatives of X are matched and how many error terms of τ_k we wish to look at. For the RK4 process, we see that in order to include the principal error term, we must examine through D^4X and, thus, we have all the terms from the orders $\ell + 1, \ldots, \ell + 4$ corresponding to these four derivatives. This would be 22 functions meaning im[0] = 22. However,

we must actually specify p; we choose p = 1 for this example. Once this is done, we have that $k \in [0, 0]$ and from Table V, we see that there are only 16 functions $\psi_0, \ldots, \psi_{15}$.

The reader should also refer to Table I, Appendix III, and examine the actual input used to generate this example. There is a direct connection between 5) and the data input that particularizes the problem, and also between 2) and the scheme that generates the output. We suggest that any integration scheme in the class of schemes considered in this work can be characterized in a simple, straight-forward fashion similar to that presented here. When this is done, the data needed to generate the scheme will be readily available for input into RKMI.

We will not discuss the actual data setup that is presented in Table I. Appendix IV contains a description of how to prepare the data for the procedure data which inputs the harmonics, and also a general description on how to use the program. This, along with a good understanding of the Tables VII and VIII of Appendix I, should enable the user to prepare data.

Given the input of Table I, Appendix III, the program RKMI generates the output given in Table II. The first section of output shows what type of language the equations are printed in (FORTRAN for this particular run), how much storage we have allowed for various items, and how we have set some internal output parameters. In particular, we have 70 characters per card (line) on the equation output, 6000 words for permanent storage, and 1000 words for temporary storage giving a total of 7000 words in the list storage array. The maximum number of approximators η, (N), or sums S that may be created is ten; the maximum number of items in a sum is ten; the maximum number of lists that we may multiply together

at any one time is ten.

We then specify the problem that is to be solved. Here a first-order differential equation $Dx = X \circ x$. The value of upper which is actually the value of the lower bound ℓ determines the order of the derivative of X that appears first in the basis functions A. Here it is D^2X which is correct for RK processes. We note, as an aside, that we could have used $\ell = 0$ as does Butcher[1]; however, this leads to an unnecessary profusion of equations and our tables in Appendix I are not valid for $\ell = 0$; they are missing terms of degree $s \geq 2$ and need to be expanded if one wishes to consider such cases. However, since we can always set $\ell = 1$, we shall pay no more attention to this.

Since the rank is 4, there are four points in one h interval; the period is 1, and we have 16 basis functions and the corresponding four derivatives of X.

We have input the data tables with no output so we simply receive the message to this effect. There is a procedure scheme to output the problem being solved, essentially as it has been stated in Equations (1) - (4) and (91) - (94) of Chapter III; we have not used it here. Our comment is then echoed on the output medium.

The rest of the output constitutes the scheme and the parameter equations associated with that scheme. At this point, we should make a general comment on that output. Originally, it was planned to present the totality of output in reference ALGOL letting all the transliteration be done by machine. This would ultimately lead to output that had both the character and quality of typed work and the dependability of machine generation and translation. In principle, this is simple to do and RKMI can generate all output in CDC ALGOL. In practice, however, it turned

out that much of the actual computations done to verify the results and generate the new processes were done in FORTRAN because of the availability of those facilities. We, thus, at the present time, have a mixture of languages in our output. We do not recommend, or even approve of this; however, practicalities have dictated the form of the present results. It would be a pleasure to see these results in reference ALGOL.

There is no lower and upper case type available on the printers and so we must remember that we have the following transliterations in our output for the examples presented here.

Text	Reference	RKMI Output
η_i	N[i]	N(/i/)
X	X	X
$\xi_i^{(k)}$	E[i,k]	E(/i,k/)
ξ_i	E[i]	E(/i/)
$\xi(\theta_i)$	E(i)	E(i)
k	k	K
B_i	B[i]	B(/i/)
$u^{(k)}(\delta,t_i)$	u[δ](t$_i$)[k]	u(/δ/)(B(/.../)(/k/)
$A_i(\delta)$	A(δ)[i]	A(δ)(/i/)

The RKMI output above refers to the examples presented in this chapter and Appendix III. For different values of type set, a slightly different transliteration will be obtained.

The scheme is developed exactly as specified by the input. The substitutions are performed and then the linear combinations are made. In order to understand the sums, recall equation (97), Chapter III and note that in the double sum the first sum on i corresponds to $\sum_{j_1 \epsilon S_1}$; that the second sum on j corresponds to $\sum_{k_1 \epsilon P}$; and that the parameters B

correspond to the $a^{(k)}_{ik_1j_1}$. The second term corresponds to the second term of Equation (97). Below the summation we identify the index sets v[0,i] and v[1,i] that tell which E and which N appear in the sum. For example, v[0,i] = v[0][i] which is the i-th element in the set v[0]. This notation, although a little obscure for elementary schemes, has advantages in that it does not change in size or form as the complexity of the scheme increases. In actual practice, it usually suffices to look at the sets v[0] and v[1].

The scheme generated can easily be identified by presenting the coefficients as in Figure 1 below:

Figure 1. RK4 Parameter Table

ξ	ξ	X			
i	4	4	3	2	1
3	77	78			
2	79	80	81		
1	82	83	84	85	
0	86	87	88	89	90

where we write as entries in the table the indices of the ξ_i, X_i, and B_i that appear in the scheme. For example, $\xi_3 = B_{77}\,\xi_4 + B_{78}\,X_4$. This presentation is essentially the same as that of Ceschino-Kuntzmann; however, from a practical viewpoint, it is convenient to present all the scheme data.

Below the scheme, we output all the parameter equations associated with that scheme. The interval parameters are defined, B_0 is not specified since it is the distance of t_0 from the origin, and the location of the origin is essentially arbitrary.

The conditions on ξ_0, ξ_1, ξ_2, ξ_3 are those given in 5) Condition A, of the summary at the end of Chapter III. These are origin independent

as presented here. They should all evaluate as zero.

The equations following next are those given in 6) Condition B of the summary. Note that if ξ_o has not been used in a substitution $X(\xi_o)$ or $DX(\xi_o)$, then these equations are origin independent in the following sense: they represent the harmonics of ξ_o expanded about the origin. To obtain an integration process, we match these to the harmonics of $\xi(\theta_o)$ and if the origin is selected at t_o, then all harmonics of $\xi(\theta_o)$ are zero. If use has been made of $X(\xi_o)$ or $DX(\xi_o)$, then it is necessary to locate the origin at $B_o = 0$, that is at t_o, in order that the equations of this section remain valid.

These first two sets of equations corresponding to Conditions A and B are the parameter defining equations of the scheme. We note, however, that we have used each approximation before it was constructed; thus, it has been represented as an expansion in terms of undetermined parameters. These are defined at the end of the output. For example, $\xi_1 = u(0,B_1) + \sum_{i=o}^{15} B_{29+i}\, A_i(0)$. The actual definition of these parameters is given in the section entitled, "Comment equations which define the undetermined parameters used in the expansion of ...". In any actual use of the equations, care must be exercised to insure that all parameters are defined before they are used. This can necessitate the reordering of the presentation of the output. However, because of the recursive nature in which the quantities appear in proceeding from lower to higher order, it should always be possible to do this. For this RK4 example, it is necessary to actually calculate the parameters as ξ_3, ξ_2, ξ_1, whereas, they have been presented in the reverse order.

At the end of the output, we have a short dump showing how much storage was used and how many parameters were used. It is evident that

we could have used considerably less storage when generating this scheme.

In Table III of Appendix III, we specify the results on four sets of coefficients. These are taken from Ceschino-Kuntzmann[2] and the error terms can be compared directly with those he presents.

We note that the array C[i,k,n] appearing in the output from RKMI is meant to have its indices start from zero; however, since this is not possible in the FORTRAN language, we have started the indices from one. This causes the index sets mentioned just before the interval parameters to be off by one. If type set is set for ALGOL output, the indices do start from zero in the output.

We shall now present a set of simple generalized Runge-Kutta methods that have all been obtained using the procedure RKMI. These examples are the simplest generalized RK schemes using data from the past. They are characterized by having a rank q = 2, and an extent e = 2, 3, 4, ... We recall that classical Runge-Kutta processes have extent 1 and rank $\geq$ 1 and finite difference methods have rank 1 with extent e $\geq$ 1; thus, the example given are indeed generalized RK schemes. There is one intermediate point and we require that the approximation at this point have an order of accuracy at least 2 orders less than that at the major point; otherwise, we shall have a classical predictor-corrector scheme.

The work that we present to a large extent parallels that of Butcher and his work has proved very helpful in providing a method[8] for the solution of the parameter equations and in furnishing examples[1] that can be used to check the correctness of the results generated by RKMI. A comparison of our results with his will show that for each example we present there is corresponding scheme of Butcher's and that because we

use an approximation from the past, we have one less function evaluation in each case.

We shall treat the following:

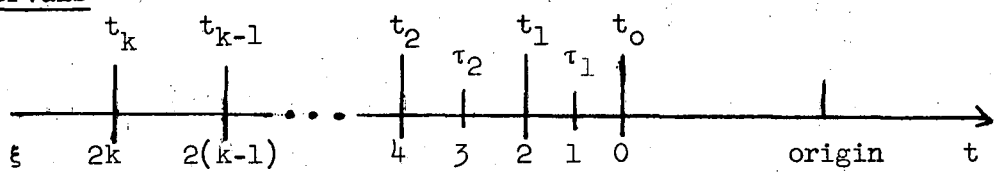
Problem $\qquad Dx = X \circ x$

Scheme

$$\xi_1 = \sum_{j=1}^{k} a_j \xi_{2j} + h \{ \sum_{j=1}^{k} b_j X_{2j} + \beta_1 X_3 \} \tag{1}$$

$$\xi_o = \sum_{j=1}^{k} A_j \xi_{2j} + h \{ \sum_{j=1}^{k} B_j X_{2j} + \bar{b}_2 X_3 + \bar{b}_1 X_1 \} \tag{2}$$

Intervals



We have $2k + 1$ coefficients in (1) and $2k + 2$ coefficients in (2). Thus, we shall seek a solution to (2) such that $\xi_o - \xi(t_o) = O(h^{2k+2})$ and $\xi_1 - \xi(\tau_1) = O(h^{2k})$. This leaves one free parameter in (1) plus the interval parameter τ_1. We shall assume $t_{i+1} - t_i = h = 1$ and $\tau_1 - t_2 = h = 1$.

Since the order of accuracy of the intermediate point ξ_1 is more than one order lower than ξ_o, we shall have error terms from ξ_1 appearing in ξ_o. Schematically, we shall have with $\ell = 2k - 1$ the situation given in Figure 2.

The equations corresponding to ψ_o, ψ_1, are what we call finite difference type equations and arise by requiring that the approximation be exact for polynomials of degree less than or equal to $2k + 1$ (see Condition A of the work of Chapter III and set $\ell = 2k + 1$). The equation corresponding to ψ_2 arises from the substitution of ξ_1 and is what we shall call a Runge-Kutta type equation. This latter equation will be satisfied using the free parameter of (1). The interval parameter τ_1

will then be available for use in obtaining a stable corrector with a minimum principal error term.

Figure 2. Differentials and their Corresponding Orders for Examples 1 - 4

ψ	Order	
0	h^{2k}	
1	h^{2k+1}	
2		
3	h^{2k+2}	Principal Error Terms
4		
5		
6		

There are a number of ways to obtain the parameter values that will make $\xi_1 - \xi(t_1) = O(h^{2k})$ and $\xi_o - \xi(t_o) = O(h^{2k+2})$. Since, however, both ξ_3 and ξ_1 represent points not on the standard interval steps, we cannot easily use a derivative interpolation formula to find these parameter values. We shall find it convenient to use the complex analysis approach used by Butcher[8] and refer the reader to his work for a slightly more detailed description than we give here.

To obtain a solution to (2), we choose a polynomial

$$\bar{\phi}(z) = \frac{(z - t_o)\, K}{\prod_{j=o}^{k} (z - t_j)^2} \left[\frac{1}{(z - \tau_1)} + \frac{M_1}{(z - \tau_1)^2} - \frac{1}{(z - \tau_2)} - \frac{M_2}{(z - \tau_2)^2} \right] \quad (3)$$

where K, M_1, M_2 are parameters as yet to be determined. We shall require that $\bar{\phi}(z) \equiv \phi(z)$ where $\phi(z)$ is determined by (2) to be

$$\phi(z) = -\frac{1}{z - t_o} + \sum_{i=1}^{k} \frac{A_i}{(z - t_i)} + h\left\{ \sum_{i=1}^{k} \frac{B_i}{(z - t_i)^2} + \frac{\bar{b}_1}{(z - \tau_1)^2} + \frac{\bar{b}_2}{(z - \tau_2)^2} \right\}. \tag{4}$$

Now let $R(\zeta)$ be the residue of $\bar{\phi}(z)$ at the point $z = \zeta$. Then we shall require that

$$\begin{aligned} R(t_o) &= -1 \\ R(\tau_1) &= 0 \\ R(\tau_2) &= 0 \end{aligned} \tag{5}$$

giving three equations for the three unknowns K, M_1, and M_2. When this is done $\bar{\phi}(z) \equiv \phi(z)$ and we can solve for A_i, B_i, and b_i in terms of K, M_1, and M_2.

Similarly, to obtain the solution to (1), write

$$\bar{\phi}(z) = \frac{1}{\prod_{j=1}^{k} (z - t_j)^2} \left[\frac{L_1}{z - \tau_1} + \frac{L_2}{z - \tau_2} + \frac{L_3}{(z - \tau_2)^2} \right] \tag{6}$$

and

$$\phi(z) = -\frac{1}{z - \tau_1} + \sum_{i=1}^{k} \frac{a_i}{(z - t_i)} + h\left\{ \sum_{i=1}^{k} \frac{b_i}{(z - t_i)^2} + \frac{\beta_1}{(z - \tau_2)^2} \right\}. \tag{7}$$

We then have that $\bar{\phi}(z) \equiv \phi(z)$ provided

$$\begin{aligned} R(\tau_1) &= -1 \\ R(\tau_2) &= 0 \end{aligned} \tag{8}$$

which furnishes two equations for the three unknowns L_1, L_2, L_3, thus, leaving one free parameter.

Let us now define

$$
\begin{aligned}
c_i &= \sum_{j=1}^{k} \frac{1}{t_j - \tau_i} \quad , \qquad i = 1, 2 \\
d_i &= \sum_{j=1}^{k}{}' \frac{1}{t_j - t_i} \quad , \qquad i = 0, 2, \ldots, k \\
e_i &= \prod_{j=1}^{k} (t_j - \tau_i)^2 \quad , \qquad i = 1, 2 \\
f_i &= \prod_{j=1}^{k}{}' (t_j - t_i)^2 \quad , \qquad i = 0, 2, \ldots, k
\end{aligned}
\tag{9}
$$

where the prime indicates the usual sum (product) with $j = i$ missing.

We can then write the solution to (1) as

$$
\begin{aligned}
L_1 &= -e_1 \\
L_2 &= -2\, c_2\, L_3 \\
L &= L_3 \\
E_j^i &= 2\, d_i + \frac{1}{\tau_j - t_i} \quad , \qquad j = 1, 2 \\
E_3^i &= E_2^i + \frac{1}{\tau_2 - t_i} \\
U_i &= \frac{1}{(\tau_2 - t_i)^2 f_i} [1 + 2(\tau_2 - t_i)\, c_2] \\
V_i &= -\frac{1}{(t_i - \tau_1)} \frac{e_1}{f_i} \\
Q_i &= \frac{1}{(\tau_2 - t_i)^2 f_i} [E_3^i + 2(\tau_2 - t_i)\, c_2\, E_2^i] \\
R_i &= -V_i\, E_1^i \\
\left.\begin{aligned} a_i &= Q_i L + R_i \\ hb_i &= u_i L + V_i \end{aligned}\right\} & \quad i = 1, 2, \ldots, k \\
h\beta_2 &= \frac{L}{e_2}
\end{aligned}
\tag{10}
$$

and the solution to (2) as

$$\frac{1}{M_1} = \frac{1}{\tau_1} - 2c_2$$

$$\frac{1}{M_2} = \frac{1}{\tau_2} - 2c_2$$

$$\frac{1}{K} = -\frac{1}{f_o}\left[\frac{1}{t_o - \tau_1} - \frac{1}{t_o - \tau_2} + \frac{M_1}{(t_o - \tau_1)^2} - \frac{M_2}{(t_o - \tau_2)^2}\right]$$

$$h\bar{b}_i = \frac{KM_i}{(\tau_1 - t_o)} \frac{1}{e_i}, \quad i = 1,2 \tag{11}$$

$$hB_i = \frac{K}{(t_i - t_o)f_i}\left[\frac{1}{t_i - \tau_1} - \frac{1}{t_i - \tau_2} + \frac{M_1}{(t_i - \tau_1)^2} - \frac{M_2}{(t_i - \tau_2)^2}\right]$$

$$A_i = \frac{-K}{(t_i - t_o)f_i}\left[\frac{1}{t_i - \tau_1} - \frac{1}{t_i - \tau_2}\right]\left[\frac{M_1}{(t_i - \tau_1)^2} + \frac{M_2}{(t_i - \tau_2)^2}\right]$$

$$+ hB_i\left[2d_i - \frac{1}{t_i - \tau_1} - \frac{1}{t_i - \tau_2} - \frac{1}{t_i - t_o}\right].$$

We see from (10) that we have a free parameter L and that the free interval parameter τ_1 appears in both solutions. We also note that it is implicitly assumed that $\tau_i \neq t_j$.

Now that we have a general solution to the scheme represented by (1) and (2), we proceed in much the same fashion as Butcher to obtain schemes for the k = 1, 2, 3, 4 cases which are Examples 1, 2, 3, and 4, respectively, as presented in Appendix III.

Tables VI, VII, VIII, and IX give, for each of these examples, all the parameters plus the harmonics of the intermediate approximation expanded about its local origin and the final approximation expanded about the origin. For example, referring to Table VI which treats Example 1, we have the interval parameters $B_1, \ldots, B_7$; the harmonics $B_{16}, \ldots, B_{22}$ of ξ_1 expanded about t_o which are also the harmonics of ξ_{2i-1}, i=2,3,..., expanded about $t_{2(i-1)}$. Note that here we simply count

the points starting from t_o and make no distinction between t and τ_1, a distinction which was convenient in deriving a solution but is not convenient in general. Next come the scheme parameter values, B_{23}, ..., B_{29} where we give also the scheme in which they are used. Finally, we give the results of substituting the parameter values into the equations that represent conditions A and B as given in Chapter III. As output from RKMI, condition A equations should evaluate to zero; condition B should evaluate to the Taylor harmonics of $\xi(t_o)$ expanded about the origin. Since the origin was chosen at t_o, these too are zero. And, finally, we have the principal error term coefficients.

The k = 1 case is, of course, stable. This is not, however, necessarily true for the k = 2, 3, 4, ... cases and we give for these cases the roots of the stability polynomial associated with the corrector. For example, in Table VII, the polynomial is

$$z^2 - B_{28}\, z - B_{29} = 0$$

and thus appears as

$$A_3 z^2 + A_2 z^1 + A_1$$

in the stability check.

For Examples 2, 3, and 4, we have presented two values of $t_1 = B_1$. The first value is the usual one where t_1 is in the interval between ξ_2, the last approximation calculated, and ξ_o, the next approximation we wish to obtain. The other value lies outside this interval and is presented mainly for comparison. It will be seen that in all cases the principal error terms are considerably larger for this non-standard choice.

For each of the examples, we present a graph of the principal error

term coefficients versus the free interval parameter t_1. Where applicable, we also give the stability range of the corrector. These are given in Figures 1 - 10 of Appendix III. An examination of these graphs shows why we have chosen the t_1 values that are presented. We note that the values of t_1 are given as unreduced fractions even though they can be reduced. This is purposefully done to indicate the resolution. For example, in Figure 2, we have that -56/256 is within 1/256 of the minimum of the maximum root.

For each scheme presented, there is an analogous scheme given by Butcher. We note that the only difference between our examples and Butcher's is the scheme. That is, we can, simply by changing the scheme and no other data, use RKMI to generate either our examples or Butcher's. Thus, his work can serve as a check on the validity of the results of RKMI. We have run the k = 1, 2, 3, 4, cases that he presents and the coefficients he presents all check. Since our basis functions are different from his, our error terms will have different numerical values; however, when error differentials E are used, we should have the same error terms and this, indeed, is the case for k = 2 which the author checked in this fashion. In Table XIII, we present a summary of the principal error term coefficients for each of our examples and the analogous case of Butcher.

We have presented in Tables IV and V the complete generation of Example 2 using RKMI. The format of the input and output is identical to RK4 and the description given previously should suffice for understanding these tables.

In order to correctly use the generalized Runge-Kutta schemes, it is necessary to start correctly. In particular, a start in which all

approximations match a Taylor's series to a certain order is not, in general, the correct start for these methods. If we refer to Table XIV of Appendix I which gives the approximation harmonics, we see that the ψ_2 term has $F^k \psi_o^o$ where ψ_o^o refers to the harmonics of ξ_1 about the origin. However, referring to Figure 1 of this chapter, we see that the ψ_o harmonics of ξ_1 are the principal error terms of ξ_1 and there is absolutely no immediately apparent reason why these should be Taylor harmonics. As a matter of fact, for Example 2, it turns out these are very close to Taylor harmonics and it would be very enteresting to know whether or not the ψ_o term of ξ_1 is a Taylor harmonic.

To obtain a start for Example 1, we can proceed as follows: Knowing ξ_2, we use an elementary Runge-Kutta start to obtain ξ_3

$$\bar{\xi}_3 = \xi_2 + h\, \bar{a}_{12}\, X_2$$

$$\xi_3 = \xi_2 + h\, [a_{12}\, X_2 + a_{13}\, \bar{X}_3] \tag{12}$$

where the coefficients are obtained from

$$\begin{aligned} \bar{a}_{12} &= \bar{\theta}_3 - \theta_2 \\ a_{12} + a_{13} &= \theta_3 - \theta_2 \\ a_{12}\, \theta_2 + a_{13}\, \bar{\theta}_3 &= \alpha - \theta_2^2/2 \end{aligned} \tag{13}$$

with $\theta_2 = -1$

$\theta_3 = \theta_2 - 35/128$

$\alpha = B_{16}$ Table VI, Appendix III

$\bar{\theta}_3 =$ a free parameter

and h is the step size $t_o - t_2$ used in the Example 1 scheme. Now that we know ξ_3 and ξ_2, we can proceed using the scheme as given in Table VI. We could also proceed in an alternate fashion. Use an RK3 start with interval h to obtain $\xi_o - \xi(t_o) = O(h^4)$. Then find

$$\xi_1 = g_{12}\,\xi_2 + h\,[a_{12}\,X_2 + a_{10}\,X_o] \tag{14}$$

to find ξ_1 where the coefficients are obtained from

$$\begin{aligned} g_{12} &= 1 \\ g_{12}\,\theta_2 + a_{12} + a_{10} &= \theta_1 \\ g_{12}\,\frac{\theta_2}{2!} + a_{12}\,\theta_2 + a_{10}\,\theta_o &= \alpha \end{aligned} \tag{15}$$

with

$$\theta_1 = -35/128$$

$$\theta_2 = -1$$

$$\theta_o = 0$$

$$\alpha = B_{16} \text{ of Table VI, Appendix III}$$

and h is the step size $t_o - t_2$ used in the Example 1 scheme.

We have used the latter starting scheme for Examples 3 and 4. In Example 3, we assume that $\xi_i - \xi(t_i) = O(h^{2k+2})$ for i = 2, 4, 6. ξ_6 is originally known and any sufficiently accurate starting scheme may be used to obtain ξ_4 and ξ_2. Once these values are known, we have sufficiently many points to obtain the intermediate point ξ_3. The scheme and coefficient set are given in Table XI, Appendix III. We obtain an intermediate approximation ξ_1 and a final approximation $\xi_3 = \xi_o$. Once

this is done, there are enough known values to continue with the regular scheme. We note that in reality the scheme to obtain $\xi_3 = \xi_0$ is the same as the regular scheme presented in Table VIII; only the parameter set is different. We also check that the starting predictor is stable. The start for Example 4 is the same, except that we have one more point to obtain before obtaining ξ_3.

The starts presented here for these examples are, in general, not the best. There is no question that one can find more efficient starts. These are presented to illustrate that some thought must be given to calculating the intermediate points correctly when starting. The obtaining of efficient, correct starts for generalized Runge-Kutta schemes needs to be investigated thoroughly as more schemes are obtained. We also point out that the same problems must be considered when changing step size.

In Examples 3 and 4, we have not actually given any explicit means of obtaining the approximations at the major points when starting the scheme. We do, however, give in Table X a complete start for Example 2. In order to do this, we proceed as in the start for Example 1. Assuming that ξ_4 is the initial value, we use coefficient set 1 to obtain an RK4 type start for the intermediate point ξ_0. We now set $\xi_6 = \xi_4$, $\xi_5 = \xi_0$ and use coefficient set 3 to obtain the next approximation ξ_0 at a major point. Then we set $\xi_4 = \xi_6$, $\xi_3 = \xi_5$, $\xi_2 = \xi_0$. There are now enough approximations to continue with the scheme as presented in Table VII.

We note first that we could have saved one substitution if we had used RK5 start for the approximation ξ_2 and then proceeded as in Examples 3 and 4 to evaluate the intermediate point. We did not do this because we wished to illustrate a point that can stand further investigation. When trying to obtain RK starts for higher order methods, one

always eventually obtains too many equations to solve effectively. One way of avoiding this is to work with approximations for which the order of accuracy is known to be greater than 1. This effectively increases the value of ℓ in the tables of Appendix I. As the value of ℓ increases, we can see from those tables that the equations that we need to solve are subsets of the equations obtained for lower values of ℓ. The manner in which this process takes place must be carefully examined in each case of interest, but it appears that by choosing a good strategy, we can effectively "boot strap" our way to higher orders. The subject needs much more work, but the possibility does seem to exist. For our starting scheme of Example 2, we have used ξ_5 in coefficient set 3 and an RK5 type start. However, since ξ_5 and ξ_6 both had a sufficiently high order of accuracy, we did not need to solve all the RK type equations. Of the 16 equations appearing in Table XIV, with k = 0, we solved only ψ_0, ψ_1, ψ_3, ψ_5, ψ_8, ψ_{10}, ψ_{12}, ψ_{14}. The solution for these automatically satisfies the remaining equations.

Another point to keep in mind is that the coefficients for the set of examples that we considere here are rational numbers. It is true that we choose L to satisfy the ψ_2 equation; however, L enters linearly in that equation. It would be nice to have these coefficients presented as rational numbers, but because of the fact that we do have a rather involved calculation when obtaining L, this has not been attempted.

What is more interesting is to establish whether the free parameter B_9 appearing in the RK4 type start is in reality $B_9 = B_8$. This parameter is a polynomial root and need not be rational. We would also like to know whether the principal error term of ξ_1, the intermediate point of these examples, is or is not a Taylor harmonic. From our calculations,

we can only say it is quite close in Example 2 for the values of t_1 that we have given here.

Since all our results are presented in floating point format, there is a question of significant figures. The numerical calculations were performed on the University of California, Lawrence Radiation Laboratory's CDC 6600 Computer located at Berkeley, California. We can expect about twelve significant figures for single precision. However, a close examination of the numerical results leads to the conclusion that, in reality, we may have only 10 - 11 significant figures. There is a feeling that the corrector coefficients are well calculated to 12 significant figures, but that the predictor coefficients may well have only ten significant figures of accuracy. It is possible that accuracy has been lost in the calculation of the correct value of L or that accuracy is lost in the calculation of the coefficients themselves. In the interest of accuracy, we have not, however, transcribed the output and present it as it was obtained.

We have not, at the present time, had an opportunity to evaluate these schemes on any examples. This is, of course, necessary before coming to any conclusion about their worth. At the moment, we can only say that for the same order of accuracy they require one less substitution than Butcher's, thus putting them in the class of predictor corrector schemes, and that they have a stability range similar to, but much more restricted than, his work. It is, in fact, questionable whether the five step scheme has a stable range.

This chapter deals with examples and yet the reader will note that we have presented no examples using the results of Chapters II, IV, or V. The reasons for this are twofold. The first and most obvious one is

space requirements. It requires a fair amount of discussion to present such examples and we wish to limit our presentation to a reasonable size. The second is that our examples have not actually been obtained in this fashion. Since the schemes can be generated by RKMI and the output so obtained is almost immediately available for use in other programs, we have not generated our examples "by hand" and then programmed them. This does not, however, mean that these results have not been useful. They have proved invaluable in that they allow us to easily check the validity of the program generated output and they provide an insight into the structure of the equations we are dealing with. It is in this manner that the results of those chapters have been used. Of course, not everyone has computing facilities at their disposal and those results should allow an easy investigation of some rather complicated schemes.

In addition to the results presented here, we have used the program RKMI to generate a number of well-known schemes on which we could check the results. These are the classical RK1, RK2, RK3 schemes using the coefficients as presented in Ceschino-Kuntzmann and the schemes of R. DeVogelaere[5] and R. E. Scraton[6] which can truly be considered a generalized RK scheme since they use data from an intermediate point. However, since in these two latter schemes the intermediate point approximations have an order of accuracy only one lower than the final approximation, these coefficients are derivable by finite difference methods.

We would like to emphasize that we have presented a wealth of raw data in this work. A sincere attempt has been made to eliminate mistakes and sheer blunders, but it is always necessary to somehow find checks

for the results obtained. With regard to the Examples 1 - 4 that we have presented here, we have that

1) The solution derived by means of residue theory yields coefficients that satisfy the parameter equations as generated by RKMI.

2) A simple variation of the scheme given from RKMI gives Butcher's schemes and his coefficients satisfy the parameter equations.

3) The starting schemes for Examples 3 and 4 are simply a re-interpretation of the main schemes for those examples and require no new generation.

4) The starting scheme for Example 2 is obtained from the RK4 example, the scheme is identical; we simply match a different set of coefficients. Since we include the error term in the RK4 example, we can also use those results to obtain the RK5 type start. This requires a different scheme for RKMI, but no other new data. The RK4 scheme, as generated by RKMI, checks completely on known coefficients.

5) At every step, all parameters have been substituted into the parameter equations to check whether a solution has, indeed, been found.

6) The results can be used to integrate known polynomials to check the order of accuracy; this last step has not been performed and, of course, should be when time permits.

VIII. COMMENTS

The work that we have presented in the preceding chapters and the accompanying appendices, though rather voluminous, can only be considered as an introduction to the investigation of generalized RKF integration schemes. We have, hopefully, laid some foundations; our work is characterized as much, if not more so, by what has been omitted than what has been presented. We shall give in this chapter a few comments on some of these omissions and shall also indicate some directions that future work might take.

We note first that although we have devoted a large amount of effort to the development of a descriptive formalism which can be directly reflected, preserved, and implemented by means of suitably defined procedures, the class of schemes we treat is, in reality, not large. One need only look at a reference such as Ceschino-Kuntzmann[2] to realize that there are many we have left out. However, their work also suggests that it should be possible to include this work; that is, it should prove possible to considerably extend the class of schemes treated and include most common integration processes. This is probably most easily done along the lines of the work of Chapter III and hopefully one could also eventually arrive at a global viewpoint similar to that given by the approximation or error harmonics.

This leads immediately, however, to another topic that we have not treated in sufficient detail; this is the classification of schemes. We have given what we believe to be a fairly adequate classification of the relevant scheme parameters for generalized RK schemes. It immediately becomes evident, however, that we must obtain a good systematic classification for the generalized RKF schemes if any serious investigation

and comparison of schemes is to be undertaken. We point this out because not only is a good, systematic classification necessary for practical reasons, it also leads naturally to (and from) a good scheme definition.

However, these problems can probably best be resolved by first obtaining some solid practical experience using the results so far obtained to carry out a thorough investigation of the schemes that are contained in the work presented here. It is the author's opinion that the present work has progressed to a stage where it should be thoroughly used and evaluated. Our work presents results that are usable; there is a wealth of unbroken ground, as well as known examples, upon which it can be tested and carrying out such an investigation will no doubt suggest modifications and improvements that will better help us understand the theoretical aspects of the generation and solution of the parameter equations associated with these integration schemes.

We might comment that we are arriving at the point where integration schemes can be generated and the associated parameter defining equations completely solved simply by presenting a rather small set of data that characterizes the process. How well this can be done depends to a high degree on understanding the equations that are created when we generate a scheme and, thus, it is necessary to actively pursue not only the obtaining of a specific scheme, that is, the local approach, but also the global approach that yields the total view of our problem. If this attack could be carried to a successful completion, one can see that there would be the possibility of tayloring the scheme to fit the differential equations being integrated.

This latter comment points out the fact that we have made no

mention of two important aspects of these schemes. One is the subject of stable schemes. With regard to this, we comment that any scheme search should be carried out within the context of stability, however, very little has been done or said about the stability of generalized RKF schemes. The other subject is a global error analysis. R. DeVogelaere has suggested to the author the possibility of handling that problem in such a fashion that it too could be treated by means of suitably defined procedures.

One aspect of our work that requires some explanation is our use of tables for the quantities that are used in the program RKMI. This is really not the way to do it. All the quantities of interest can, and should be machine generated. The use of tables externally input is subject to too many errors and is only a temporary means of obtaining these quantities. One of the first, parallel tasks of any investigation of generalized RK schemes using RKMI is to program not only the generation of there quantities, but also the checking of the generated quantities. This can be done and will relieve the user of an enormous amount of needless preparatory work.

In the same spirit, the generic definition given in Definition 3, Chapter IV, should be built into a procedure so that we can investigate at will the quantities so laboriously tabulated in Appendix I.

The examples that we have presented in Chapter VII and Appendix III can only be considered to be illustrative of the type of work that can be done with the results presented. It is immediately obvious that no Frey type schemes have been presented.

RKMI is, we believe, quite capable of handling these schemes and also schemes for higher order differential equations. Their omission

here has been caused mainly by the lack of space and time to adequately treat their generation. We trust that in the future there will be an opportunity to undertake the scheme investigation that appears as the next natural step of this work.

ACKNOWLEDGMENTS

This work derives from a suggestion by Professor R. DeVogelaere that I investigate integration methods that make use of information from the past and also the first derivative DX after the fashion of Frey. During the past four years that we have worked on this project, various examples of schemes with memory have appeared. Because of his foresight in insisting that this work be as general as possible, we are able to handle such schemes as special cases. I must thank him for suggesting such a rich field of study and for his untiring patience and understanding in dealing with the problems of developing this work.

Thanks are also due to the Mathematics and Computing Division of the Lawrence Radiation Laboratory. Without their liberal support, this work would not have been brought to completion. Most of this work was written there. I wish to acknowledge my indebtedness to Mrs. A. Rutan who not only has patiently typed this manuscript, but also has drawn the figures included in the text.

The ALGOL program was developed using the University of California Computer Center facilities and was supported by funds made available by the Committee on Research at the University of California, Berkeley, and by the State of California. The actual numerical calculations were performed using the computer facilities of the Lawrence Radiation Laboratory at Berkeley, California.

All of this work is dedicated to my wife, Tsuru, for her patience and understanding.

The work presented here was partially supported by the United States Atomic Energy Commission.

Appendix I

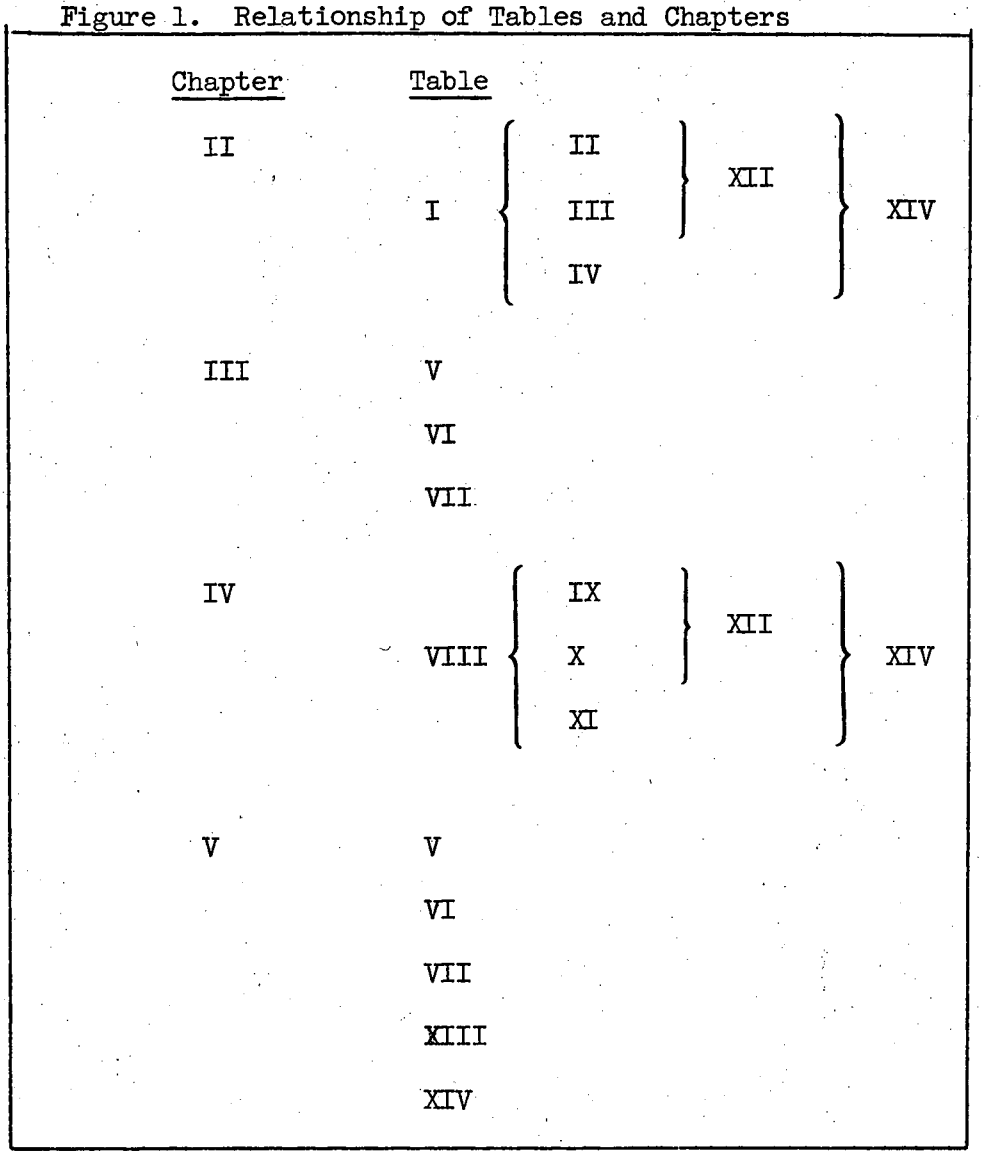
TABLES

Appendix I is devoted to the presentation of tables of which the items represent the various quantities discussed in Chapters II - V. We give below a short description of each of these tables. The first set of tables, Tables I - IV, pertains to Chapter II; the next set, Tables V - VII, to Chapter III. Tables VIII - XII come from Chapter IV, while Tables XIII and XIV come from Chapter V. There is a dependence among the tables that we shall describe before proceeding to the individual table descriptions. Tables II, III, and IV can be derived from Table I. Tables IX, X, XI are derivable from Table VIII. Tables I and VIII are in a certain sense equivalent. They represent the same items in that for $k = 0$ Table VIII would reduce to Table I, except for the fact that we have left out the terms with more than one E; whereas, in Table I, we have extended the table to the terms with two E factors.

Table XII is equivalent to Tables IX and X since $\pi = \alpha\gamma$. Table XIV, likewise, is equivalent to IX, X, XI, and XII since, if we combine the quantities to obtain the parameter equations, we will arrive at the results presented in Table XIV. We give below in Figure 1 the relationship of the tables and the chapters in which they are used or defined.

Tables I - IV will be discussed first. These tables are used to tabulate the various quantities mentioned in Chapter II. Table I is a generic table from which Tables II, III, and IV can be generated. We shall show how this is done shortly. Table II gives the derivative harmonics α that are defined in Definition 6 of Chapter II; Table III gives the polynomial weights γ (Definition 8); and Table IV gives the elementary polynomials Γ (Definition 7). Referring to Condition B,

Figure 1. Relationship of Tables and Chapters



Equation (62), Chapter II, we see that these are precisely the quantities needed to write down the non-linear parameter defining equations for any generalized RK scheme as developed in that chapter. It should be kept in mind that Chapter II has as its generalization the work of Chapter IV and that the corresponding tables for that chapter (in particular, we refer to Tables VIII, IX, X, and XI) should reduce when $k = 0$ to the Tables I, II, III, and IV. It is realized that there is a certain amount of redundancy in the work as presented here; however, we have presented only a small, but representative, section of the tabulated work and the redundancy along with the interrelations of the tables should help provide a check on the material.

It is possible to give a rather complete description about how to systematically construct the generic table, Table I, and such a discussion is necessary if one considers the generation of these quantities using suitably defined procedures. Since we have not built such procedure, we shall avoid any lengthy description of that process. However, it is very helpful to visualize this table as a two-dimensional array A_{ij} where i = order, j = rank, and the A are blocks of elements containing the items of given order and rank. Then, referring to Theorem 1, to Equation (16'), and to Equation (14) of Chapter II, we are able to see why the various quantities in that chapter have the same recursive definition. We shall limit ourselves here to the use of these tables.

In Table I, every item has a set of quantities associated with it that uniquely specify that item. These are the order r, rank R, degree s, value of ℓ (see Equation (2), Chapter II), and sequential position a of the item within the set of all items of given order, rank, degree, and f value. Thus, for any item ψ in Table I, we have

$$\psi = \psi[\ell, R, r, s, a].$$

It will be seen that these quantities are tabulated in the first few columns of the table. We have, however, specified the rank as R in the upper left corner of each block of order R and we have not sequentially ordered all the functions, only those with rank = order and for these, we have also found it convenient to sequentially count the functions. For example, the reader will find

$$\psi_{10} = \psi[\infty, \ell + 4, \ell + 4, 1, 2]$$

and he will find that

$$\psi[\infty, \ell + 5, \ell + 4, 1, 2] = E\, C_2^1\, C_1^1\, C_1^0$$

has not been give a sequential count.

In order to have before us a concrete example of the items of Table I, we imagine that the ψ represent the weighted differentials W of Definition 3, Chapter II, and also recall that we gave there a short introduction to this table. The first item ψ_0 is then the weighted differential $W[\infty, \ell + 1, \ell + 1, 0, 0]$ of rank $\ell + 1$, order $\ell + 1$, degree zero, position zero, and this function will appear for all values of ℓ which is indicated by setting ∞ into the ℓ position. Keep in mind that if we picture a table for each value of ℓ, then all items of degree zero or degree 1 will correspond identically for each table; that is, we could overlay the tables and see no change. This is not true for items of degree greater than one. Thus, ∞ as an ℓ value implies that the item appears for all values of ℓ, whereas, a numerical value such as 1, 2, etc., imply that this item appears only for that particular value of ℓ. We also point out that the blank spaces in the first three columns of Table I are to be filled in by the number appearing above in that column.

See, for example, our reference above to ψ_{10} where we have supplied the missing quantities.

We have as our first item ψ_o, the item of lowest order and degree; then we write ψ_1, the next item of degree zero; and then, ψ_2 which is derived from ψ_o. Once we have all the items of a given rank and order, here rank $\ell + 2$, order $\ell + 2$, we write the degree zero item of next higher order and rank. Note that we write all the rank R = order r items first. Thus, we have ψ_3 and then ψ_4, ψ_5, ψ_6 and we proceed onward in this fashion to as high an order as necessary. We raise the rank R by 1 by applying the E operator and then proceed again from the lowest order to generate all the function of rank R = order r + 1. Having once obtained all of these, we again apply the E operator to all of these items and proceed as before. The reader can best become familiar with these items by generating the tables himself using Definition 3 of Chapter II as a guide. He will see that one must be quite careful in obtaining the terms of degree greater than 1. We point out that the items of rank R = order r are the same as presented in Table V which proceeds to a higher order (refer to the discussion immediately preceding Definition 6, Chapter IV), and, thus, the results obtained can be verified for higher orders than given in Table I.

To use Table I, we replace the operator in Table I by the quantities defined in Chapter II and presented again in the Supplement to Table I. Using $C_1^1 E C_1^1 C_1^o$, we illustrate with one example for each case:

1) Weighted Differential W

$$C_1^1 E C_1^1 C_1^o = \sum_{j_1} R_4 f_{ij_1}^{R_4-1} D^o(D_{N_1} X \circ u_{j_1}) \sum_{j_2} R_3 g_{j_1 j_2} \sum_{j_3} R_2 f_{j_2 j_3}^{R_2-1}$$
$$D^o(D_{N_2} X \circ u_{j_3})_{N_1} \sum_{j_4} R_1 f_{j_3 j_4}^{R_1-1} D^\ell(X \circ u_{j_4})_{N_2}.$$

2) Elementary Differential A.

$$C_1^1 EC_1^1 C_1^0 = D^0(D_{N_1} X \circ u)\, D^0(D_{N_2} X \circ u)_{N_1} \cdots D^{\ell-1}(X \circ u)_{N_2}$$

where we choose for A any function from the set of all functions of order r since all functions are the same.

3) Weighted Polynomial Φ.

$$C_1^1 EC_1^1 C_1^0 = \sum_{j_1} R_4 f_{ij_1}^{R_4-1} \theta_{j_1}^0 \sum_{j_2} R_3 g_{j_1 j_2} \sum_{j_3} R_2 f_{j_2 j_3}^{R_2-1} \theta_{j_3}^0$$

$$\sum_{j_4} R_1 f_{j_3 j_4}^{R_1-1} \theta_{j_4}^{\ell} .$$

4) Derivative harmonics α. (Table II).

$$C_1^1 EC_1^1 C_1^0 = \frac{(R_4-1)!}{0!} \frac{1}{(R_3)!} \frac{(R_2-1)!}{0!} \frac{1}{(R_1)!} .$$

5) Elementary Polynomial Γ. (Table IV).

$$C_1^1 EC_1^1 C_1^0 = \sum_{j_1} f_{ij_1}^{R_4-1} \theta_{j_1}^0 \sum_{j_2} g_{j_1 j_2} \sum_{j_3} f_{j_2 j_3}^{R_2-1} \theta_{j_3}^0 \sum_{j_4} f_{j_3 j_4}^{R_1-1} \theta_{j_4}^{\ell} .$$

6) Polynomial Weight γ. (Table III).

$$C_1^1 EC_1^1 C_1^0 = R_4\, R_3\, R_2\, R_1$$

where, in each case, $R_i = \ell + i$ is the rank of the item.

We remind the reader that care must be taken when interpreting quantities of degree greater than 1. Consider, for example, the weighted differential

$$W_i = C_1^2 C_1^0 (EC_1^0) = \sum_{j_1} R_5 f_{ij_1}^{R_5-1} D^0(D_{N_1N_2} X \circ u_{j_1})(\sum_{j_2} R_1 f_{j_1j_2}^{R_1-1} D^{\ell}(X \circ u_{j_2})_{N_1})$$

$$(\sum_{j_3} R_2 g_{j_1j_3} \sum_{j_4} R_1 f_{j_3j_4}^{R_1-1} D^{\ell}(X \circ u_{j_4})_{N_2})$$

and we note that both the second and third factors depend on j_1. This is seen in Equation (20) of Chapter II.

We have written these quantities out in detail here once to illustrate the notation; however, in general, it is not necessary to do this except with the elementary polynomials and then only when we wish to write down the equation for a particular scheme.

For the W, A, and Φ, it is more profitable to utilize Table I only as a pattern establishing table. For the α and γ, it is easy enough to generate them from their immediate factors rather than re-expressing these factors in terms of the succeeding factors and the same is true for the generation of the elementary polynomials in Table IV.

As a check on our work, we note that each of the quantities W, A, Φ, α, γ, Γ has a separate definition. However, we know that $W = \Phi A$, $\Phi = \gamma\Gamma$, and will subsequently see that $\frac{\alpha\gamma}{R!} = \pi$ where π are product coefficients defined in Table XII.

In Table II, we note that we have consistently used an abbreviated factorial type notation $a^i = a^{(i)} = (a + \ell)(a + \ell - 1) \ldots (a + \ell - i + 1)$ along with the fact that we also have consistently left out the value of ℓ for these terms. Thus, $3^1 2^1/2! = (\ell + 3)(\ell + 2)/2!$ which could just as well be also written as $3^2/2!$.

Table III also uses this notation; thus, $4^3 = (\ell + 4)(\ell + 3)(\ell + 2)$.

Table IV has as its first column $\phi^j = \theta^{\ell+j}$ and the Taylor harmonics

we are trying to match are $\frac{\theta^{\ell+j}}{(\ell+j)!}\alpha$ where α is the derivative harmonic corresponding to that position. For example, for the 4th harmonic, we have $(\ell + 2)\frac{\theta^{\ell+3}}{(\ell+3)!}$. The G and F are essentially the E and C of Table I. We have, however, appended the appropriate order to F as a subscript; thus, F_i has order $r = \ell + i$. We have that

$$F_3 G\, F_2\, F_1\, \phi^{\circ} = \sum_{j_1} f_{ij_1} \sum_{j_2} g_{j_1 j_2} \sum_{j_3} f_{j_2 j_3} \sum_{j_4} f_{j_3 j_4}\, \theta^{\ell}_{j_4}.$$

It will be recalled that exact points have a special set of scheme coefficients and these are indicated in Table IV. We have used the order r as a subscript for the F instead of the rank R since it suffices for the purpose of identifying the f_{ij} for exact points. This turns out to be the case since $g_{ij} \equiv 0$ for such points. The use of r gives us then a simpler table that can more easily be checked for mistakes.

The use of these tables to generate the equations given by Condition B, Equation (62), Chapter II, is quite easy. We simply multiply the corresponding table entries of Tables II, III, and IV together to obtain $\alpha\gamma\Gamma$, divide by R! where R is the corresponding rank, sum up a row of given order and set it equal to the corresponding Taylor harmonic multiplied by the derivative harmonic of that rank and order.

There is, therefore, no problem to obtain the parameter defining equations. However, they can be obtained even more easily using approximation or error harmonics presented in Table IX. This result was not at first apparent and, thus, the need for Tables I - IV. They are presented here, however, because they furnish explicit examples of the quantities discussed in Chapter II and are possibly in themselves of some interest if one wishes to study the development of the equations, rather than generate schemes.

Tables V - VII pertain to the work of Chapter III. Again, our first table of this set is, in a sense, pattern establishing. We have defined here the differentials A of Chapter III, Definition 2. These quantities depend on the value of ℓ, the order r, degree s, index set $k_1, \ldots, k_s$, and are sequentially positioned within that set.

The first five columns of Table V give this information. We have actually included an extra piece of information in the index set $k_1, \ldots, k_s$. For example, ψ_{19} is of degree 1 and should thus have only one value of k, $k_1 = 0$. However, we have added also the values of k that appear in the factors of ψ_{19} that are not equal to 0. Therefore, the first s values of $k_1, \ldots, k_s$ are the k values and the remaining set of values are the other values of k that appear in the factor of ψ. In this way, we can see at a glance what value of k ψ actually depends on. This is rather useful because if we were to delete all ψ terms depending on k = 1; that is, we consider a first order differential equation, then we would certainly delete ψ_{20}, but might forget to delete ψ_{19}.

The items tabulated in the column definition are the same A in the same notation as those given in Chapter III. We note, however, that since for degree zero terms, we have r_o = order - 1 and since the order depends on ℓ, we have used not r_o, but $r_o - \ell$. For example {3} is actually $\{\ell + 3\}$. This is not true for terms of degree higher than 0. Thus, $\{3, \psi_o\}$ is really $\{3, 0; \psi_o\}$.

The rest of the columns give the sequential count of ψ where k is in the closed set heading that column. For example

$$\psi_{22} \text{ is } \psi_8 \text{ if } k \in [1, \infty], \psi_5 \text{ if } k \in [2, \infty], \text{ etc.}$$

Note that this pertains also to the column definition. In particular, we change both the k set and the reference indexing column when order-ℓ

becomes 6. The reason for this is obvious, there are too many functions. However, it is easy enough to fill in this table if that proves desirable as indeed it may for higher order Runge-Kutta starting schemes.

One of the most interesting things about Table V is that it makes quite evident the tremendous simplification that takes place when $k = 0$, that is, when the term of highest derivative, is missing from the right-hand side of the differential equation. The disappearance of the $k = 1$ term again simplifies the table in the same fashion and, of course, this continues as we move up in order.

Tables VI and VII are numbered in the same fashion as Table V. The harmonics, as labled in Table VI, are those of Definition 4, Defintion 5, and Definition 7 of Chapter III. It is these harmonics that the procedure RKMI uses. These derivative harmonics are the same as those with rank R = order r in Table II provided we limit ourselves to $k \in [0, 0]$. The factorial type notation is again used $a^i = (\ell + a)(\ell + a - 1) \ldots (\ell + a - i + 1)$ for the derivative harmonics.

The substitution harmonics are presented exactly as defined in Definition 5. We have, however, omitted the index r specifying the order. Thus, $\beta_{16} = \alpha_{0,6}$ is the 16th harmonic of $x = X(u + T)$ where α_{06} is the $m = (6 + 0 \cdot n)$th component of T; that is, the 6th component of the $k = 0$ set.

The multiplication harmonics are defined exactly as in Definition 7; we have that γ corresponds to T and α corresponds to S. Again, the order r index is omitted.

Table VII gives us the translation harmonics corresponding to the differentials A. We indicate their use at the top of this table. We have factored out of each block of harmonics $t^j/j!$ and this is

indicated by the (j) at the left corner of each block. We have also used the factorial type notation $a^j = (\ell + a)(\ell + a - 1)\ldots(\ell + a - j + 1)$ for those quantities that depend on ℓ. This table is obtained from Definition 9 of Chapter III.

It is always nice to have checks for the tables that are constructed and, to a certain extent, this is possible. We note the following:

The first column of Table VII should be the derivative harmonics β presented in Table VI. We also have the following check sum for Table VII.

Rule: Take any row i with element γ_{ij}. Factor out the term $\frac{h^j}{j!}$ (that is, use the table entries as presented in Table VII); then

$$\sum_{j \in S_r} \gamma_{ij}\,\gamma_{jo} = \gamma_{io} = \beta_i$$

the translation harmonic corresponding to ψ_i. Note that j is restricted to the set of coefficients with order r. This result is true for all orders in a given row.

For example,
$$\gamma_{90}\,\gamma_{00} = \beta_9$$
$$\gamma_{91}\,\gamma_{10} + \gamma_{92}\,\gamma_{20} = \beta_9$$
$$\gamma_{93}\,\gamma_{30} + \gamma_{94}\,\gamma_{40} + \gamma_{95}\,\gamma_{50} + \gamma_{96}\,\gamma_{60} + \gamma_{97}\,\gamma_{70} = \beta_9.$$

The elements given in Table VII should be orthogonal as stated in the corollary to Theorem 9 in Chapter III. It is possible to check not only the table, but its use by performing an expansion $\xi_i = \Sigma\, \alpha_i\, A_i$, translating to obtain $\xi_i = \Sigma\, \bar{\alpha}_i\, \bar{A}_i$ and then substituting Taylor harmonics for α_i; that is, assume $\xi_i = \xi(\theta_i)$ is an exact solution and has a Taylor's expansion. We then know the expansion harmonics of ξ with respect to both origins and can evaluate $\bar{\alpha}_i(\alpha_i)$ to see if the results are correct.

The substitution harmonics prove slightly more troublesome. The derived equations can be checked on known examples. This is to be considered a necessary check, but certainly not sufficient. Another check is to realize that if all points of a scheme are exact solutions $\xi(\theta_i)$, then each set of harmonics that is constructed should reduce to Taylor harmonics and whether this is, indeed, the case can be easily check numerically by a machine.

Tables VIII - XII are obtained from the work of Chapter IV. The work of that chapter is a generalization of the work of Chapter II. Therefore, the tables here are simply a generalization of Tables I - IV and, thus, do not require a very detailed explanation. We note that an explicit dependence on k has been introduced. Since the fact that $k \in P = \{0, \ldots, p - 1\}$, there can be a profusion of terms in the equations for higher order equations. We have limited the number of terms present in the tables by tabulating only those for which one E appears and omitting those for which there is more than one E.

The application of the C and E operators to obtain the quantities given in the supplement to Table VIII is done in the same manner as previously with Table I. However, now we must also pay attention to the value of k. We should mention that for ψ_{21}, the $(6 + k_1 + k_2)$ terms can be arrived at as follows. Consider $k \in P = \{0, \ldots, p - 1\}$. For each k, there is a $\psi = E_o^o C_{1k} = \psi_k$. We, thus have $\psi_o, \ldots, \psi_{p-1}$. Because $A = \{0, 0, 0, A_1 A_2\}$ has $k_1 = k_2$, there are no permissible permutations. Therefore, we get new functions for the sets

$$(\psi_0, \psi_0), (\psi_1, \psi_1), \ldots, (\psi_{p-1}, \psi_{p-1})$$
$$(\psi_0, \psi_1), (\psi_1, \psi_2), \ldots$$
$$\vdots$$

(ψ_0, ψ_{p-1}), (ψ_1, ψ_{p-1}), ...

and having chosen $\psi_i\psi_j$, we cannot choose $\psi_j\psi_i$.

We have again used the factorial type notation and omitted the value of ℓ when tabulating the quantities of Tables IX and X. The elementary polynomials of Table XI are also easily understood in terms of those previously described. We should, however, note a couple of things. One is that the derivative harmonics defined in Definition 5 of Chapter IV are not what has been tabulated. We have, as indicated, factored out $(R + k)!$ and to obtain α_{Rrj} of Chapter IV, we must divide the table entry by $(R + k)!$ Also, we note that we have not tabulated the product coefficients π of Chapter IV, but instead π^{-1}.

The relations between the various quantities are given in Theorem 3 and Theorem 5 where we show that $W = \Phi A$, $\alpha\Phi = \alpha\gamma\Gamma = \pi\Gamma = H$. We see that we have the previously mentioned check $\pi = \alpha\gamma$. The use of these table entries is defined in Equation (45) of Chapter IV.

The last two tables, XIII and XIV, contain, in a certain sense, the most interesting and pleasing results. Table XIII presents the translation harmonics as they pertain to error harmonics E. There is no difference between the interpretation of this table and the previously described translation table. There is, however, a tremendous simplification of the results. These results are, in fact, so simplifed that the only check left to us is the orthogonality check; the check sum is trivially true.

Table XIV gives the approximation harmonics as defined in Definition 12 of Chapter IV or equivalently the error harmonics given in Chapter V. Our ordering is that of Table V; we simply interpret the ψ_i as either approximation harmonics or error harmonics. For any

approximation, we write its expansion in these harmonics and then make the appropriate summations as shown to the right of the table. This is, of course, nothing more than what RKMI does for RK schemes; however, we have before us here the totality of equations and can see their interdependence. Since Table XIV is closely related to RKMI, it can furnish a means by which we can check the reasonableness of the answers generated by that program. This is quite a necessary task because both people and machines make mistakes.

SUPPLEMENT TO TABLE I

NAME	OPERATOR DEFINITION R = RANK		
Ψ_i	C_j^0	$C_j^s \quad s \geq 1$	E
W_i	$\sum_{j_1} R\, f_{ij_1} D^{j+l-1}(X \circ u_{j_1})$	$\sum_{j_1} R\, f_{ij_1} D^{j-1}(D_{N_1 \cdots N_S} X \circ u_{j_1})$	$\sum_{j_1} R\, g_{ij_1}$
Φ_i	$\sum_{j_1} R\, f_{ij_1} \theta_{j_1}^{j+l-1}$	$\sum_{j_1} R\, f_{ij_1} \theta_j^{j-1}$	$\sum_{j_1} R\, g_{ij_1}$
A	$D^{j+l-1}(X \circ u)$	$D^{j-1}(D_{N_1 \cdots N_S} X \circ u)$	0
Γ_i	$\sum_{j_1} f_{ij_1} \theta_{j_1}^{j+l-1}$	$\sum_{j_1} f_{ij_1} \theta_{j_1}^{j-1}$	$\sum_{j_1} g_{ij}$
γ	R	R	R
α	1	$\frac{(R-1)!}{(j-1)!} \prod_{i=1}^{r} \frac{1}{(u_i)!\,(R_i)!^{u_i}}$	1

TABLE I

GENERIC FUNCTIONS

ℓ VALUES	ORDER-ℓ	DEGREE, S	POSITION	ψ	RANK-ℓ					
8	1	0	0	0	① C_1^0	② EC_1^0	③ $E^2C_1^0$			
8	2	0	0	1	② C_2^0	③ EC_2^0		④ $E^2C_2^0$		
		1	0	2	$C_1'\psi_0$	$EC_1'C_1^0$	$C_1'EC_1^0$	$E^2C_1'C_1^0$	$EC_1'EC_1^0$	$C_1'E^2C_1^0$
8	3	0	0	3	③ C_3^0	④ EC_3^0			⑤ $E^2C_3^0$	
		1	0	4	$C_2'\psi_0$	$EC_2'C_1^0$	$C_2'EC_1^0$		$E^2C_2'C_1^0$	$EC_2'EC_1^0$
			1	5	$C_1'\psi_1$	$EC_1'C_2^0$	$C_1'EC_2^0$		$E^2C_1'C_2^0$	$EC_1'EC_2^0$
			2	6	$C_1'\psi_2$	$E(C_1')^2C_1^0$	$C_1'EC_1'C_1^0$	$(C_1')^2EC_1^0$	$E^2(C_1')^2C_1^0$	$EC_1'EC_1'C_1^0$
8	4	0	0	7	④ C_4^0	⑤ EC_4^0				⑥ $E^2C_4^0$
		1	0	8	$C_3'\psi_0$	$EC_3'C_1^0$	$C_3'EC_1^0$			$E^2C_3'C_1^0$
			1	9	$C_2'\psi_1$	$EC_2'C_2^0$	$C_2'EC_2^0$			$E^2C_2'C_2^0$
			2	10	$C_2'\psi_2$	$EC_2'C_1'C_1^0$	$C_2'EC_1'C_1^0$	$C_2'C_1'EC_1^0$		$E^2C_2'C_1'C_1^0$
			3	11	$C_1'\psi_3$	$EC_1'C_3^0$	$C_1'EC_3^0$			$E^2C_1'C_3^0$
			4	12	$C_1'\psi_4$	$EC_1'C_2'C_1^0$	$C_1'EC_2'C_1^0$	$C_1'C_2'EC_1^0$		$E^2C_1'C_2'C_1^0$
			5	13	$C_1'\psi_5$	$E(C_1')^2C_2^0$	$C_1'EC_1'C_2^0$	$(C_1')^2EC_2^0$		$E^2(C_1')^2C_2^0$
			6	14	$C_1'\psi_6$	$E(C_1')^3C_1^0$	$C_1'E(C_1')^2C_1^0$	$(C_1')^2EC_1'C_1^0$	$(C_1')^3EC_1^0$	$E^2(C_1')^3C_1^0$
1	4	2	0	15	$C_1^2(\psi_6)^2$	$EC_1^2(C_1^0)^2$	$C_1^2C_1^0(EC_1^0)$			$E^2C_1^2(C_1^0)^2$

TABLE I CONT.

Ψ	RANK-ℓ
0	
1	
2	
3	⑤
4	$C_2' E^2 C_1^0$
5	$C_1' E^2 C_2^0$
6	$E(C_1')^2 E C_1^0$ $C_1' E^2 C_1' C_1^0$ $C_1' E C_1' E C_1^0$ $(C_1')^2 E^2 C_1^0$
7	⑥
8	$E C_3' E C_1^0$ $C_3' E^2 C_1^0$
9	$E C_2' E C_2^0$ $C_2' E^2 C_2^0$
10	$E C_2' E C_1' C_1^0$ $E C_2' C_1' E C_1^0$ $C_2' E^2 C_1' C_1^0$ $C_2' E C_1' E C_1^0$ $C_2' C_1' E^2 C_1^0$
11	$E C_1' E C_3^0$ $C_1' E^2 C_3^0$
12	$E C_1' E C_2' C_1^0$ $E C_1' C_2' E C_1^0$ $C_1' E^2 C_2' C_1^0$ $C_1' E C_2' E C_1^0$ $C_1' C_2' E^2 C_1^0$
13	$E C_1' E C_1' C_2^0$ $E(C_1')^2 E C_2^0$ $C_1' E^2 C_1' C_2^0$ $C_1' E C_1' E C_2^0$ $(C_1')^2 E^2 C_2^0$
14	$E C_1' E(C_1')^2$ $E(C_1')^2 E C_1' C_1^0$ $E(C_1')^3 E C_1^0$ $C_1' E^2 (C_1')^2 C_1^0$ $C_1' E C_1' E C_1' C_1^0$ $C_1' E(C_1')^2 E C_1^0$ $(C_1')^2 E^2 C_1' C_1^0$
15	$E C_1^2 C_1^0 (E C_1^0)$ $C_1^2 C_1^0 (E^2 C_1^0)$ $C_1^2 (E C_1^0)^2$

TABLE I CONT.

ψ	RANK-ℓ	
0		
1		
2		
3		
4		
5		
6		
7	⑥	
8		
9		
10		
11		
12		
13		
14	$(C_1')^2 E C_1' E C_1^0$ $(C_1')^2 E^2 C_1^0$	
15		

TABLE II

DERIVATIVE HARMONICS

NOTE $a^{i} \equiv \binom{l+a}{i} i!$

ORDER-l	ψ	RANK-l																					
1	0	① 1			② 1			③ 1															
2	1	② 1			③ 1			④ 1															
	2	1			1		1	1		1	1		1										
3	3	③ 1			④ 1			⑤ 1															
	4	$2'$			$2'$		$3'$	$2'$		$3'$	$4'$												
	5	1			1		1	1		1	1												
	6	1			1		1	1		1	1		1	1		1							
4	7	④ 1			⑤ 1			⑥ 1															
	8	$3'2'/2!$			$3'2'/2!$		$4'3'/2!$	$3'2'/2!$		$4'3'/2!$	$5'4'/2!$												
	9	$3'$			$3'$		$4'$	$3'$		$4'$	$5'$												
	10	$3'$			$3'$		$4'$	$4'$		$5'$	$5'$		$5'$										
	11	1			1		1	1		1	1												
	12	$2'$			$2'$		$2'$	$3'$		$2'$	$3'$		$4'$										
	13	1			1		1	1		1	1		1										
	14	1			1		1	1		1	1		1		1								
	15	$\frac{3'2'}{(1')!2!}$			$\frac{3'2'}{(1')!2!}$		$\frac{4'3'}{(1')!}$	$\frac{3'2'}{(1')!2!}$		$\frac{4'3'}{(1')!}$	$\frac{5'4'}{(1')!}$		$\frac{5'4'3'}{(1')!2!}$										

TABLE III

POLYNOMIAL WEIGHTS

NOTE $a^{\underline{i}} \equiv \binom{\ell+a}{i} i!$

ORDER-ℓ	ψ	RANK-ℓ																				
1	0	① 1^1	② 2^2	③ 3^3																		
2	1	② 2^1	③ 3^2		④ 4^3																	
	2	2^2	3^3	3^3	4^4	4^4	4^4															
3	3	③ 3^1	④ 4^2		⑤ 5^3																	
	4	$3^1 1^1$	$4^2 1^1$	$4^1 2^2$	$5^3 1^1$	$5^2 2^2$	$5^1 3^3$															
	5	3^2	4^3	4^3	5^4	5^4	5^4															
	6	3^3	4^4	4^4	4^4	5^5	5^5	5^5	5^5	5^5	5^5											
4	7	④ 4^1	⑤ 5^2			⑥ 6^3																
	8	$4^1 1^1$	$5^2 1^1$	$5^1 2^2$		$6^3 1^1$	$6^2 2^2$	$6^1 3^3$														
	9	$4^1 2^1$	$5^2 2^1$	$5^1 3^2$		$6^3 2^1$	$6^2 3^2$	$6^1 4^3$														
	10	$4^1 2^2$	$5^2 2^2$	$5^1 3^3$	$5^1 3^3$	$6^3 2^2$	$6^2 3^3$	$6^2 3^3$	$6^1 4^4$	$6^1 4^4$	$6^1 4^4$											
	11	4^2	5^3	5^3		6^4	6^4	6^4														
	12	$4^2 1^1$	$5^3 1^1$	$5^3 1^1$	$5^3 2^2$	$6^4 1^1$	$6^4 1^1$	$6^3 2^2$	$6^4 1^1$	$6^3 2^2$	$6^2 3^3$											
	13	4^3	5^4	5^4	5^4	6^5	6^5	6^5	6^5	6^5	6^5											
	14	4^4	5^5	5^5	5^5	5^5	6^6	6^6	6^6	6^6	6^6	6^6	6^6	6^6	6^6	6^6	6^6					
	15	$4^1 1^1 1^1$	$5^2 1^1 1^1$	$5^1 1^1 2^2$		$6^3 1^1 1^1$	$6^2 1^1 2^2$	$6^1 1^1 3^3$	$6^1 2^2 2^2$													

TABLE IV

ELEMENTARY POLYNOMIALS Γ (RANK-2)

$\phi^{\alpha} \equiv \theta^{\ell+\alpha}$

EXACT POINTS HAVE $\begin{cases} g \to g_{ij} \equiv 0 \\ F_\alpha \to f_{ij} \to \begin{cases} f_{ij} = 0 & i \neq j \\ f_{ii} = \frac{\theta^i}{\ell+\alpha} \end{cases} \end{cases}$

ORDER-ℓ	ψ														
1	0	① ϕ^1	① $F_1\phi^0$	② $gF_1\phi^0$	③ $g^2F_1\phi^0$										
2	1	② ϕ^2	② $F_2\phi^1$	③ $gF_2\phi^1$		④ $g^2F_2\phi^1$									
	2	ϕ^2	$F_2F_1\phi^0$	$gF_2F_1\phi^0$	$F_2gF_1\phi^0$	$g^2F_2F_1\phi^0$	$gF_2gF_1\phi^0$	$F_2g^2F_1\phi^0$							
3	3	⑤ ϕ^3	③ $F_3\phi^2$	④ $gF_3\phi^2$		⑤ $g^2F_3\phi^2$									
	4	ϕ^3	$F_3\theta^1F_1\phi^0$	$gF_3\theta^1F_1\phi^0$	$F_3\theta^1gF_1\phi^0$	$g^2F_3\theta^1F_1\phi^0$	$gF_3\theta^1gF_1\phi^0$	$F_3\theta^1g^2F_1\phi^0$							
	5	ϕ^3	$F_3F_2\phi^1$	$gF_3F_2\phi^1$	$F_3gF_2\phi^1$	$g^2F_3F_2\phi^1$	$gF_3gF_2\phi^1$	$F_3g^2F_2\phi^1$							
	6	ϕ^3	$F_3F_2F_1\phi^0$	$gF_3F_2F_1\phi^0$	$F_3gF_2F_1\phi^0$	$F_3F_2gF_1\phi^0$	$g^2F_3F_2F_1\phi^0$	$gF_3gF_2F_1\phi^0$	$gF_3F_2gF_1\phi^0$	$F_3g^2F_2F_1\phi^0$	$F_3gF_2gF_1\phi^0$	$F_3F_2g^2F_1\phi^0$			
4	7	④ ϕ^4	④ $F_4\phi^3$	⑤ $gF_4\phi^3$		⑥ $g^2F_4\phi^3$									
	8	ϕ^4	$F_4\theta^2F_1\phi^0$	$gF_4\theta^2F_1\phi^0$	$F_4\theta^2gF_1\phi^0$	$g^2F_4\theta^2F_1\phi^0$	$gF_4\theta^2gF_1\phi^0$	$F_4\theta^2g^2F_1\phi^0$							
	9	ϕ^4	$F_4\theta^1F_2\phi^1$	$gF_4\theta^1F_2\phi^1$	$F_4\theta^1gF_2\phi^1$	$g^2F_4\theta^1F_2\phi^1$	$gF_4\theta^1gF_2\phi^1$	$F_4\theta^1g^2F_2\phi^1$							
	10	ϕ^4	$F_4\theta^1F_2F_1\phi^0$	$gF_4\theta^1F_2F_1\phi^0$	$F_4\theta^1gF_2F_1\phi^0$	$F_4\theta^1F_2gF_1\phi^0$	$g^2F_4\theta^1F_2F_1\phi^0$	$gF_4\theta^1gF_2F_1\phi^0$	$gF_4\theta^1F_2gF_1\phi^0$	$F_4\theta^1g^2F_2F_1\phi^0$	$F_4\theta^1gF_2gF_1\phi^0$	$F_4\theta^1F_2g^2F_1\phi^0$			
	11	ϕ^4	$F_4F_3\phi^2$	$gF_4F_3\phi^2$	$F_4gF_3\phi^2$		$gF_4F_3\phi^2$	$gF_4gF_3\phi^2$	$F_4g^2F_3\phi^2$						
	12	ϕ^4	$F_4F_3\theta^1F_1\phi^0$	$gF_4F_3\theta^1F_1\phi^0$	$F_4gF_3\theta^1F_1\phi^0$	$F_4F_3\theta^1gF_1\phi^0$	$g^2F_4F_3\theta^1F_1\phi^0$	$gF_4gF_3\theta^1F_1\phi^0$	$gF_4F_3\theta^1gF_1\phi^0$	$F_4g^2F_3\theta^1F_1\phi^0$	$F_4gF_3\theta^1gF_1\phi^0$	$F_4F_3\theta^1g^2F_1\phi^0$			
	13	ϕ^4	$F_4F_3F_2\phi^1$	$gF_4F_3F_2\phi^1$	$F_4gF_3F_2\phi^1$	$F_4F_3gF_2\phi^1$	$g^2F_4F_3F_2\phi^1$	$gF_4gF_3F_2\phi^1$	$gF_4F_3gF_2\phi^1$	$F_4g^2F_3F_2\phi^1$	$F_4gF_3gF_2\phi^1$	$F_4F_3g^2F_2\phi^1$			
	14	ϕ^4	$F_4F_3F_2F_1\phi^0$	$gF_4F_3F_2F_1\phi^0$	$F_4gF_3F_2F_1\phi^0$	$F_4F_3gF_2F_1\phi^0$	$F_4F_3F_2gF_1\phi^0$	$g^2F_4F_3F_2F_1\phi^0$	$gF_4gF_3F_2F_1\phi^0$	$gF_4F_3gF_2F_1\phi^0$	$gF_4F_3F_2gF_1\phi^0$	$F_4g^2F_3F_2F_1\phi^0$	$F_4gF_3gF_2F_1\phi^0$	$F_4F_3g^2F_2F_1\phi^0$	$F_4F_3F_2g^2F_1\phi^0$
	15	ϕ^4	$F_4(F_1\phi^0)^2$	$gF_4(F_1\phi^0)^2$	$F_4(F_1\phi^0)(gF_1\phi^0)$		$g^2F_4(F_1\phi^0)^2$	$gF_4(F_1\phi^0)(gF_1\phi^0)$	$F_4(F_1\phi^0)(g^2F_1\phi^0)$	$F_4(gF_1\phi^0)^2$					

ψ	
14	⑥ $F_4F_3g^2F_2F_1\phi^0$ $F_4F_3gF_2gF_1\phi^0$ $F_4F_3F_2g^2F_1\phi^0$

TABLE V

DIFFERENTIALS A AND E

ℓ	ORDER-ℓ	DEGREE, S	$k_1,\dots,k_s$	POSITION	$k\in[0,\infty]$ DEFINITION	ψ \ k	$[0,\infty]$	$[1,\infty]$	$[2,\infty]$	$[3,\infty]$	$[4,\infty]$	$[5,\infty]$	$[90]$	$[0,1]$	$[0,2]$	$[0,3]$	$[0,4]$	$[1,1]$	$[1,2]$	$[1,3]$
0	1	0		0	$\{0\}$		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	2	0		0	$\{1\}$		1	1	1	1	1	1	1	1	1	1	1	1	1	1
		1	0	0	$\{0,\psi_0\}$		2						2	2	2	2	2			
0	3	0		0	$\{2\}$		3	2	2	2	2	2	3	3	3	3	3	2	2	2
		1	0	0	$\{1,\psi_0\}$		4						4	4	4	4	4			
			0	1	$\{0,\psi_1\}$		5						5	5	5	5	5			
			0	2	$\{0,\psi_2\}$		6						6	6	6	6	6			
			1	0	$\{0,\psi_0\}$		7	3						7	7	7	7	3	3	3
0	4	0		0	$\{3\}$		8	4	3	3	3	3	7	8	8	8	8	4	4	4
		1	0	0	$\{2,\psi_0\}$		9						8	9	9	9	9			
			0	1	$\{1,\psi_1\}$		10						9	10	10	10	10			
			0	2	$\{1,\psi_2\}$		11						10	11	11	11	11			
			0	3	$\{0,\psi_3\}$		12						11	12	12	12	12			
			0	4	$\{0,\psi_4\}$		13						12	13	13	13	13			
			0	5	$\{0,\psi_5\}$		14						13	14	14	14	14			
			0	6	$\{0,\psi_6\}$		15						14	15	15	15	15			
			1	0	$\{1,\psi_0\}$		16	5						16	16	16	16	5	5	5
			1	1	$\{0,\psi_1\}$		17	6						17	17	17	17	6	6	6
			2	0	$\{0,\psi_0\}$		18	7	4						18	18	18		7	7
			0,1	0	$\{0,\psi_7\}$		19							18	19	19	19			
			1,0	0	$\{0,\psi_2\}$		20							19	20	20	20			
1	4	2	0,0	0	$\{0,\psi_0^2\}$		21						15	20	21	21	21			
0	5	0		0	$\{4\}$		22	8	5	4	4	4	16	21	22	22	22	7	8	8
		1	0	0	$\{3,\psi_0\}$		23						17	22	23	23	23			
			0	1	$\{2,\psi_1\}$		24						18	23	24	24	24			
			0	2	$\{2,\psi_2\}$		25						19	24	25	25	25			
			0	3	$\{1,\psi_3\}$		26						20	25	26	26	26			
			0	4	$\{1,\psi_4\}$		27						21	26	27	27	27			
			0	5	$\{1,\psi_5\}$		28						22	27	28	28	28			

TABLE V CONT.

ℓ	ORDER-ℓ	DEGREE, S	$k_1,\ldots,k_S$	POSITION	$k \in [0,\infty]$ DEFINITION	k / ψ	$[0,\infty]$	$[1,\infty]$	$[2,\infty]$	$[3,\infty]$	$[4,\infty]$	$[5,\infty]$	$[0,0]$	$[0,1]$	$[0,2]$	$[0,3]$	$[0,4]$	$[1,1]$	$[1,2]$	$[1,3]$
8	5	1	0	6	$\{1, \psi_6\}$		29						23	28	29	29	29			
			0	7	$\{0, \psi_8\}$		30						24	29	30	30	30			
			0	8	$\{0, \psi_9\}$		31						25	30	31	31	31			
			0	9	$\{0, \psi_{10}\}$		32						26	31	32	32	32			
			0	10	$\{0, \psi_{11}\}$		33						27	32	33	33	33			
			0	11	$\{0, \psi_{12}\}$		34						28	33	34	34	34			
			0	12	$\{0, \psi_{13}\}$		35						29	34	35	35	35			
			0	13	$\{0, \psi_{14}\}$		36						30	35	36	36	36			
			0	14	$\{0, \psi_{15}\}$		37						31	36	37	37	37			
7	5	1	0	0	$\{0, \psi_{21}\}$		38						32	37	38	38	38			
8	5	1	1	0	$\{2, \psi_0\}$		39	9						38	39	39	39	8	9	9
			1	1	$\{1, \psi_1\}$		40	10						39	40	40	40	9	10	10
			1	2	$\{0, \psi_3\}$		41	11						40	41	41	41	10	11	11
			1	3	$\{0, \psi_7\}$		42	12						41	42	42	42	11	12	12
			2	0	$\{1, \psi_0\}$		43	13	6						43	43	43		13	13
			2	1	$\{0, \psi_1\}$		44	14	7						44	44	44		14	14
			3	0	$\{0, \psi_0\}$		45	15	8	5						45	45			15
			0,1	0	$\{1, \psi_7\}$		46							42	45	46	46			
			0,1	1	$\{0, \psi_{16}\}$		47							43	46	47	47			
			0,1	2	$\{0, \psi_{17}\}$		48							44	47	48	48			
			0,1	3	$\{0, \psi_{19}\}$		49							45	48	49	49			
			0,1	4	$\{0, \psi_{20}\}$		50							46	49	50	50			
			1,0	0	$\{1, \psi_2\}$		51							47	50	51	51			
			1,0	1	$\{0, \psi_4\}$		52							48	51	52	52			
			1,0	2	$\{0, \psi_5\}$		53							49	52	53	53			
			1,0	3	$\{0, \psi_6\}$		54							50	53	54	54			
			0,2	0	$\{0, \psi_{18}\}$		55								54	55	55			
			2,0	0	$\{0, \psi_2\}$		56								55	56	56			
2	5	2	0,0	0	$\{0, \psi_0^2\}$		57						33	51	56	57	57	-	-	-

TABLE V CONT.

ℓ	ORDER-ℓ	DEGREE, S	$k_1,\dots,k_s$	POSITION	$k\in[0,\infty]$ DEFINITION	ψ \ k	0,∞	[1,∞]	[2,∞]	[3,∞]	[4,∞]	[5,∞]	[0,0]	[0,1]	[0,2]	[0,3]	[0,4]	[1,1]	[1,2]	[1,3]
1	5	2	0,0	0	$\{1,\psi_0^2\}$		58						34	52	57	58	58			
			0,0	1	$\{0,\psi_0\psi_1\}$		59						35	53	58	59	59			
			0,0	2	$\{0,\psi_0\psi_2\}$		60						36	54	59	60	60			
			1,0	0	$\{0,\psi_0^2\}$		61							55	60	61	61			
					$k\in[1,\infty]$ DEFINITION															
∞	6	0		0	$\{5\}$		62	16	9	6	5	5	37	56	61	62	62	12	15	16
		1	1	0	$\{3,\psi_0\}$			17										13	16	17
			1	1	$\{2,\psi_1\}$			18										14	17	18
			1	2	$\{1,\psi_2\}$			19										15	18	19
			1	3	$\{1,\psi_3\}$			20										16	19	20
			1	4	$\{0,\psi_4\}$			21										17	20	21
			1	5	$\{0,\psi_5\}$			22										18	21	22
			1	6	$\{0,\psi_6\}$			23										19	22	23
			2	0	$\{2,\psi_0\}$			24	10										23	24
			2	1	$\{1,\psi_1\}$			25	11										24	25
			2	2	$\{0,\psi_2\}$			26	12										25	26
			3	0	$\{1,\psi_0\}$			27	13	7										27
			3	1	$\{0,\psi_1\}$			28	14	8										28
			4	0	$\{0,\psi_0\}$			29	15	9	6									
			1,2	0	$\{0,\psi_7\}$			30											26	29
			2,1	0	$\{0,\psi_3\}$			31											27	30
1	6	2	1,1	0	$\{0,\psi_0^2\}$			32										20	28	31
∞	7	0		0	$\{6\}$			33	16	10	7	6						21	29	32
		1	1	0	$\{4,\psi_0\}$			34										22	30	33
			1	1	$\{3,\psi_1\}$			35										23	31	34
			1	2	$\{2,\psi_2\}$			36										24	32	35
			1	3	$\{2,\psi_3\}$			37										25	33	36
			1	4	$\{1,\psi_4\}$			38										26	34	37

TABLE V CONT.

ℓ	ORDER-ℓ	DEGREE, s	$k_1,\ldots,k_s$	POSITION	$k \in [1,\infty]$ DEFINITION (k / ψ)	$[0,\infty]$	$[1,\infty]$	$[2,\infty]$	$[3,\infty]$	$[4,\infty]$	$[5,\infty]$	$[90]$	$[0,1]$	$[0,2]$	$[0,3]$	$[0,4]$	$[1,1]$	$[1,2]$	$[1,3]$
8	7	1	1	5	$\{1,\psi_5\}$		39										27	35	38
			1	6	$\{1,\psi_6\}$		40										28	36	39
			1	7	$\{0,\psi_8\}$		41										29	37	40
			1	8	$\{0,\psi_9\}$		42										30	38	41
			1	9	$\{0,\psi_{10}\}$		43										31	39	42
			1	10	$\{0,\psi_{11}\}$		44										32	40	43
			1	11	$\{0,\psi_{12}\}$		45										33	41	44
			2	0	$\{3,\psi_0\}$		46	17										42	45
			2	1	$\{2,\psi_1\}$		47	18										43	46
			2	2	$\{1,\psi_2\}$		48	19										44	47
			2	3	$\{0,\psi_4\}$		49	20										45	48
			2	4	$\{0,\psi_7\}$		50	21										46	49
			3	0	$\{2,\psi_0\}$		51	22	11										50
			3	1	$\{1,\psi_1\}$		52	23	12										51
			3	2	$\{0,\psi_2\}$		53	24	13										52
			4	0	$\{1,\psi_0\}$		54	25	14	8									
			4	1	$\{0,\psi_1\}$		55	26	15	9									
			5	0	$\{0,\psi_0\}$		56	27	16	10	7								
			1,2	0	$\{1,\psi_7\}$		57											47	53
			1,2	1	$\{0,\psi_{13}\}$		58											48	54
			1,2	2	$\{0,\psi_{14}\}$		59											49	55
			2,1	0	$\{1,\psi_3\}$		60											50	56
			2,1	1	$\{0,\psi_5\}$		61											51	57
			2,1	2	$\{0,\psi_6\}$		62											52	58
			1,3	0	$\{0,\psi_{15}\}$		63												59
			3,1	0	$\{0,\psi_3\}$		64												60
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
2	7	2	1,1	0	$\{0,\psi_0^2\}$		65										34	53	61
1	7	2	1,1	0	$\{1,\psi_0^2\}$		66										35	54	62
			1,1	1	$\{0,\psi_0\psi_1\}$		67										36	55	63
			1,2	0	$\{0,\psi_0^2\}$		68											56	64
∞	8	0		0	$\{7\}$		69	28	17	11	8						37	57	65

TABLE VI

ORDER-ℓ	$k \in [0,\infty)$ ψ	DERIVATIVE HARMONICS β	SUBSTITUTION HARMONICS β	MULTIPLICATION HARMONICS β
1	0	1	$\theta^{\ell}/\ell!$	0
2	1	1	$\theta^{\ell+1}/(\ell+1)!$	0
	2	1	$\alpha_{0,0}$	$\alpha_{0,0}$
3	3	1	$\theta^{\ell+2}/(\ell+2)!$	0
	4	2^1	$\alpha_{0,0}\,\theta$	$\alpha_{0,0}\,\theta$
	5	1	$\alpha_{0,1}$	$\alpha_{0,1}$
	6	1	$\alpha_{0,2}$	$\alpha_{0,2}$
	7	1	$\alpha_{1,0}$	$\alpha_{1,0}$
4	8	1	$\theta^{\ell+3}/(\ell+3)!$	0
	9	$3^2/2!$	$\alpha_{0,0}\,\theta^2/2!$	$\alpha_{0,0}\,\theta^2/2!$
	10	3^1	$\alpha_{0,1}\,\theta$	$\alpha_{0,1}\,\theta$
	11	3^1	$\alpha_{0,2}\,\theta$	$\alpha_{0,2}\,\theta$
	12	1	$\alpha_{0,3}$	$\alpha_{0,3}$
	13	2^1	$\alpha_{0,4}$	$\alpha_{0,4}$
	14	1	$\alpha_{0,5}$	$\alpha_{0,5}$
	15	1	$\alpha_{0,6}$	$\alpha_{0,6}$
	16	3^1	$\alpha_{1,0}\,\theta$	$\alpha_{1,0}\,\theta$
	17	1	$\alpha_{1,1}$	$\alpha_{1,1}$
	18	1	$\alpha_{2,0}$	$\alpha_{2,0}$
	19	1	$\alpha_{0,7}$	$\alpha_{0,7}$
	20	1	$\alpha_{1,2}$	$\alpha_{1,2}$
	21	3	$\alpha^2_{0,0}/2!$	$\alpha_{0,0}\,\gamma_{0,0}$

TABLE VII

TRANSLATION HARMONICS $\gamma_j^i(t)$ $\quad A^i(t+\tau,0)=\sum_j \gamma_j^i(t)A^j(\tau,0)$

Ψ	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0	⓪ 1																					
1	① 1	⓪ 1																				
2	1		1																			
3	② 1	① 1		⓪ 1																		
4	2′	1′	1		1																	
5	1		1			1																
6	1		1				1															
7	1	1						1														
8	③ 1	② 1		① 1					⓪ 1													
9	$3^2/2$	$(3^2-2)/2$	1	$2^2/2$	1					1												
10	3′	1′	2		1	1					1											
11	3′	1′	2		1		1					1										
12	1		1			1							1									
13	2′		2′			1′	1							1								
14	1		1				1								1							
15	1		1				1									1						
16	3′	3′		2′				1									1					
17	1	1						1										1				
18	1	1																	1			
19	1		1			1														1		
20	1	1						1													1	
21	3	2	1		1																	1

SUPPLEMENT TO TABLE VIII

NAME	OPERATOR DEFINITION ($k, k_0 \in \bar{P} \subset P = \{0, \cdots, p-1\}$; R = RANK)		
Ψ^k	$C^{0}_{j,k}$	$C^{s}_{j,k}$ $\quad s \geq 1$	E_k
$W_i^{(k)}$	$\sum_{j_1} \frac{(R+k)!}{(R-1)!} f^{(k)}_{ij_1} D(X \circ u_{j_1}^{j+\ell-1})$	$\sum_{j_1} \frac{(R+k)!}{(R-1)!} f^{(k)}_{ij_1} C^{(j-1, k_1 \cdots k_s)}_{j_1 N_0 \cdots N_s}$	$\sum_{j_1} \frac{(R+k)!}{(R-1)!} g^{(k)}_{ik_0 j_1}$
$\Phi_i^{(k)}$	$\sum_{j_1} \frac{(R+k)!}{(R-1)!} f^{(k)}_{ij_1} \Theta_{j_1}^{j+\ell-1}$	$\sum_{j_1} \frac{(R+k)!}{(R-1)!} f^{(k)}_{ij_1} \Theta_{j_1}^{j-1}$	$\sum_{j_1} \frac{(R+k)!}{(R-1)!} g^{(k)}_{ik_0 j_1}$
A	$D(X \circ u)^{j+\ell-1}$	$C^{(j-1, k_1, \cdots, k_s)}_{N_0 N_1 \cdots N_s}$	0
$\Gamma_i^{(k)}$	$\sum_{j_1} f^{(k)}_{ij_1} \Theta_{j_1}^{j+\ell-1}$	$\sum_{j_1} f^{(k)}_{ij_1} \Theta_{j_1}^{j-1}$	$\sum_{j_1} g^{(k)}_{ik_0 j_1}$
$\gamma^{(k)}$	$\frac{(R+k)!}{(R-1)!}$	$\frac{(R+k)!}{(R-1)!}$	$\frac{(R+k)!}{(R-1)!}$
$\alpha^{(k)}$	1	$\frac{(R-1)!}{(j-1)!} \prod_{i=1}^{\delta} \frac{1}{(\omega_i)! (R_i+k_i)!^{\omega_i}}$	1
$\pi^{(k)}$	$\frac{1}{(R-1)!}$	$\frac{1}{(j-1)!} \prod_{i=1}^{\delta} \frac{1}{(\omega_i)!}$	1

TABLE VIII

GENERIC FUNCTIONS

$k \in P = \{0, 1, \cdots, p-1\}$

$k_1 \in \bar{P} \subset P$

ℓ	ORDER-ℓ	DEGREES	$k_1, \cdots, k_s$	POSITION	$k \in [0,\infty]$ ψ_i^k	$k \in [1,\infty]$ ψ_i^k	RANK-ℓ	$(2+k_1)$	$(3+k_1)$	$(4+k_1)$	$(5+k_1)$	$(6+k_1+k_2)$
0	1	0		0	0	0	① $C^0_{1,k}$	$E_k C^0_{1,k_1}$				
0	2	0		0	1	1	② $C^0_{2,k}$	$E_k C^0_{2,k_1}$				
		1	0	0	2		$C_{1,k}\psi^0_0$	$E_k C^1_{1,k_1} C^0_{1,0}$	$C^1_{1,k} E_0 C^0_{1,k_1}$			
0	3	0		0	3	2	③ $C^0_{3,k}$	$E_k C^0_{3,k_1}$				
		1	0	0	4		$C^1_{2,k}\psi^0_0$	$E_k C^1_{2,k_1} C^0_{1,0}$	$C^1_{2,k} E_0 C^0_{1,k_1}$			
			0	1	5		$C^1_{1,k}\psi^0_1$	$E_k C^1_{1,k_1} C^0_{2,0}$	$C^1_{1,k_1} E_0 C^0_{2,k_1}$			
			0	2	6		$C^1_{1,k}\psi^0_2$	$E_k C^1_{1,k_1} C^1_{1,0} C^0_{1,0}$	$C^1_{1,k} E_0 C^1_{1,k_1} C^0_{1,0}$	$C^1_{1,k} C^1_{1,0} E_0 C^0_{1,k_1}$		
		1	0	0	7	3	$C^1_{1,k}\psi^1_0$	$E_k C^1_{1,k_1} C^0_{1,1}$	$C^1_{1,k} E_1 C^0_{1,k_1}$			
0	4	0		0	8	4	④ $C^0_{4,k}$	$E_k C^0_{4,k_1}$				
		1	0	0	9		$C^1_{3,k}\psi^0_0$	$E_k C^1_{3,k_1} C^0_{1,0}$	$C^1_{3,k} E_0 C^0_{1,k_1}$			
			0	1	10		$C^1_{2,k}\psi^0_1$	$E_k C^1_{2,k_1} C^0_{2,0}$	$C^1_{2,k} E_0 C^0_{2,k_1}$			
			0	2	11		$C^1_{2,k}\psi^0_2$	$E_k C^1_{2,k_1} C^1_{1,0} C^0_{1,0}$	$C^1_{2,k} E_0 C^1_{1,k_1} C^0_{1,0}$	$C^1_{2,k} C^1_{1,0} E_0 C^0_{1,k_1}$		
			0	3	12		$C^1_{1,k}\psi^0_3$	$E_k C^1_{1,k_1} C^0_{3,0}$	$C^1_{1,k} E_0 C^0_{3,k_1}$			
			0	4	13		$C^1_{1,k}\psi^0_4$	$E_k C^1_{1,k_1} C^1_{2,0} C^0_{1,0}$	$C^1_{1,k} E_0 C^1_{2,k_1} C^0_{1,0}$	$C^1_{1,k} C^1_{2,0} E_0 C^0_{1,k_1}$		
			0	5	14		$C^1_{1,k}\psi^0_5$	$E_k C^1_{1,k_1} C^1_{1,0} C^0_{2,0}$	$C^1_{1,k} E_0 C^1_{1,k_1} C^0_{2,0}$	$C^1_{1,k} C^1_{1,0} E_0 C^0_{2,k_1}$		
			0	6	15		$C^1_{1,k}\psi^0_6$	$E_k C^1_{1,k_1} (C^1_{1,0})^2 C^0_{1,0}$	$C^1_{1,k} E_0 C^1_{1,k_1} C^1_{1,0} C^0_{1,0}$	$C^1_{1,k} C^1_{1,0} E_0 C^1_{1,k_1} C^0_{1,0}$	$C^1_{1,k} (C^1_{1,0})^2 E_0 C^0_{1,k_1}$	
			1	0	16	5	$C^1_{2,k}\psi^1_0$	$E_k C^1_{2,k_1} C^0_{1,1}$	$C^1_{2,k} E_1 C^0_{1,k_1}$			
			1	1	17	6	$C^1_{1,k}\psi^1_1$	$E_k C^1_{1,k_1} C^0_{2,1}$	$C^1_{1,k} E_1 C^0_{2,k_1}$			
			2	0	18	7	$C^1_{1,k}\psi^2_0$	$E_k C^1_{1,k_1} C^0_{1,2}$	$C^1_{1,k} E_2 C^0_{1,k_1}$			
			0,1	0	19		$C^1_{1,k}\psi^0_7$	$E_k C^1_{1,k_1} C^1_{1,0} C^0_{1,1}$	$C^1_{1,k} E_0 C^1_{1,k_1} C^0_{1,1}$	$C^1_{1,k} C^1_{1,0} E_1 C^0_{1,k_1}$		
			1,0	0	20		$C^1_{1,k}\psi^1_2$	$E_k C^1_{1,k_1} C^1_{1,0} C^0_{1,0}$	$C^1_{1,k} E_1 C^1_{1,k_1} C^0_{1,0}$	$C^1_{1,k} C^1_{1,1} E_0 C^0_{1,k_1}$		
1	4	2	0,0	0	21		$C^2_{1,k}(\psi^0_0)^2$	$E_k C^2_{1,k_1} (C^0_{1,0})^2$	$C^2_{1,k} C^0_{1,0} (E_0 C^0_{1,k_1})$			$C^2_{1,k} (E_0 C^0_{1,k_1})(E_0 C^0_{1,k_2})$

TABLE IX

DERIVATIVE HARMONICS $\alpha^{(k)} \times (R+k)!$

$k_1 \in \bar{P} \subset P = \{0, \cdots, p-1\}$

NOTE $a^i \equiv (a)^i \equiv i!\binom{l+a}{i}$, $i \geq 0$

ψ	RANR-ℓ	$2+k_1$			
0	① 1	1			
1	② 1	1			
2	1	1	1		
3	③ 1	$4+k_1$ 1			
4	2^1	2^1	$(3+k_1)^1$		
5	1	1	1		
6	1	1	1	1	
7	1	1	1		
8	④ 1	$5+k_1$ 1			
9	$3^2/2!$	$3^2/2!$	$(4+k_1)^2/2!$		
10	3^1	3^1	$(4+k_1)^1$		
11	3^1	3^1	$(4+k_1)^1$	$(4+k_1)^1$	
12	1	1	1		
13	2^1	2^1	2^1	$(3+k_1)^1$	
14	1	1	1	1	
15	1	1	1	1	1
16	3^1	3^1	$(4+k_1)^1$		
17	1	1	1		
18	1	1	1		
19	1	1	1	1	
20	1	1	1	1	$k_1 \neq k_2$ / $k_1 = k_2$
21	3	3	$(4+k_1)^2/2!$	$6+k_1+k_2$	$\frac{(5+k_1+k_2)!}{(2+k_1)!(2+k_2)!}$ / $\frac{(5+2k_1)!}{2[(2+k_1)!]^2}$

TABLE X

POLYNOMIAL WEIGHTS $\gamma_i^{(k)}$

$k_1, k_1 \in \bar{P} \subset P = \{0, \cdots, p-1\}$

NOTE $a^{\underline{i}} \equiv (a)^{\underline{i}} \equiv i!\binom{k+a}{i}$

ψ	RANK-ℓ			
0	① $(1+k)^{k+1}$	(2+k_1) $(2+k_1+k)^{k_1+k+2}$		
1	② $(2+k)^{k+1}$	(3+k_1) $(3+k_1+k)^{k_1+k+2}$		
2	" $\cdot 1^1$	" $\cdot 1^1$	$(3+k_1+k)^{k_1+k+3}$	
3	③ $(3+k)^{k+1}$	(4+k_1) $(4+k_1+k)^{k_1+k+2}$		
4	" $\cdot 1^1$	" $\cdot 1^1$	$(4+k_1+k)^{k+1}\cdot(2+k_1)^{k_1+2}$	
5	" $\cdot 2^1$	" $\cdot 2^1$	" $\cdot(3+k_1)^{k_1+2}$	
6	" $\cdot 2^2$	" $\cdot 2^2$	" $\cdot$ " $\cdot 1^1$	$(4+k_1+k)^{k+1}\cdot(3+k_1)^{k_1+3}$
7	" $\cdot 2^2$	" $\cdot 2^2$	" $\cdot(3+k_1)^{k_1+3}$	
8	④ $(4+k)^{k+1}$	(5+k_1) $(5+k_1+k)^{k_1+k+2}$		
9	" $\cdot 1^1$	" $\cdot 1^1$	$(5+k_1+k)^{k+1}\cdot(2+k_1)^{k_1+2}$	
10	" $\cdot 2^1$	" $\cdot 2^1$	" $\cdot(3+k_1)^{k_1+2}$	
11	" $\cdot 2^2$	" $\cdot 2^2$	" $\cdot$ " $\cdot 1^1$	$(5+k_1+k)^{k+1}\cdot(3+k_1)^{k_1+3}$
12	" $\cdot 3^1$	" $\cdot 3^1$	" $\cdot(4+k_1)^{k_1+2}$	
13	" $\cdot 3^1 1^1$	" $\cdot 3^1 1^1$	" $\cdot$ " $\cdot 1^1$	$(5+k_1+k)^{k+1}\cdot(4+k_1)^{1}\cdot(2+k_1)^{k_1+2}$
14	" $\cdot 3^2$	" $\cdot 3^2$	" $\cdot$ " $\cdot 2^1$	" $\cdot(4+k_1)^{k_1+3}$
15	" $\cdot 3^3$	" $\cdot 3^3$	" $\cdot$ " $\cdot 2^2$	" $\cdot$ " $\cdot 1^1$ $(5+k_1+k)^{k+1}\cdot(4+k_1)^{k_1+4}$
16	" $\cdot 2^2$	" $\cdot 2^2$	" $\cdot(3+k_1)^{k_1+3}$	
17	" $\cdot 3^2$	" $\cdot 3^2$	" $\cdot(4+k_1)^{k_1+3}$	
18	" $\cdot 3^3$	" $\cdot 3^3$	" $\cdot(4+k_1)^{k_1+4}$	
19	" $\cdot 3^3$	" $\cdot 3^3$	" $\cdot(4+k_1)^{k_1+2}\cdot 2^2$	$(5+k_1+k)^{k+1}\cdot(4+k_1)^{k_1+4}$
20	" $\cdot 3^3$	" $\cdot 3^3$	" $\cdot(4+k_1)^{k_1+3}\cdot 1^1$	" $\cdot$ "
21	" $\cdot 1^1\cdot 1^1$	" $\cdot 1^1\cdot 1^1$	" $\cdot(2+k_1)^{k_1+2}\cdot 1^1$	(6+k_1+k_2) $(6+k_1+k_2+k)^{k+1}\cdot(2+k_1)^{k_1+2}\cdot(2+k_2)^{k_2+2}$

TABLE XI

ELEMENTARY POLYNOMIALS $\Gamma^{(k)}$

$k \in [0,\infty]$ Ψ	$k, k_1 \in \bar{P} \subset P = \{0, \dots, p-1\}$ RANK-ℓ		FOR EXACT POINTS $\begin{cases} g^k \to g^k_{i k_0 j} \equiv 0 \begin{cases} 0, & i \neq j \\ \frac{(\ell+\alpha-1)!}{(\ell+\alpha+k)!}\Theta_i^{k+1}, & i=j \end{cases} \\ E^k_\alpha \to f^k_{ij} \to \end{cases}$			
0	① ϕ^{k+1}	$F_1^k\phi^0$	(2+k₁) $g^k F_1^{k_1}\phi^0$			
1	② ϕ^{k+2}	$F_2^k\phi^1$	(3+k₁) $g^k F_2^{k_1}\phi^0$			
2	"	$F_2^k F_1^0\phi^0$	$g^k F_2^{k_1} F_1^0\phi^0$	$F_2^k g^0 F_1^{k_1}\phi^0$		
3	③ ϕ^{k+3}	$F_3^k\phi^2$	(4+k₁) $g^k F_3^{k_1}\phi^2$			
4	"	$F_3^k \Theta F_1^0\phi^0$	$g^k F_3^{k_1}\Theta F_1^0\phi^0$	$F_3^k \Theta g^0 F_1^{k_1}\phi^0$		
5	"	$F_3^k F_2^0\phi^1$	$g^k F_3^{k_1} F_2^0\phi^0$	$F_3^k g^0 F_2^{k_1}\phi^1$		
6	"	$F_3^k F_2^0 F_1^0\phi^0$	$g^k F_3^{k_1} F_2^0 F_1^0\phi^0$	$F_3^k g^0 F_2^{k_1} F_1^0\phi^0$	$F_3^k F_2^0 g^0 F_1^{k_1}\phi^0$	
7	"	$F_3^k F_1^1\phi^0$	$g^k F_3^{k_1} F_1^1\phi^0$	$F_3^k g^1 F_1^{k_1}\phi^0$		
8	④ ϕ^{k+4}	$F_4^k\phi^3$	(5+k₁) $g^k F_4^{k_1}\phi^3$			
9	"	$F_4^k \Theta^2 F_1^0\phi^0$	$g^k F_4^{k_1}\Theta^2 F_1^0\phi^0$	$F_4^k \Theta^2 g^0 F_1^{k_1}\phi^0$		
10	"	$F_4^k \Theta F_2^0\phi^1$	$g^k F_4^{k_1}\Theta F_2^0\phi^1$	$F_4^k \Theta g^0 F_2^{k_1}\phi^1$		
11	"	$F_4^k \Theta F_2^0 F_1^0\phi^0$	$g^k F_4^{k_1}\Theta F_2^0 F_1^0\phi^0$	$F_4^k \Theta g^0 F_2^{k_1} F_1^0\phi^0$	$F_4^k \Theta F_2^0 g^0 F_1^{k_1}\phi^0$	
12	"	$F_4^k F_3^0\phi^2$	$g^k F_4^{k_1} F_3^0\phi^2$	$F_4^k g^0 F_3^{k_1}\phi^2$		
13	"	$F_4^k F_3^0 \Theta F_1^0\phi^0$	$g^k F_4^{k_1} F_3^0 \Theta F_1^0\phi^0$	$F_4^k g^0 F_3^{k_1}\Theta F_1^0\phi^0$	$F_4^k F_3^0 \Theta g^0 F_1^{k_1}\phi^0$	
14	"	$F_4^k F_3^0 F_2^0\phi^1$	$g^k F_4^{k_1} F_3^0 F_2^0\phi^1$	$F_4^k g^0 F_3^{k_1} F_2^0\phi^1$	$F_4^k F_3^0 g^0 F_2^{k_1}\phi^1$	
15	"	$F_4^k F_3^0 F_2^0 F_1^0\phi^0$	$g^k F_4^{k_1} F_3^0 F_2^0 F_1^0\phi^0$	$F_4^k g^0 F_3^{k_1} F_2^0 F_1^0\phi^0$	$F_4^k F_3^0 g^0 F_2^{k_1} F_1^0\phi^0$	$F_4^k F_3^0 F_2^0 g^0 F_1^{k_1}\phi^0$
16	"	$F_4^k \Theta F_1^1\phi^0$	$g^k F_4^{k_1}\Theta F_1^1\phi^0$	$F_4^k \Theta g^1 F_1^{k_1}\phi^0$		
17	"	$F_4^k F_2^1\phi^1$	$g^k F_4^{k_1} F_2^1\phi^1$	$F_4^k g^1 F_2^{k_1}\phi^1$		
18	"	$F_4^k F_1^2\phi^0$	$g^k F_4^{k_1} F_1^2\phi^0$	$F_4^k g^2 F_1^{k_1}\phi^0$		
19	"	$F_4^k F_3^0 F_1^1\phi^0$	$g^k F_4^{k_1} F_3^0 F_1^1\phi^0$	$F_4^k g^0 F_3^{k_1} F_1^1\phi^0$	$F_4^k F_3^0 g^1 F_1^{k_1}\phi^0$	
20	"	$F_4^k F_2^1 F_1^0\phi^0$	$g^k F_4^{k_1} F_2^1 F_1^0\phi^0$	$F_4^k g^1 F_2^{k_1} F_1^0\phi^0$	$F_4^k F_2^1 g^0 F_1^{k_1}\phi^0$	
21	"	$(F_4^k)_1 (F_1^0\phi^0)_1^2$	$g^k (F_4^{k_1})_1 (F_1^0\phi^0)_1^2$	$(F_4^{k_1})_1 (F_1^0\phi^0)_1 (g^0 F_1^{k_1}\phi^0)_1$	(6+k₁+k₂) $(F_4^{k_1})_1 (g^0 F_1^{k_2}\phi^0)_1 (g^0 F_1^{k_2}\phi^0)_1$	

TABLE XII

PRODUCT COEFFICIENTS $(\pi^{(k)})^{-1}$

$k, k_i \in \bar{P} \subset P = \{0, \cdots, p-1\}$

NOTE $(a) \equiv (\ell + a)$

Ψ	RANK-ℓ	$2+k_i$		$3+k_i$	$4+k_i$
0	① (0)!	(0)!			
1	② (1)!	(1)!			
2	(0)!	(0)!	(0)!		
3	③ (2)!	(2)!			
4	(0)!	(0)!	(0)!		
5	(1)!	(1)!	(1)!		
6	(0)!	(0)!	(0)!	(0)!	
7	(0)!	(0)!	(0)!		
8	④ (3)!	(3)!			
9	2!(0)!	2!(0)!	2!(0)!		
10	(1)!	(1)!	(1)!		
11	(0)!	(0)!	(0)!	(0)!	
12	(2)!	(2)!	(2)!		
13	(0)!	(0)!	(0)!	(0)!	
14	(1)!	(1)!	(1)!	(1)!	
15	(0)!	(0)!	(0)!	(0)!	(0)!
16	(0)!	(0)!	(0)!		
17	(1)!	(1)!	(1)!		
18	(0)!	(0)!	(0)!		
19	(0)!	(0)!	(0)!	(0)!	
20	(0)!	(0)!	(0)!	(0)!	
21	2!	2!	1	$6+k_1+k_2$	1 $\{k_1 \neq k_2\}$ 2! $\{k_1 = k_2\}$

TABLE XIII

TRANSLATION HARMONICS $\gamma_j^i(t)$ $E^i(t+\rho,0)=\sum_j \gamma_j^i(t)E^j(\rho,0)$

ψ	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0	⑥ 1																					
1	① 1	⑤ 1																				
2			1																			
3	② 1	① 1		⑥ 1																		
4			1		1																	
5			1			1																
6							1															
7								1														
8	③ 1	② 1		① 1					⑥ 1													
9			1		1					1												
10			2		1	1					1											
11							1					1										
12			1			1							1									
13							1							1								
14							1								1							
15																1						
16								1									1					
17								1										1				
18																			1			
19																				1		
20																					1	
21																						1

TABLE XIV

APPROXIMATION/ERROR HARMONICS

$k\in$ [1,∞]	$k\in$ [0]	$k\in$ [g∞]	$k\in\bar{P}\subset P=\{0,\cdots,p-1\}$ ORDER-ℓ
0	0	0	① $g^k\psi_0 + F^k\phi^0$
1	1	1	② $g^k\psi_1 + F^k\phi^1$
		2	$g^k\psi_2 + F^k\psi_0^0$
2	3	3	③ $g^k\psi_3 + F^k\phi^2$
		4	$g^k\psi_4 + F^k\phi^1\psi_0^0$
		5	$g^k\psi_5 + F^k\psi_1^0$
		6	$g^k\psi_6 + F^k\psi_2^0$
3		7	$g^k\psi_7 + F^k\psi_0^1$
4	7	8	④ $g^k\psi_8 + F^k\phi^3$
	8	9	$g^k\psi_9 + F^k\phi^2\psi_0^0$
	9	10	$g^k\psi_{10} + F^k\phi^1\psi_1^0$
	10	11	$g^k\psi_{11} + F^k\phi^1\psi_2^0$
	11	12	$g^k\psi_{12} + F^k\psi_3^0$
	12	13	$g^k\psi_{13} + F^k\psi_4^0$
	13	14	$g^k\psi_{14} + F^k\psi_5^0$
	14	15	$g^k\psi_{15} + F^k\psi_6^0$
5		16	$g^k\psi_{16} + F^k\phi^1\psi_0^1$
6		17	$g^k\psi_{17} + F^k\psi_1^1$
7		18	$g^k\psi_{18} + F^k\psi_0^2$
		19	$g^k\psi_{19} + F^k\psi_7^0$
		20	$g^k\psi_{20} + F^k\psi_2^1$
	15	21	$g^k\psi_{21} + \frac{1}{2!}F^k(\psi_0^0)^2$

$$g^k\psi_m = \sum_{j,k_0} g^k_{ik_0j}\,\psi^{k_0}_{jm}$$

$$F^k\phi^\alpha = \sum_i f^k_{ij}\,\frac{\theta_j^{\ell+\alpha}}{(\ell+\alpha)!}$$

$$F^k\phi^\alpha\psi^k_{m_1}\cdots\psi^k_{m_2} = \sum_j f^k_{ij}\,\frac{\theta_j^\alpha}{\alpha!}\,\psi^{k_1}_{jm_1}\cdots\psi^{k_2}_{jm_2}$$

Appendix II

PROCEDURE RKMI SOURCE LISTINGS

In this appendix are presented two schematic source listings, a list of procedure descriptions for all the main procedures, a list of variables, and a complete source listing of RKMI. The first source listing gives an overall view of the structure of RKMI; the second source listing gives declarations of all the variables of the procedures. The procedure descriptions give short descriptions of the task that the procedure is to perform. The list of variables gives a description of most of the global variables. The last source listing is the program RKMI given in reference ALGOL.

It is hoped that these listings, along with those of Appendix III and along with the discussion of Chapter V, will serve as a guide for the reader that wishes to understand how the program works. These listings and discussions can, however, only serve as a guide and, as is unfortunately always the case with programs, the interested reader must address himself directly to the program and the problem it solves.

Schematic Source Listing 1

The following is a schematic source listing of RKM1 in which all the procedures and the global variables are declared. This listing provides an overall view of the structure of the program.

```
Program RKM1
begin   integer field,decimal,n for print,nil,line length,left margin,
                right margin,count,height,height1,pmax,last,last1,
                last data,l,lastl,temp0,type,der,order,upper,q,e,period,
                origin,mode,mode1,length,length1,length2,control,list
                length,n,no,no1,no2,a0,A,A1,i,i1,i2,j,j1,name,type set;
        Boolean temp,tempS,print scheme,linear comb,B00,B0,B01,B02,BA1,
                BA2,left adjust,inout;
        integer array im[0:1],pring length[0:12];
begin   comment input-output procedures;
        procedure dump(a);  ;
        procedure lines(n);  ;
        procedure spaces(n);  ;
        procedure page;  ;
        procedure s(string);  ;
        procedure sr(string);  ;
        procedure ps(i,string);  ;
        procedure check margin(a);  ;
        procedure pi(a);  ;
        procedure pfi(a,c);  ;
        integer procedure ioi(string);  ;
        integer procedure iob(string);  ;
        procedure title;  ;
```

Source Listing 1 Cont.

```
begin: comment  There appears here a section which inputs data that is
                used to set computer parameters;
begin  comment  Variable array declarations followed by procedure
                declarations;
       integer array num,no3[-1:2],cond[0:height-1];T[-1:0,0:exq],
                     E[-1:3,0:exq],V[0:list length],Z[-1:2,0:order-1,
                     0:exq,0:im[0]-1],Zp[0:order-1,0:im[0]-1],
                     v[0:1,0:exq,0:height1-1],vs[0:1,0:height1-1],
                     W[1:2,0:im[0]-1],D[0:im[0]-1,0:im[0]-1],
                     a[-1:0,0:im[0]-1,0:im[0]-1],B11,B2[0:pmax];
       procedure check;   ;
       procedure normalize(a,b);   ;
       integer procedure index(i);   ;
       integer procedure fact(n);   ;
       procedure debug(orgin,a1,a2,a3);   ;
       integer procedure father(n,BO,B,name of son,son);   ;
       integer procedure son(father entry,son1,B);   ;
       integer procedure get atom(father entry,atomic set,atom,B);   ;
       integer procedure atom(i);   ;
       integer procedure collection(n,B,set,f);   ;
       Boolean procedure Bncollection(B2,B1,n,BO2,k);   ;
       procedure sum(name,i,length of sum,name of list);   ;
       procedure print sum(name);   ;
       integer procedure minimum(v,length,w);   ;
       integer procedure JPn(cn,cd,n,B1);   ;
       integer procedure Ze;   ;
```

Source Listing 1 Cont.

```
        procedure create E(i,BB);   ;
        procedure translate(v,name1,name2,num,type);   ;
        procedure print list(i,length,name,name of list,sign);   ;
        procedure list(name of list,BB);   ;
        procedure conditions E(name of vector,name);   ;
        procedure data;   ;
        procdeure check list(first,last);   ;
        procedure scheme;   ;
        comment All declarations have been made. The following section
                constitutes the control section;
again1:
again:
        if control = -1 then       ;
        if control = 0  then begin      end else
        if control = 1  then begin      end else
        if control = 2  then begin      end else
        if control = 3  then begin      end else
        if control = 4  then begin      end of control;
fin:
end     of program;
end of computations:
end
end of RKM1
```

Schematic Source Listing 2

In the following listing there appear the declarations of all formal parameters, all variables and all procedures used by each procedure. In order to declare the global variables we have introduced the type declaration global. The procedures are listed in the order of their appearance in RKM1.

```
procedure dump(a); integer a;
begin integer array B[0:1]; end dump;

procedure lines(n); value n; integer n;
begin integer i;
      global procedure output;
      global integer count;
end lines;

procedure spaces(n); value n; integer n;
begin integer i;
      global procedure outcharacter;
      global integer count;
end spaces;

procedure page;
begin global procedure output; end page;

procedure s(string); string string;
begin integer i,length;
      global procedure chlength,outcharacter;
```

Source Listing 2 Cont.

```
        global integer count;
end s;

procedure sr(string); string string;
begin global integer line length;
      global Boolean inout;
      global procedure spaces
end sr;

procedure ps(i,string); integer i; string string;
begin global integer array print length;
      global procedure check margin,s;
end ps;

procedure check margin(a); value a; integer a;
begin integer i;
      global integer count,right margin,left margin;
      global procedure lines, spaces,s;
end check margin;

procedure pi(a); value a; integer a;
begin procedure layout;
      begin global integer field;
            global procedure format;
      end layout;
      procedure list(item); procedure item;   ;
      global integer count,field;
```

Source Listing 2 Cont.

```
        global procedure spaces,s,outlist;
end pi;

procedure pfi(b,c); value b,c; integer b,c;
begin integer a,n,n1;
      global integer right margin,left margin;
      global procedure s;
      begin procedure layout;
            begin global procedure format; end layout;
            procedure list(item); procedure item;   ;
            global integer count,typeset;
            global procedure check margin,outlist;
      end
end pfi;

integer procedure ioi(string); string string;
begin integer n;
      global Boolean inout;
      global procedure input;
      begin global integer line length;
            global Boolean left adjust;
            global procedure lines,spaces,s,output;
      end
end ioi;
```

Source Listing 2 Cont.

```
Boolean procedure iob(string); string string;
begin integer i,j;
      Boolean b;
      global integer line length;
      global Boolean inout, left adjust;
      global procedure eof,incharacter,lines,spaces,s;
end iob;

procedure title;
begin integer i,j,left margin1;
      global integer count,left margin;
      global procedure eof,incharacter,check margin,s,equiv,
                       input,output;
end title;

procedure check;
begin global integer last,temp0,field,list length;
      global Boolean temp;
      global procedure lines,s,pi,check list;
      global lable end of computations;
end check;

procedure normalize(a,b); integer a,b;    ;

procedure index(i); integer i;
begin global integer q,i1,i2,j,j1,period end index;
```

Source Listing 2 Cont.

```
procedure fact(n); value n; integer n;
begin integer i,j end fact;

procedure debug(origin,a1,a2,a3); integer origin,a1,a2,a3;
begin integer i,field1;
      global integer n for pring,field,no,no1,no2,last,nil;
      global integer array no3,V;
      global Boolean right adjust;
      global procedure lines,s,pi;
end debug;

integer procedure father(n,BO,name of son, son);
        integer n,name of son, son; Boolean B,BO;
begin integer copy,son1;
      global integer last,nil;
      global integer array V;
      global procedure debug;
end father;

integer procedure son(father entry,son1,B);
        integer father entry,son1; Boolean B;
begin global integer nil;
      global integer array V;
      global procedure debug;
end son;
```

Source Listing 2 Cont.

```
integer procedure get atom(father entry,atomic set, atom,B);
        integer father entry,atomic set, atom; Boolean B;
comment unpacked;
begin global integer nil;
      global integer array V;
      global procedure debug;
end get atom;

integer procedure get atom(father entry,atomic set,atom,B);
        integer father entry,atomic set,atom; Boolean B;
comment packed 2 atoms/word;
begin integer n1,n2;
      global integer nil;
      global integer array V;
      global Boolean BA1;
      global procedure abs,debug;
end get atom;

integer procedure atom(i); value i; integer i;
comment unpacked;
begin global integer last,nil;
      global integer array V;
      global procedure check,debug,lines,s,pfi,check list,abs;
      global lable end of computations;
end atom;
```

Source Listing 2 Cont.

```
integer procedure atom(i); value i; integer i;
comment packed 2 atoms/word;
begin global integer last,no,nil;
      global integer arrary V;
      global Boolean BA2;
      global procedure abs, lines,s,pfi,check list, check,debug,dump;
      begin integer copy,n1,n2 end
end atom;

integer procedure collection(n,B,set,f);
        integer n,set; Boolean B; integer procedure f;
begin integer copy,son1;
      Boolean BB;
      global integer last,nil,no,A;
      global Boolean B0,B00;
      global procedure father,son,debug;
end collection;

Boolean procedure Bncollection(B2,B1,n,B02,k);
        value n,k; integer n,k; integer array B2,B1; Boolean B02;
begin integer j;
      own integer arrary B11[0:20];
      own Boolean array B01[0:20];
      global integer nil;
      global procedure son;
end Bncollection;
```

Source Listing 2 Cont.

```
procedure sum(name,i,length of sum,name of list);
        value length of sum;
        integer name,i,length of sum,name of list;
begin integer l1;
      integer procedure add end(u); integer u;
              begin global integer nil,i,no,a0,A1,l;
                    global Boolean B0,B00,linear comb;
                    global procedure get atom,atom;
              end add end;
      global integer last,last1,temp0,l,der,order,no2;
      global integer array im;
      global Boolean temp,B02,linear comb;
      global procedure father,collection;
end sum;

procedure print sum(name); integer name;
begin integer n,n1,n2,n3,n4,n5;
      global integer field,length1,length,order,count,left margin,
                     der,order,control,l;
      global integer array vs;
      global Boolean B0,linear comb;
      global procedure lines,s,pi,spaces;
end print sum;
```

Source Listing 2 Cont.

```
integer procedure minimum(v,length,w);
        integer length; integer array v,w;
begin integer j,k,m,min;
      global integer length,nil;
end minimum;

integer procedure JPn(cn,cd,n,B1);
        integer cn,cd,n; integer array B1;
begin integer j,c1,c2,n1,n2,m;
      integer array B,v,w[0:n-1];
      global integer nil,last,a0,no;
      global Boolean B0,BA1;
      global procedure get atom,atom,normalize,minimum;
end JPn;

integer procedure Ze;
begin integer n,n1,j,type0;
      global integer type,nil,no,no1,no2,A,A1,a0;
      global integer array W,B2,T,num,Zp,Z,B11;
      global Boolean B0,B01,B02,tempS;
      global procedure father,son,get atom,Bncollection,JPn;
end Ze;

procedure create E(i,BB); value i; integer i; Boolean BB;
begin integer l1;
      integer array element[-1:0,0:3];
```

Source Listing 2 Cont.

```
      Boolean B;
      global integer mode,e,period,q,der,order,no,no1,no2,
                     A,A1,j,j1,nil;
      global integer array im,E,T,Z;
      global Boolean BO1,BO;
      global procedure index,father,atom;
end create E;

procedure translate(v,name1,name2,num,type);
          value v,num,type; integer v,name1,name2,num,type;
begin integer t,i,j,j1,j2,k1,k2,k3,k4,smax1,num1,num2,temp00,temp1,temp2;
      Boolean B1,BB1,BB2;
      integer array vs[0:order-1,0:smax1-1],b[0:6×k1-1],
                    a1[0:smax1-1,0:im[0]-1],Vs[0:order-1,0:smax1-1,0:im[0]-1];
      integer procedure store(u); integer u;
      begin global integer i,k1,type,t,no,A,a0,nil;
            global integer array b;
            global Boolean BO,BOO;
            global procedure abs,get atom,atom;
      end store;
      global integer origin,q,der,order,upper,last,last1,temp0,nil,no,
                     no1,no2,A,A1,i1;
      global integer array im,D,Zp,B2,B11,a;
      global Boolean linear comb,temp,BO1,BO;
      global procedure index,fact,atom,father,Bncollection,JPn,sum;
end translate;
```

Source Listing 2 Cont.

```
procedure print list(i,length,name,name of list,sign);
          value length; integer i,length,name,name of list,sign;
begin integer line,left margin,j,j1,n1,n2,n3;
      global integer left margin,field,der,order,a0,A,A1,no2,count,nil;
      global integer array im;
      global Boolean B01,B0;
      global procedure lines,spaces,pfi,pi,ps,son,get atom;
end print list;

procedure list(name of list,BB); integer name of list; Boolean BB;
begin integer j,B2;
      global integer field,nil;
      global integer array V;
      global procedure lines,spaces,s,pi;
end list;

procedure conditions E(name of vector,name);
          integer name of vector,name;
begin integer kmax,tmin,tmax,smin,i,t,left margin,n1;
      procedure B(i,j); integer i,j;
      begin global procedure ps,pfi end B;
      procedure theta(i,e1)times B:(j,e2); integer i,e1,j,e2;
      begin global integer no2,i1,q;
            global procedure ps,pfi,fact,B;
      end theta;
```

Source Listing 2 Cont.

```
      global integer field,left margin,count,n,no,no1,no2,length,length1,
                     length2,der,order,upper,e,q,i1,period,type set;
      global integer array B,v,cond;
      global procedure index,ps,pi,lines,spaces;
end conditions E;

procedure data;
begin procedure table(j,length,name); value length;
      integer j,length,name;
      begin integer procedure store;
            begin global integer no,A,i1;
                  global Boolean BO;
                  global procedure ioi,father,atom;
            end store;
            global integer no1,A1,i2,nil;
            global Boolean BO,BO1;
            global procedure ioi,iob,lines,s,list,father;
      end table;
      global integer control,type,i,no2,last,last data;
      global integer array im,a,D,W;
      global Boolean temp,left adjust;
end data;

procedure check list(first,last); integer first,last;
begin integer i,i1,j,k,type,der,no2,field1,l0,l1,l2,l3,col,num col;
      procedure fields(n2,n1); integer n1,n2;
```

Source Listing 2 Cont.

```
        begin global integer l2,l1,l0,num col,line length end;
        global integer e,q,order,field,last,nil;
        global integer array im,V,T,Z;
        global procedure page,lines,s,spaces,pi,chlength;
end check list;

procedure scheme;
begin global procedure page,lines,s end scheme;
```

Procedure Descriptions

Each procedure in RKM1 performs a well defined task when it is called. We give below a concise description of this task.

```
procedure dump(a);    ;
comment dump is used to obtain a stack dump by violating the bounds of
        the local array B[a];
procedure lines(n);    ;
comment A new line, carrage return is performed n times on channel 2 using
        the global procedure output;
procedure spaces(n);    ;
comment n spaces are written on channel 2 using the global procedure
        outcharacter;
procedure page;    ;
comment A new line, carriage return is performed on channel 2 and a new
        page is started using the global procedure output;
procedure s(string);    ;
comment Using the global procedures chlength and outcharacter, procedure
        s outputs the string -string- on channel 2;
procedure sr(string);    ;
comment if inout then sr outputs on channel 2 a right adjusted string
        -string- . The global Boolean -inout- indicates whether the user
        desires to have data read under an input - output mode; unless
        inout is true, there will be no output from this procedure. The
        program RKM1 is designed to give as output text, the totality of
        which constitutes a meaningful description of the integration
        scheme and the parameter defining equations associated with that
        scheme. When input - output of the data is desired, for example
```

when checking data, it is helpful to have the data output separated from the program output. This is accomplished by using procedure sr to right adjust the data output;

```
procedure ps(i,string);    ;
comment ps is used to output text on channel 2. It is used principally
        in the printing of equations. The integer i identifies the actual
        string parameter, the global variable print length[i] furnishes
        the length in characters of the string, and the global variable
        type set chooses the transliteration that will be used when the
        string is printed. By means of this procedure, it is possible to
        obtain the output of the parameter equations in different program
        languages. Presently there is a choice of CDC Algol and Fortran
        using either subscripted or simple variables;
procedure check margin(a);    ;
comment Using the character count specified by the global variable -count-,
        a check is made whether the quantity (count + a) exceeds the right
        margin. If so, a new line - carrage return is performed on
        channel 2 and the print position is set so that the next character
        printed will be at the margin position specified by the global
        variable left margin. An elementary exit procedure is provided to
        avoid an infinite loop should the field width be to large;
procedure pi(a);    ;
comment The integer a is output on channel 2 using the field width
        specified by the global variable -field-. The character counter
        -count- is increased by the field width, however no margin checking
        is performed;
```

```
procedure pfi(a,c);    ;
comment The integer c is output on channel 2.  The field width used is
        determined from c.  Margin checking is performed.  If type set is
        equal to 2, thus indicating Fortran output, periods are inserted
        in column 6 and the integer c is printed as a fixed point real
        number;
integer procedure ioi(string);    ;
comment ioi:= n where n is an integer input from channel 1 using standard
        format (10). if inout then n is output on channel 2 as -string:= n-.
        The output begins at (if left adjust then left margin + 5 else
        right margin - 30).  Note that the output can be omitted by setting
        inout:= false;
integer procedure iob(string);    ;
comment iob:= b where b is a Boolean input from channel 1 using standard
        format, if inout then b is output on channel 2 as -string:= b-.
        The output begins at (if left adjust then left margin + 5 else
        right margin - 30).  The value true is identified by the letter t,
        false by the letter f, all other characters are ignored.  After
        accepting the Boolean value, a delimeter consisting of a blank or
        a comma is looked for.  Thus the value true can be represented as
        t, or true, or anything with a t in it.  After recognizing the t
        the only other characters recognized are the comma or blank.
        Output can be omitted by setting inout:= false;
procedure title;    ;
comment title inputs from channel 1 and outputs on channel 2 an Algol
        comment.  The first set of characters on channel 1 should be
        < comment > followed by < any text for which there is an internal
```

representation > followed by < ., >. Margin checking is performed using the procedure check margin. An error message is furnished if an end of file is incountered on channel1;

*

integer procedure index(i); ;

comment index:= (if j1 + j = 0 mod(q) then 0 else j1 + j) where j = i mod(q) and j1 = (i - j) mod(period × q);

integer procedure fact(n); ;

comment fact:= n factorial for n an integer ≥ 0;

procedure debug(origin,a1,a2,a3); ;

comment debug furnishes an elementary monitoring of the list elements which is useful in debugging a program using the procedures father, son get atom collection, and atom;

integer procedure father(n,B0,B,name of son,son); ;

comment This procedure upon entry first sets B0:= true. It then creates a list with name -father- the elements of which are sequentially ordered with n = 0, 1, 2, ... These elements are named and created by setting name of son:= son. Elements are added to the list until B:= false. If the list -father- is empty, then the value father:= nil is returned;

integer procedure son(father entry,son1,B); ;

comment Given the non-empty list with name -father entry-, the procedure son furnishes in succession son1:= son:= the name of the next son in the list. For a given father entry, the first call to son should be indicated by previously setting B:= true;

integer procedure get atom(father entry,atomic set,atom,B); ;

comment father entry is the name of a list the elements of which are atoms. The first entry to this list is made by calling get atom with B = true.

*See Page 286 for a description of the procedure check and normalize.

```
        It then furnishes in succession atom:= get atom:= the atomic
        value.  When the list is exhausted, atom furnishes nil.  At each
        call atomic set := the atom name.  If father entry is nil, then
        the procedure furnishes nil;
integer procedure atom(i);  ;
comment This procedure stores a non-nil atomic value i in the array element
        V[last] and returns the value atom:= last, values of i which are
        nil are simply not stored;
integer procedure collection(n,B,set,f);  ;
comment collection creats a list with the name -collection- the elements
        of which are themselves lists which are sequentially ordered with
        n = 0, 1, 2, ... .  Elements are added to collection until B:= false.
        The parameter -set- is the name of a list with elements son1, son2,
        ..., soni, ... .  Given a value of n, then to each son of set there
        corresponds an element of collection with name -list- and the
        elements of list are obtained as f(soni);
Boolean procedure Bncollection(B2,B1,n,BO2,k);  ;
comment Given the lists B2[0],...,B2[n-1], Bncollection furnishes from
        these lists at each call one n-tuple of sons, (B1[0],...,B1[n-1]),
        and the value Bncollection:= true.  When all n-tuples have been
        obtained, Bncollection:= false.  If any list is an empty list,
        that is, a nil list, Bncollection:= false.  BO2 must be set to
        true to indicate the first call to Bncollection, it is returned
        as false and thereafter is to remain false.  The parameter k is
        the number of lists, starting with B2[0], for which only the first
        son is to appear in the n-tuple;
```

```
procedure sum(name,i,length of sum,name of list);    ;
comment Vectors of dimension order × im[0] are summed. The elements of the
        vectors are lists. The parameter i is the summation index starting
        with 0, name of list is the current component of the vector being
        added to the sum, length of sum is the number of items summed,
        name is the name of the current component of the sum. The component
        indices are the global variables der and no2, if linear comb then
        the sum represents a linear conbination of the vectors in which the
        coefficients are B[...], if temp then the sum is stored in temporary
        storage else in permanent storage;
procedure print sum(name);    ;
comment print sum defines the sum S[name,k], the approximations E[name,k],
        and the associated undetermined parameters B[...] by printing their
        algebraic representations. In order that parameter indexing
        remain consistent with that used in procedure sum, it is necessary
        that print sum be called immediately preceeding the call to sum
        which actually constructs the sum;
integer procedure minimum(v,length,w);    ;
comment This procedure finds min:= the minimum of the elements of v. It
        sets minimum:= m which is the number of elements of v which are
        equal to min. The array w, (w[0],...,w[m-1]), contains in an in-
        creasing order the subscripts of those elements of v which are equal
        to min. A nil element is ignored. If all v[j] are nil, then
        minimum:= nil;
integer procedure JPn(cn,cd,n,B1);    ;
comment JPn creates a normal form list of atoms corresponding to the product
        c × B1[1] × ... × B[n-1] where c is a constant cn/cd and B1[j] are
```

```
        lists of atoms in normal form.  A normal form list is of the form
        (c1,c2,index[0],exponent[0],...,index[i],exponent[i],...,index[p],
        exponent[p]) with c1,c2 relatively prime integers, and index[i] <
        index[j] if i < j.  Note, at the present time the constants c1
        and c2 are not made relatively prime since this has not been
        necessary;
integer procedure Ze;    ;
comment Ze performs if type = 1 then a substitution X(E[...]) else if
        type = 2  then a multiplication DX(E[...]) × S[...].  The work
        is carried out in the coefficient space of the functions A(0)[i]
        using tables W[type,no2] to represent the substitution and multipli-
        cation operators;
procedure create E(i,BB);    ;
comment The approximation E[i] is created with undetermined parameters.
        if mode = -1 then
        E[i]:= u[-i1×h](T[-1,i]) +
        (if BB then sum(i,0,im[0]-1,B[...+i] × A(-i1×h)[i]) else nil)
        else if mode = 0 then
        E[i]:= u[0](T[0,i]) +
        (if BB then sum(i,0,im[0]-1,B[...+i] × A(0)[i]) else nil);
procedure translate(v,name1,name2,num,type);    ;
comment The procedure translate is used to translate a vector with name -v-
        of dimension ( im[0] × order ) the elements of which are lists.
        The original component name is -name1-, the final component name is
        name2, the number of intervals is -num-, the translation is re-
        presented by a table a[type,...,...].  This table nominally represents
        a translation of one h interval, the procedure substitutes the
```

correct value. The translation is carried out in the coefficient space of the functions A(0)[...]. Note that if type = -1 then the translation is in the -h direction else if type = 0 then the translation is in the +h direction;

procedure print list(i,length,name,name of list,sign); ;

comment For (0 ≤ i < length) print list outputs in algebraic form (order × im[0]) normal form lists which have the structure father(...,father(...)). The parameter -name of list- is the name of the list to be printed, -name- identifies the output, -sign- determines the sign of addition between lists, 0 for - and 1 for +. A normal form list is defined in the description of procedure JPn;

procedure list(name of list,BB); ;

comment The procedure list outputs on channel 2 the list with name -name of list-, if BB then a heading is provided;

procedure conditions E(name of vector,name); ;

comment In order that E[name of vector]:= u[0](...) + sum(i,0,im[0],B[... + i] × A(0)[i]), where A(0)[i] are the basic functions, be a valid representation of the approximation it is necessary that the parameters B[... + i] satisfy certain conditions. The procedure conditions E prints these conditions in algebraic form as C[name, der,t]:= (the equations arrising from these conditions).

procedure data; ;

comment data reads from channel 1 a data table and stores it in a list with structure father(...,father(...)), that is, a list with father, sons, atoms. The table is actually stored by the procedure table which makes the list. The number of fathers is -length-, the number of sons is -i2-, the number of atoms is -i1-. The table

represents if control = 0 then derivative harmonics else if control = 1 then translation harmonics else if control = 2 ∧ type = 0 then a substitution table else if control = 2 ∧ type = 1 then a multiplication table. Note that if BO then the list representing the table is output on channel 2;

procedure check list(first,last); ;

comment When called, check list prints the list storage array V[i], i = first,(1),last - 1, and the names of the lists Z[type,der,no3[type], no2] and T[type,no3[type]]. This procedure can by called at any time without changing any of the local or global parameters of RKM1;

procedure scheme; ;

comment scheme defines the problem considered and furnishes definitions which along with the program output define a method of solution;

procedure check; ;

comment if last ≧ temp0 ∧ ¬ temp ∨ last ≧ list length then a message is printed, the list elements V[i], 0 ≦ i ≦ last are dumped and the program RKMI is stopped;

procedure normalize(a,b); ;*

comment given the integers a,b the procedure normalize returns them as relatively prime integers a,b;

*See the description of JPn on page 284 and also the bottom of page 183.

Variable List

The following is a list of the principal variables used in RKM1. It is not inclusive; it should, however, prove helpful to the reader who wishes to understand how the program works.

Global Variables

integer

identifier	comment
field	The field width used in printing a numerical quantity.
decimal	The number of decimal digits in a printed number.
n for print	After n for print approximations E[no3[0]] have been created, the procedure debug will be activated and will print.
nil	Consistently used to indicate the empty set or zero.*
line length	The length in characters of a print line. Used principally to determine the print line length of the printed (punched) parameter defining equations.
left margin	The position of the left margin.
right margin	The position of the right margin.
last data	A list pointer pointing to the first available storage location after storing all the data tables.
l	The next free parameter index for the parameters B.
lastl	Many parameter values B[i] are reserved for interval parameters and undetermined parameters, lastl is the first free parameter available for the scheme parameters.
temp0	The beginning of temporary storage.

*Presently nil is represented by -2 ↑ 48.

Variable List Cont.

identifier	comment
type	Usually used to determine the type of a quantity in the following sense: there are approximations, approximators, and sums, and the approximations can have local origin expansions or origin expansion. See the table in Appendix III, Source Listing I and note the various references to type.
der	A counter rather consistently used to represent the derivative index k in P = (0,...,p-1) associated with E[i,k].
order	The order of the differential equation. Refered to as p in the text. D↑p x = X o E, order = p.
upper	The lower bound 1 of the lowest order approximation, upper = 1, see Chapter II, equation (3).
q	The rank of the scheme which is the number of approximations in one h interval.
e	The extent of the scheme which is the total number of h intervals considered.
count	A pointer indicating the current print position.
height	The value of the dimension limiting the number of approximators N or sums S that may be constructed. One cannot construct more than height sums S or approximators N.
height1	The value of the dimension limiting the maximum number of items that may appear in a sum S. One cannot construct a sum that contains more than height1 items.
pmax	The value of the dimension limiting the maximum number of lists that may be multiplied together. Note that a substitution (multiplication) requires $\theta^j = \theta_i \times \ldots \times \theta_i$ which

Variable List Cont.

identifier	comment
	is j multiplications and that the θ_i are stored as lists. However, the dimension is input data and can easily be set as large as needed.
last	A list pointer pointing to the next available storage location.
last1	A list pointer usually used to remember where last was when last is temporarily shifted.
period	The period of the scheme.
origin	The location of the origin, presently set to 0 with the understanding that zero corresponds to B[0] = 0. Remember that the development of the results of Chapters II, III, IV, and V never fixed the origin, but presently RKM1 does.
mode	The type of translation, mode = -1 implies a forward translation (undetermined parameters created about the local origin), mode = 0 implies a backward translation (undetermined parameters created about the origin).
mode1	The variable mode is in the set (0,-1). The variable mode1 always has the other value of mode.
length	The number of items appearing in a sum S.
length1	The number of approximations E appearing in a sum S.
length2	The number of approximators N appearing in a sum that were created by means of a substitution.
control	A simple variable used exclusively by the user to control the action of the program.
list length	The dimension limiting the maximum length of the list storage array. This includes the temporary storage.

Variable List Cont.

identifier	comment
n	A counter
no	Usually sequences the elements of a list of level 0.
no1	Usually sequences the elements of a list of level 1.
no2	Usually sequences the elements of a list of level 2.
a0	Usually the value of an atom from the list.
A	Usually the name of an atom.
A1	Usually the name of an element of a list of level 1
i, i1, i2, j, j1	Very often simply counters. Remember that these are global variables here. Beware, when the procedure index is used, it sets i1, i2, j, j1 to values with a particular significance. See Chapter V.
type set	Selects the language of the equation output. See Appendix III.
Boolean	
temp	The value true means that storage in the list will take place in the section reserved for temporary storage.
print scheme	The value true means that the scheme description will be printed.
linear comb	The value true means that the procedure sum will perform a linear combination of the given items. Thus new parameters will multiply each item of the sum. The value false means that the procedure sum will simply sum the items.
BOO	Global Boolean available for general use. See, however, procedure collection.

Variable List Cont.

identifier	comment
BO	Used principally as an actual parameter of the procedure father for lists of level 0.
BO1	Used principally as an actual parameter of the procedure father for lists of level 1.
BA1 BA2	Used exclusively in the packed version of atom and get atom to indicate which half of the word we are currently working in. A change in the number of atoms per word requires thought about the use of BA1 and BA2. The unpacked versions of atom and get atom make no use of these variables.
left adjust	The value true means that output will be left adjusted, otherwise it is right adjusted.
inout	The value true means that all input will be output on channel 2. The value false means no input will be output.
integer array	
im	im[0] is the number of fasis functions. We implicitly assume that we will choose all the harmonics corresponding to some order r of the tables given in Appendix I. This is not absolutely necessary for RKM1 but is necessary if the output is to contain all the equations. im[1] is the number of derivatives we attempt to match in $v[m] = \sum_{r=0}^{im[0]-1} D^{p+k+r}{}_x \frac{I^{k+\ell+1+r}}{(k+\ell+1+r)!}$. Since r corresponds to the order mentioned above with regard to im[0], we see that there is one derivative for each order of the differentials that are the basis functions.
no3	The names of various quantities. This is usually used as no3[type] where type indicates what quantity we are

Variable List Cont.

identifier	comment
	refering to. See the description of type.
cond	The names of approximators that have conditions attached to their parameters. They have conditions when they have been used in a sum and the approximator was created by means of a substitution.
T	The names (list entry points) of the interval parameters θ_i in the quantities $u(\theta_i)$.
E	If an approximation ξ_i has been created then E[0,i] is zero. If undetermined parameters have been used to create this approximation, then E[-1,i] is zero. The quantity E[0,i] is the smallest index of the parameters appearing in ξ_i. E[1,i] is the number of approximations appearing in ξ_i, E[2,i] is the number of approximators appearing in ξ_i that have been created by a substitution, E[3,i] is the total number of items in the sum.
V	This is the list storage array. All list values are stored in V.
Z	Names of the lists. We usually have Z[type,der,i,no2]. The variable type gives the quantity we are identifying. See the description of type.
Zp	Names of the lists. Used in the same fashion as Z except that these lists are stored in temporary storage. The name is valid for an item only as long as temporary storage is not used again. Remember, translation may use temporary storage. Thus a sum that is stored temporarily may be destroyed if one subsequently creates an approximation with

Variable List Cont.

identifier	comment
	undetermined parameters before using S.
v	The quantities v[0,no3[0],i] are the names of the approximations appearing in the construction of ξ_j , j = no3[0]. The quantities v[1,no3[0],i] are the names of the approximators appearing in the construction.
vs	Temporary storage of names.
W	The quantities W[1,i] are entry points to the substitution table, W[2,i] are entry points to the multiplication table. See procedures data and Ze.
D	The quantities D are derivative harmonics. See procedures data and translate.
a	The quantities a are translation harmonics. See procedures data and translate.
B11 B2	Global storage for names when multiplying lists. See procedures Ze and translate.

RKMI ALGOL 60 SOURCE LISTING

The following source listing has been prepared from a flexowriter tape that contains the program RKMI. The only non-standard ALGOL symbol is the use of ⊔ for the string space character ␣. In order that this listing be as accurate as possible, the flexowriter tape was translated by machine to CDC ALGOL card images and that source deck was then compiled and run on the two examples, Example 1 and 2, for which we have presented complete data. The results obtained agree with those presented here. The few errors that were detected during this process were then corrected on the flexowriter tape.

At the end of the source listing, we have presented a list of known restrictions and omissions. These are of a minor nature and have not affected our use of the program.

The source listing presented on pages 295 through 336 is the program RKMI. The versions of atom and get atom which appear in this listing are packed two atoms per word. To obtain an unpacked version of RKMI, replace these two procedures by the procedures atom and get atom that are given on page 337.

```
begin   comment The following program, RKM1, is a generator of generalized Runge-Kutta-Frey type schemes
        with memory including 1st derivative DX to be used in the numerical solution of n-th.
        order systems of p-th. order ordinary differential equations. All parameter defining equations
        are output along with the scheme definition;

        integer field,decimal,n for print,nil,line length,left margin,right margin,count,height,height1,pmax,
        last,last1,last data,l,lastl,temp0,type,der,order,upper,q,e,period,origin,mode,mode1,length,
        length1,length2,control,list length,n,no,no1,no2,a0,A,A1,i,i1,i2,j,j1,name,type set;

        Boolean temp,tempS,print scheme,linear comb,B00,B0,B01,B02,BA1,BA2,left adjust,inout;

        integer array im[0:1],print length[0:12];

begin

        procedure dump(a); integer a;
        begin integer array B[0:1]; B[a] := 1 end dump;
```

```
procedure title;
begin integer i,j,k,left margin1;
      eof(1,exit); left margin1:= left margin; j:= i:= 0;
      for j:= j + i while j ≠ 24 do incharacter(1,'coment',i); lines(1); s('comment'); k:= j:= 0;
      for j:= i while (i ≠ equiv(',') ∨ k ≠ equiv('.')) do
      begin k:= i; input(1,'a',i); check margin(1); count:= count + 1; output(2,'a',i) end;
      go to end;
exit: s('eof␣incountered␣in␣title');
end:  left margin:= left margin1
end title;

procedure lines(n); value n; integer n;
begin integer i; for i:= 1 step 1 until n do output(2,'/'); count:= 1 end lines;

procedure spaces(n); value n; integer n;
begin integer i; for i:= 1 step 1 until n do outcharacter(2,'␣',1); count:= count + n end spaces;

procedure page; output(2,'↑');
```

```
procedure s(string); string string;
begin integer i,length;
      length:= chlength(string);
      for i:= 1 step 1 until length do outcharacter(2,string,i); count:= count + length
end s;

procedure sr(string); string string;
if inout then begin lines(1); spaces(line length - 30); s(string) end sr;

procedure ps(i,string); integer i; string string;
begin check margin(print length[i]);
      if typeset = 1 then begin if count = left margin then spaces(1); s(string) end else
      if typeset = 2 then
      begin if count = 6 then s('.'); if i = 1 ∨ i = 3 then s('(') else
            if i = 2 ∨ i = 4 then s(')') else if i = 9 then s('××') else
            if i = 10 then s('=') else if i ≠ 12 then s(string)
      end else
      if typeset = 3 then
      begin if count = left margin then spaces(1); if i ≠ 1 ∧ i ≠ 2 ∧ i ≠ 11 then s(string) end else
```

```
      if typeset = 4 then
      begin if count = 6 then s('.'); if i = 3 then s('(') else
            if i = 4 then s(')') else if i = 9 then s('××') else
            if i = 10 then s('=') else if i ≠ 12 ∧ i ≠ 11 ∧ i ≠ 1 ∧ i ≠ 2 then s(string)
      end
end ps;

procedure check margin(a); value a; integer a;
begin integer i; i:= 0;
      again: i:= i + 1; if i > 2 then go to exit;
      if count + a > right margin then
      begin lines(1); spaces(left margin - 1); go to again end;
      go to fin;
exit: s('check␣margin');
fin:
end check margin;
```

```
procedure pi(a); value a; integer a;
begin procedure layout; format('-xzd',field - 2);
      procedure list(item); procedure item; item(a);
      count:= count + field;
      if a = 0 then begin spaces(field - 1); s('0') end else outlist(2,layout,list)
end pi;

procedure pfi(b,c); value b,c; integer b,c;
begin integer a,n,n1;
      n:= c; a:= 2;
      for n1:= a + 1 while n | 0 do begin n:= n ÷ 10; a:= n1 end;
      if (right margin - left margin) < a then s('line␣length␣<␣field␣width') else
      begin procedure layout; format('-xd',a - 2);
            procedure list(item); procedure item; item(c);
            check margin(a + 1);
            if count = 6 ∧ (typeset = 2 ∨ typeset = 4) then s('.');
            if c = 0 then s(',0') else begin outlist(2,layout,list); count:= count + a - 1 end;
            if b = 100 ∧ (typeset = 2 ∨ typeset = 4) then s('.')
      end
end pfi;
```

```
integer procedure ioi(string); string string;
begin integer n;
      input(1,'',n); ioi:= n; if inout then
      begin lines(1); spaces(if left adjust then 5 else line length - 30); s(string); s('␣:=');
            if n = 0 then s('0') else output(2,'-4zd',n)
      end
end ioi;

Boolean procedure iob(string); string string;
begin Boolean b; integer i,j;
      j:= 0; eof(1,exit); i:= 0;
      for j:= j + 1 while i ≠ 1 ∧ i ≠ 2 do incharacter(1,'tf',i); b:= (i = 1); j:= i:= 0;
      for j:= j + 1 while i ≠ 1 ∧ i ≠ 2 do incharacter(1,',␣',i);
      if inout then
      begin lines(1); spaces(if left adjust then 5 else line length - 30);
            s(string); s('␣:=␣'); if b then s('true') else s('false')
      end; go to end iob;
exit: s('eof␣read␣in␣procedure␣iob');
end iob: iob:= b
end iob;
```

```
begin:  inout:= left adjust:= true; count:= 0; field:= 3; decimal:= 0; page;
        if ioi('control') < 1 then go to end of computations else lines(2);
        s('␣data␣input␣that␣sets␣computer␣variables'); lines(2);
        s('i:=␣0;␣for␣print␣length[i]:='); inout:= false;
        for i:= 0 step 1 until 12 do
        begin print length[i]:= ioi(''); pi(print length[i]); if i ≠ 12 then s(',') end;
        s('␣do␣i:=i+1;'); inout:= true; typeset:= ioi('typeset');
        n for print:= ioi('n␣for␣print'); line length:= ioi('line␣length'); left margin:= 2;
        right margin:= line length - 2; temp0:= ioi('origin␣of␣temporary␣store');
        list length:= ioi('list␣length'); height:= ioi('maximum␣number␣of␣vectors␣N␣or␣sums␣S');
        height1:= ioi('maximum␣length␣of␣a␣sum'); pmax:= ioi('maximum␣length␣of␣a␣list␣product');
define problem: lines(2);
        left adjust:= true; s('data␣input␣that␣particularizes␣the␣problem'); lines(2);
        order:= ioi('order␣of␣the␣differential␣equation'); upper:= ioi('D↑(order␣+␣upper)x␣is␣the␣
        first␣neglected␣Taylor␣term,␣upper'); q:= ioi('number␣of␣points␣in␣one␣h␣interval');
        e:= ioi('number␣of␣h␣intervals'); period:= ioi('period␣of␣scheme'); im[0]:= ioi('number␣
        of␣basic␣functions'); im[1]:= ioi('number␣of␣derivatives␣we␣attempt␣to␣match');
        left adjust:= false;
```

```
begin   comment variable array declarations followed by procedure declarations;
        integer array   num,no3[-1:2],cond[0:height - 1],T[-1:0,0:exq],E[-1:3,0:exq],V[0:list length],
                        Z[-1:2,0:order - 1,0:exq,0:im[0] - 1],Zp[0:order - 1,0:im[0] -1],
                        v[0:1,0:exq,0:height1 - 1],vs[0:1,0:height1 - 1],W[1:2,0:im[0] - 1],
                        D[0:im[1] -1,0:im[0] -1],a[-1:0,0:im[0]-1,0:im[0] -1],B11,B2[0:pmax];

        procedure check; if last ≥ temp0 ∧ ¬ temp ∨ last ≥ list length then
        begin lines(1); s('overlap,,last,:=');  field:=5; pi(last);
              check list(0,last); go to end of computations
        end check;

        procedure normalize(a,b); integer a,b; ;

        integer procedure index(i); integer i;
        begin i1:= i + q; j:= i - i1 × q; i2:= i1 + period; j1:=(i1 - i2 × period) × q;
              index:= if (j1 + j) - ((j1 + j) ÷ q) × q = 0 then 0 else j1 + j
        end index;
```

```
integer procedure fact(n); value n; integer n;
begin   integer i,j; if n = 0 then fact:= 1 else
        begin j:= 1; for i:= 1 step 1 until n do j:= i × j; fact:= j end
end fact;

procedure debug (origin,a1,a2,a3); integer origin,a1,a2,a3; ;

integer procedure father(n,B0,B,name of son,son); integer n,name of son,son; Boolean B,B0;
begin   integer copy,son1;
        n:= 0; copy:= last; last:= last + 1; B0:= true;
again:  if B then  begin name of son:= son1:= son; debug(1,son1,nil,nil); n:= n+1; go to again end
        else if copy = last-1 then begin father:= nil; last:= last-1 end
        else begin father:= copy; V[copy]:= last; debug(2,copy,nil,nil) end
end father;

integer procedure son(father entry,son1,B); integer father entry, son1; Boolean B;
begin   if B then
        begin if father entry = nil then begin son1:= son:= nil; go to end end;
              B:= false; son1:= son:= father entry + 1
```

```
        end else
        begin son1:= son:= if V[son1] < V[father entry] then V[son1] else nil end;
end:    debug(3,father entry, son1,V[son1])
end son;

integer procedure atom(i); value i; integer i;
comment packed two atoms/word;
begin atom:= last; if no = 0 then BA2:= true; if i ≠ nil then
begin if abs(i) > 8 388 607 then
    begin lines(1); s('atom␣too␣large:='); pfi(0,i);
        checklist(0,last); go to end of computations
    end else
    begin integer copy,n1,n2;
        check; n1:= 16 777 216; n2:= 8 388 608;
        if BA2 then
        begin V[last]:= abs(i) × n1 + n2; if i < 0 then V[last]:= -V[last];
            BA2:= ¬ BA2; last:= last + 1;
        end else
        begin last:= last - 1; copy:= abs(V[last]) + abs(i);
            if i ≥ 0 then copy:= copy - n2; if V[last] < 0 then copy:= - copy;
```

```
                    V[last]:= copy; last:= last + 1; BA2:= ¬ BA2
                end
        end; debug(6,i,nil,nil)
end
end atom;

integer procedure get atom(father entry,atomic set,atom,B);
integer father entry,atomic set,atom; Boolean B;
comment packed two atoms/word;
begin integer n1,n2;
        n1:= 16 777 216; n2:= 8 388 608; if B then
        begin if father entry = nil then begin get atom:= atom:= nil; go to end end;
                B:= ¬ B; atomic set:= father entry + 1; BA1:= true
        end else if BA1 then atomic set:= atomic set + 1;
        if atomic set < V[father entry] then
        begin if BA1 then
                begin get atom:= atom:= V[atomic set] + n1; BA1:= ¬ BA1 end else
                begin atom:= abs(abs(V[atomic set]) - abs((V[atomic set] + n1) × n1));
                        if atom = n2 then begin get atom:= atom:= nil; go to end end;
```

```
                    if atom > n2 then atom:= -(atom - n2);
                    get atom:= atom; BA1:= ¬ BA1
                end
            end else get atom:= atom:= nil;
end: debug(4,father entry,atomic set,atom)
end get atom;

integer procedure collection(n,B,set,f); integer n,set; Boolean B; integer procedure f;
begin   integer copy,son1; Boolean BB;
        n:= -1; copy:= last; last:= last + 1; BB:= true;
        for n:= n + 1 while B do
        begin if set ≠ nil then
              for n:= n while son(set,son1,BB) ≠ nil do
              begin BOO:= true; father (no,BO,BOO,A,f(son1)) end;
              BB:= true
        end; if copy = last - 1
        then begin collection:= nil; last:= last- 1 end
        else begin collection:= copy; V[copy]:= last; debug(5,copy,nil,nil) end
end collection;
```

```
Boolean procedure Bncollection(B2,B1,n,BO2,k); value n,k;
integer n,k; integer array B2,B1; Boolean BO2;
begin   integer j; own integer array B11[0:20]; own Boolean array BO1[0:20];
        if BO2 then
        begin BO2:= false; for j:= 0 step 1 until n-1 do
              begin if B2[j]= nil then begin Bncollection:= false; go to end end;
                    BO1[j]:= true; B1[j]:= son(B2[j],B11[j],BO1[j])
              end; Bncollection:= true; go to end
        end else if n = 1 then begin Bncollection:= false; go to end end
        else B1[k]:= son(B2[k],B11[k],BO1[k]);
        for j:= k step 1 until n-1 do
        begin if B1[j]= nil then
              begin if j = n-1 then begin Bncollection:= false; go to end end;
                    B1[j+1]:= son(B2[j+1],B11[j+1],BO1[j+1]); BO1[j]:= true;
                    B1[j]:= son(B2[j],B11[j],BO1[j])
              end
        end; Bncollection:= true;
end:
end Bncollection;
```

```
procedure sum(name,i,length of sum,name of list); value length of sum;
integer name,i,length of sum,name of list;
begin   integer l1;
        integer procedure add end(u); integer u;
        begin BOO:= (get atom(u,A1,a0,BO) ≠ nil);
              if BOO then add end:= atom(a0)
              else if linear comb then begin add end:= atom(i + 1); add end:= atom(1) end
        end add end;
        if temp then begin last1:= last; last:= temp0 end; l1:= 1; der:= -1;
        for der:= der + 1 while der < order do
        begin l:= l1 + der × length of sum;
              father(no2,BO2,no2 < im[0],name,collection(i,i < length of sum,name of list, add end));
        end; l:= if linear comb then l1 + order × length of sum else l1;
        if temp then begin l1:= last; last:= last1; last1:= l1 end
end sum;
```

```
procedure print sum(name); integer name;
begin   integer n1,n2,n3,n4,n5;
        n4:= 3; n5:= 4; field:= n4; n1:= if length1 ≠ 0 then 0 else 1;
        n2:= if length - order × length1 ≠ 0 then 1 else 0;
        BO:= true; lines(2); count:= left margin;
        for der:= 0 step 1 until order - 1 do
        begin if control = 2 then s('S') else s('E'); s('['); pi(name); s(','); pi(der); s(']:=');
              for n:= n1 step 1 until n2 do
              begin lines(1);
                    L1: s('sum('); if BO then s('i') else s('j'); s(',0,');
                    pi(if n=0 ∧ BO then length1-1 else if n=0 ∧ ¬ BO then order-1 else
                       length-order×length1-1);
                    s(','); if BO ∧ n = 0 then begin BO:= false; go to L1 end;
                    if linear comb then
                    begin field:= n5;
                          s('B['); pi(1 + der×length + (if n = 0 then 0 else order×length1));
                          field:= n4; s('+i'); if n = 0 then begin s('×'); pi(order); s('+j') end;
                          s(']×')
                    end; if n = 0 then s('E') else s('N'); s('[v['); pi(n); s(',i]');
                    if n = 0 then begin s(','); s('j') end; s('])');
```

```
                    if n = 0 then s(')+');
                    BD:= true; if n = n2 then lines(2);
                end
        end; lines(1); field:= n5;
        for n:= n1 step 1 until 1 do
        begin s('where␣␣v['); pi(n); s(']:=(');
            n2:= (if n = 0 then length1 else length-order×length1) - 1;
            for n3:= 0 step 1 until n2 do
            begin pi(vs[n,n3]); if n3 ≠ n2 then s(',') else s(')') end;
            s('␣␣ and ␣␣v['); pi(n); s(',i]:=␣v['); pi(n); s(']['i']'); lines(1)
        end
end print sum;

integer procedure minimum(v,length,w); integer length; integer array v,w;
begin   integer j,k,m,min;
        k:=length-1; m:= 0; min:= 9999;
        for j:= 0 step 1 until k do if v[j] < min ∧ v[j] ≠ nil then min:= v[j];
        for j:=0 step 1 until k do if v[j] = min then begin w[m]:= j; m:= m + 1 end;
        minimum:= if min = 9999 then nil else m
end minimum;
```

```
integer procedure JPn(cn,cd,n,B1); integer cn,cd,n; integer array B1;
begin   integer j,c1,c2,n1,n2,m; integer array B,v,w[0:n-1];
        if n = 0 ∨ n = nil then begin JPn:= last; BO:= false; go to fin end;
        c1:= cn; c2:= cd; n1:= n-1; n2:= 1;
        for j:= 0 step 1 until n1 do
        begin BO:= true; c1:= c1 × get atom(B1[j],B[j],a0,BO); c2:= c2 × get atom(B1[j],B[j],a0,BO) end;
        normalize(c1,c2); JPn:= atom(c1); no:= 1; JPn:= atom(c2);
        for j:= 0 step 1 until n1 do
        begin BA1:= true; if get atom(B1[j],B[j],a0,BO) = nil then
                        begin if n2 = n then go to fin else n2:= n2+1 end; v[j]:= a0
        end;
again:  m:= minimum(v,n,w); atom(v[w[0]]); c1:= 0;
        for j:= 0 step 1 until m-1 do
        begin BA1:= false; c1:= c1 + get atom(B1[w[j]],B[w[j]],a0,BO);
              if get atom(B1[w[j]],B[w[j]],a0,BO) = nil then n2:= n2 + 1; v[w[j]]:= a0
        end; if c1 = 0 then last:= last - 1 else JPn:= atom(c1);
        if n2 ≤ n then go to again;
fin:
end JPn;
```

```
integer procedure Ze;
begin   integer n,n1,j,type0;
        BO2:= true;
        if son(W[type,no2],A1,BO2) = nil then Ze:= nil else
        begin B2[0]:= W[type,no2]; n1:= 0; j:= 0;
              for j:= j + 1 while son(B2[0],A1,BO2) ≠ nil do
              begin BO:= true; type0:= if j = 2 ∧ type = 2 then type else 0;
                    for n:= n1 + 1 while get atom (A1,A,a0,BO) ≠ nil ∧ a0 ≥ 0 do
                    begin if j = 1 then
                          begin for n1:= n step 1 until n + a0 - 1 do B2[n1]:= T[0,num[0]];
                                n1:= n + a0 - 1
                          end else
                          begin B2[n]:= if j = 2 ∧ tempS ∧ type = 2 then Zp[a0,get atom(A1,A,a0,BO)]
                                        else Z[type0,a0,num[type0],get atom(A1,A,a0,BO)]; n1:= n
                          end
                    end
              end; n:= n1 + 1;
              Ze:= father(no1,BO1,Bncollection(B2,B11,n,BO1,1),A1,father(no,BO,BO,A,JPn(1,1,n,B11)))
        end
end Ze;
```

```
procedure create E(i,BB); value i,BB; Boolean BB; integer i;
begin integer l1; integer array element[-1:0,0:3]; Boolean B;
      l1:= index(i); B:=(l1 = 0 ∧ mode = -1) ∨ (i = 0 ∧ mode = 0);
      element[0,1]:= element[0,3]:= element[-1,1]:= element[-1,3]:= element[0,0]:= 1;
      element[0,2]:= 2 × period × q + i; element[-1,0]:=  if l1 = 0 then 0 else 1; element[-1,2]:= l1;
      for l1:= -1,0 do
      T[l1,i]:= father(no1,BO1,no1 < 1,A1,father(no,BO,no < 4,A,atom(if element[l1,0] = 0 then
                                                                     nil else element[l1,no])));
      E[0,i]:= 0;
      if BB then
      begin der:= -1; element[0,0]:= 1;
            for der:= der + 1 while der < order do
            for no2:= 0 step 1 until im[0] -1 do
            begin element[0,2]:= 1 + 2 × period × q + e × q + (if mode = -1 then (j + j1) else i)
                                 × im[0] × order + der × im[0] + no2;
                  Z[mode,der,i,no2]:= if B then nil else father(no1,BO1,no1 < 1,A1,father(no,BO,
                                                                no < 4,A,atom(element[no1,no])))
            end; if ¬ B then E[-1,i]:= 0
      end
end createE;
```

```
procedure translate(v,name1,name2,num,type); value v,num,type; integer v,name1,name2,num,type;
begin   integer t,i,j,j1,j2,k1,k2,k3,k4,smax1,num1,num2,temp00,temp1,temp2;
        Boolean B1,BB1, BB2;
        if (v ≥ origin ∧ v < origin + q) then
        begin for der:= 0 step 1 until order - 1 do
              begin for no2:= 0 step 1 until im[0] -1 do name2:= name1 end; go to end translate
        end;  temp1:= last1; last1:= last; last:= temp00:= temp0; smax1:= im[0] + im[1];
              B1:= (type = -1); BB2:= linear comb; linear comb:= false; BB1:= temp; temp:= true;
              k1:= order + upper;
        if B1 then begin k1:= (k1↑2 - k1) ÷ 2 + k1; num2:= -1 end else num2:= 1;
        begin   integer array vs[0:order-1,0:smax1-1],b[0:6×k1],a1[0:smax1-1,0:im[0]-1],
                                Vs[0:order-1,0:smax1-1,0:im[0]-1];
                integer procedure store(u); integer u;
                begin if B0 then
                      begin k1:= -1;
                            for k1:= k1 + 1 while get atom(u,A,a0,B0) ≠ nil do
                            begin i:= k1; b[i]:= a0 end
                      end; B00:= (no ≤ i);
```

```
        if BOO then store:= atom(if(no=0 ∧ type =-1 ∧ i=3) then (abs(b[0])×((-1)↑b[3]))
                                 else if (no=0 ∧ type=0) then abs(b[0])
                                 else if no=2 then t else b[no])
               else store:= nil
end store;
j:= index(v); t:= (2 × period + i1) × q;
if type = -1 then j:= 2 × period × q + v;
if j < t then begin k3:= 2; k4:= 4 end else begin k3:= 4; k4:= 2 end;
for no2:= 0 step 1 until smax1 -1 do
begin for der:= 0 step 1 until order - 1 do
begin k1:= der + upper; k2:= 0;
      if no2 < im[0] then vs[der,no2]:= name1
      else begin for i:= 0 step 1 until k1 do
           begin j1:= no2 - im[0] + upper - i + der + 1;
                 b[k2]:= (num2↑j1);
                 b[k2+1]:= fact(j1) × fact(i);
                 b[k2+k3]:= j; b[k2+k3+1]:= i; b[k2+k4]:= t;
                 b[k2+k4+1]:= j1; k2:= k2 + 6
```

```
                end;  vs[der,no2]:= father(no1,B01,no1 < k2+6,A1,father(no,B0,no < 6,A,
                                                atom(if (b[6×no1+2] = 0 ∧ b[6×no1+3] ≠ 0) ∨
                                                    (no-2×(no+2) = 0 ∧ b[6×no1+no+1] = 0) ∨
                                                    (b[6×no1+no] = 0) then nil
                                                    else b[6×no1+no])))
                end
end; j1:= last;
     for j2:= 0 step 1 until im[0] - 1 do
     if no2 < im[0] then
     begin last:= a[0,no2,j2];
          a1[no2,j2]:= if last = nil then last else
                       collection(no1,no1 < 1,a[0,no2,j2],store)
     end else a1[no2,j2]:= D[no2 - im[0],j2]; last:= j1
end; temp2:= last; last:= last1; temp0:= temp2; num:= 1;
for num1:= 0 step 1 until num-1 do
begin last1:= last; last:= temp0; temp:= true; B1:= (num1 < num -1);
      for der:= 0 step 1 until order -1 do
      for no2:= 0 step 1 until if B1 then im[0] - 1 else smax1 -1 do
```

```
                    for j:= 0 step 1 until im[0] -1 do
                    begin B2[0]:= if no2 < im[0] ∧ num1 ≠ 0 then Zp[der,no2] else vs[der,no2];
                          B2[1]:= a1[no2,j]; BO1:= true;
                          Vs[der,no2,j]:= father(no1,BO1,Bncollection(B2,B11,2,BO1,0),A1,
                                                 father(no,BO,BO,A,JPn(1,1,2,B11)))
                    end;
                    j1:= last1; last1:= temp0:= last; last:= j1;
                    temp:= false; sum(name2,j,smax1,Vs[der,j,no2])
               end; temp0:= temp00; last1:= temp1
          end; temp:= BB1; linear comb:= BB2;
     end translate:
end translate;
```

```
procedure print list(i,length,name,name of list,sign); value length;
integer i,length,name,name of list,sign;
begin   integer line,left margin1,j,j1,n1,n2,n3;
        left margin1:= left margin; lines(1);
        if type set = 2 ∨ type set = 4 then n3:= 1 else n3:= 0;
        if type set = 2 ∨ type set = 4 then left margin:= 6;
        n1:= 3; n2:= 4; der:= -1; j:= 0;
        for der:= der + 1 while der < order do
        begin no2:= -1; for no2:= no2 + 1 while no2 < im[0] do
              begin lines(1); spaces(left margin); field:= n1; line:= 1;
                    ps(0,'c'); ps(1,'['); pi(name); ps(11,','); pi(der + n3);
                    ps(11,','); pi(no2 + n3); ps(2,']'); ps(10,':='); j1:= 0; i:= -1;
                    for i:= i + 1 while i < length do if name of list ≠ nil then
                    begin B01:= true; j:= j + 1;
                    L1:   B0:= true; if son(name of list,A1,B01) = nil then go to end of list;
                          if line < 3 then begin line:= line + 1; lines(1); spaces(left margin - 1) end;
                          if sign < 0 then ps(5,'␣-␣') else ps(6,'␣+␣');
                          ps(3,'('); pfi(100,get atom(A1,A,a0,B0)); ps(8,'/');
                          pfi(100,get atom(A1,A,a0,B0)); ps(4,')'); field:= n2; j1:= 0;
                          for j1:= j1 + 1 while get atom(A1,A,a0,B0) ≠ nil do
```

```
                    begin ps(7,'␣×␣'); ps(0,'b'); ps(1,'['); pfi(0,a0); ps(2,']');
                        if get atom(A1,A,a0,B0) ≠ 1 then begin ps(9,'↑'); pfi(0,a0) end
                    end; go to L1;
                end of list:
                end; if j = 0 then pi(0); ps(12,';'); lines(1)
            end; lines(2)
        end; name:= name + 1; left margin:= left margin1
end print list;

procedure list(name of list,BB); integer name of list; Boolean BB;
begin   integer j,B2; B2:= name of list;
        if BB then begin lines(2); BB:= false; s('␣␣␣i'); spaces(4); s('V[i]'); lines(1) end;
        if B2 = nil then s('␣␣nil␣␣') else
        begin for j:= B2 step 1 until V[B2] - 1 do
            begin lines(1); field:= 4; pi(j); spaces(4); field:= 18; pi(V[j]) end
        end
end list;
```

```
procedure conditions E(name of vector,name); integer name of vector,name;
begin integer kmax,tmin,tmax,smin,i,t,left margin1,n1;
      procedure B1(i); integer i;
      begin ps(0,'b'); if i = 0 ∧ type set = 2 then pfi(0,i) else
            begin ps(1,'['); pfi(0,i); ps(2,']') end
      end B;
      procedure B(i,j); integer i,j; if j ≠ 0 then
      begin ps(7,'₂×₂'); ps(0,'b'); ps(1,'['); pfi(0,i); ps(2,']');
            if j ≠ 1 then begin ps(9,'↑'); pfi(0,j) end
      end B;
      procedure theta(i,e1)times B:(j,e2); integer i,e1,j,e2;
      begin ps(3,'('); pfi(100,if no2 = 0 then 1 else (- i1) ↑ no2);
            ps(8,'/'); pfi(100,fact(no2) × fact(e1-no2)); ps(4,')');
            if i < q then begin B(i,e1-no2); B(q,no2) end
            else begin B(q,no2); B(i,e1-no2) end;
            B(j,e2)
      end theta;
      i:= name of vector; field:= 3; left margin1:= left margin;
      if type set = 2 ∨ type set = 4 then left margin:= 6;
```

```
if type set = 2 ∨ type set = 4 then n1:= 1 else n1:= 0;
if i = 0 then for t:= 0 step 1 until e × q do
if E[0,t] ≠ nil then
begin lines(1); spaces(left margin);
      B1(t + 2 × period × q); ps(10,':='); B1(0);
      if index(t) ≠ 0 then begin ps(6,'␣+␣'); B1(index(t)) end;
      if i1 ≠ 0 then begin pi(-i1); B(q,1) end;
      ps(12,';')
end;
if E[0,name of vector] > 0 then
begin lines(2);
      s('comment␣conditions␣on␣E['); pi(i); s('];'); lines(2);
      length1:= E[1,i]; length2:= E[2,i]; length:= E[3,i];
      for der:= 0 step 1 until order - 1 do
      for n:= 0,1,2 do
      begin tmin:= if n = 0 then 0 else if n = 1 then order-der-1 else order;
            tmax:= if n = 0 then order-2-der else if n = 1 then order-1 else order-1+upper;
            for t:= tmin step 1 until tmax do
```

```
begin lines(2); spaces(left margin); field:= 3;
      ps(0,'c'); ps(1,'['); pi(name); ps(11,','); pi(der + n1);
      ps(11,','); pi(t + n1); ps(2,']'); ps(10,':=');lines(1);
      spaces(left margin -1); field:= 4; smin:= if n = 2 then t - order + 1 else 0;
      for no:= 0 step 1 until length1 -1 do
      if v[0,i,no] ≠ nil then
      begin for no1:= smin  step 1 until t do
            for no2:= 0 do
            begin if no2 ≠ 0 ∨ no1 ≠ smin ∨ no ≠ 0 then ps(6,'₂+₂');
                  theta(2×period×q + v[0,i,no],no1)timesB:
                  (E[0,i] + der×length + no×order + (no1-t+order-1),1);
            end
      end; if n = 2 then
      begin for no:= 0 step 1 until length2 - 1 do
            if v[1,i,no] ≠ nil then
            for no2:= 0 do
            begin if length1 = 0 ∧ (no ≠ 0 ∨ no2 ≠ 0) ∨ length1 ≠ 0 then
                  ps(6,'₂+₂'); theta(2×period×q + cond[v[1,i,no]], t-order)
```

```
                                        timesB:(E[0,i] + der×length + length1×order + no, 1)
                        end
                    end; if n ≠ 0 then
                    for no2:= 0 do
                    begin if (length1= 0 ∧ length2 = 0 ∧ no2 ≠ 0) ∨
                          (length1 ≠  0 ∨ length2 ≠ 0) then ps(5,'␣-␣');
                          theta(2×period×q + i, t-order+1+der)timesB:(nil,0)
                    end;
                    if count = left margin -1 then pi(0); ps(12,';')
                end t;
            end n; name:= name + 1
        end; left margin:= left margin1;
end conditons E;
```

```
procedure data;
begin   procedure table (j,length,name); value length; integer j,length,name;
        begin integer procedure store;
              begin i1:= ioi('num_of_atoms'); store:= father(no,B0,no<i1,A,atom(ioi('atom'))) end store;
              for j:= 0 step 1 until length - 1 do
              begin i2:= ioi('num_of_sons');
                    name:= if i2 = 0 then nil else father(no1,B01,no1<i2,A1,store)
              end;
              if iob('data_list_out') then
              begin lines(2); s('_data'); lines(1); s('__generating_tables__');
                    lines(2); B0:= true;
                    for j:= 0 step 1 until length - 1 do list(name,B0); lines(4)
              end
        end table; temp:= left adjust:= false; lines(2); s('data_table_input');
again:  control:= ioi('type_of_table'); lines(1); if control = 0 then
        begin sr('harmonics_of_derivatives');
              for i:= 0 step 1 until im[1] - 1 do table(no2,im[0],D[i,no2])
        end else if control = 1 then
```

```
        begin type:= ioi('type');
              sr('translation table');
              if type = -1 then sr('-,h') else if type = 0 then sr('+,h');
              for i:= 0 step 1 until im[0] - 1 do table(no2,im[0],a[type,i,no2])
        end else if control = 2 then
        begin type:= ioi('type');
              if type = 1 then
              sr('substitution table') else if type = 2 then sr('multiplication table');
              table(no2,im[0],W[type,no2])
        end; if control ≠ -1 then go to again else last data:= last
end data;
```

```
procedure check list(first,last); integer first,last;
begin   integer i,i1,j,k,type,der,no2,field1,col,num col,l0,l1,l2,l3;
        procedure fields(n2,n1); integer n1,n2;
        begin l2:= n2; l1:= n1; num col:= line length ÷ (l1 + l0) end fields;
        l0:= 7; l1:= 16; field1:= field; i1:= first; l3:= chlength('y[i]');
        fields((l1 - l3) + 2,l1); col:= num col -1;
again:  page; for i:= 0 step 1 until col do
        begin spaces(l0 - 2); s('i,'); spaces(l2); s('V[i]'); spaces(l1 - l2 - l3) end; lines(1);
        for i:= i1 step 1 until if col = 0 then last else i1 + 49 do
        begin lines(1); for j:= 0 step 1 until col do
              if i + j×50 < last then
              begin k:= i+ j×50; field:= l0; pi(k); field:= l1; pi(V[k]) end
              else col:= col - 1
        end; if col = num col - 1  then begin i1:= k+1; go to again end;
        page; s('Z[type,der,no3[type],no2]'); lines(2); k:= 1;
        fields(12,12 + chlength('Z[,,,]:='));
        for type:= -1,0,1,2 do
        for i:= 0 step 1 until if type < 1 then e × q else no3[type] do
        for der:= 0 step 1 until order - 1 do
```

```
        for no2:= 0 step 1 until im[0] -1 do
        begin if k ≥ num col then begin lines(1); k:= 1 end;
              field:= 3; s('␣Z['); pi(type); s(','); pi(der); s(',');
              pi(i); s(','); pi(no2); s(']:=␣'); field:= 10;
              if Z[type,der,i,no2] = nil then begin s('␣nil'); spaces(10 - 4) end else
              pi(Z[type,der,i,no2]); k:= k + 1
        end;
        lines(5); s('T[type,i]'); lines(2); k:= 1;
        fields(6,6 + chlength('T[,]:='));
        for i:= 0 step 1 until e × q do
        for type:= -1,0 do
        begin if k ≥ num col then begin lines(1); k:= 1 end;
              field:= 3; s('␣T['); pi(type); s(','); pi(i); s(']:=␣');
              field:= 10; if T[type,i] = nil then begin s('␣nil'); spaces(10 -4) end else
              pi(T[type,i]); k:= k + 1
        end;
        field:= field1
end check list;
```

```
        procedure scheme;
        comment this is a dummy version of scheme;
        begin   page; if print scheme then
                begin s('solution,scheme.,,This,is,a,dummy,scheme,routine.,print,scheme:=,true') end
                else
                begin s('dummy,print,scheme.,,,print,scheme:=,false') end;
                page
        end scheme;

        comment All declarations have been made.  The following section constitutes the control section;
        nil:= -(2↑48); last data:= 0; print scheme:= false; last l:= nil; linear comb:= true;
        inout:= iob('inout'); origin:= 0;
again1: last:= last1:= last data; for i:= -1 step 1 until 2 do no3[i]:= -1; l:= last l;
        for n:= 0 step 1 until e × q do
        begin for type:= -1 step 1 until 3 do
              begin if type < 1 then T[type,n]:= nil; E[type,n]:= nil;
                    if (type > -1 ∧ type < 2) then
                    for no2:= 0 step 1 until height1 - 1 do v[type,n,no2]:= nil;
                    if type ≠ 3 then
```

```
                begin for der:= 0 step 1 until order - 1 do
                      begin for no2:= 0 step 1 until im[0]-1 do Z[type,der,n,no2]:= nil end
                end
           end
      end; for n:= 0 step 1 until height - 1 do cond[n]:= nil;
           for n:= last step 1 until list length do V[n]:= nil;
      if last data = 0 then data; scheme; lines(2); title; lines(2);
again: control:= ioi('control'); if control < -1 then go to fin;
      if control = -1 then begin print scheme:= true; go to again1 end;

      if control = 0 then
      begin mode:= ioi('mode of memory'); mode1:= if mode = 0 then -1 else 0; lines(1);
            l:= last l:= 2×period×q + e×q + 1 + order×im[0]×(if mode= -1 then period×q else e×q + 1);
      end else
      if control = 1 then
      begin  comment  construct a new approximator N;
            sr('new N'); no3[1]:= no3[1] + 1; lines(1); field:= 3;
            if iob('substitution') then
            begin num[0]:= ioi('E[i],i'); type:= 1; tempS:= temp:= false; cond[no3[1]]:= num[0];
                  lines(2); s('N['); pi(no3[1]); s(']:=');
```

```
            s('X(E['); pi(num[0]); s('])');
        end else
        begin type:= 2; num[0]:= ioi('DX(E[i]),␣i');
            num[type]:= ioi('S[j],j'); tempS:= iob('temporary␣sum');
            lines(2); s('N['); pi(no3[1]); s(']:=␣');
            s('DX(E['); pi(num[0]); s(']),␣×␣S['); pi(num[type]); s(']')
        end; temp:= false;
        if E[0,num[0]] = nil then
        begin create E(num[0],true);
            translate(num[0],Z[mode,der,num[0],no2],Z[mode1,der,num[0],no2],num[0]+q,mode1)
        end; father(no2,BO2,no2 < im[0],Z[1,0,no3[1],no2],Ze);
end else
if control = 2 then
begin   comment create a new sum S;
        sr('new␣sum');
        no3[2]:= no3[2] + 1; temp:= iob('temporary␣store'); linear comb:= iob('linear␣combination');
new E:  length:= ioi('length␣of␣sum'); num[0]:= num[1]:= 0;
        for i:= 0 step 1 until length -1 do
        begin type:= ioi('type␣of␣vector'); vs[type,num[type]]:= ioi('num[type]');
            num[type]:= num[type] + 1
```

```
end;
comment begin normalization of input; length2:= 0;
i:= -1; for i:= i + 1 while i < num[1] do if cond[vs[1,i]] ≠ nil then length2:= length2 + 1;
length1:= num[0]; i:= -1;
for i:= i + 1 while i < num[1] do
begin BO:= true; if cond[vs[1,i]] = nil then
      begin j:= i; for j:= j+ 1 while BO ∧ j < num[1] do
            begin BO:= (cond[vs[1,j]] = nil); if ¬ BO then
                  begin i1:= vs[1,i]; vs[1,i]:= vs[1,j]; vs[1,j]:= i1 end
            end
      end
end; BO:= true;
for type:= 0,1,1 do
begin n:= if type = 0 then length1 - 1 else if BO then length2 - 1 else num[1] - 1;
      for i:= if BO then 0 else length2 step 1 until n do
      begin j1:= i; for j:= i step 1 until n do
            begin if j = i then i1:= vs[type,i]; if vs[type,j] < vs[type,i] then
                  begin vs[type,i]:= vs[type,j]; j1:= j end
            end; vs[type,j1]:= i1; if type = 1 then BO:= false
```

```
      end
end of normalization of input;
i:= -1; for i:= i +1 while i < length1 do
begin if E[0,vs[0,i]] = nil then
      begin create E(vs[0,i],true);
            translate(vs[0,i],Z[mode,der,vs[0,i],no2],Z[mode1,der,vs[0,i],no2],vs[0,i]+q,mode1)
      end
end; length:= length + length1 × (order-1);
if control = 3 then
begin for i:= 0 step 1 until length1-1 do v[0,no3[0],i]:= vs[0,i];
      for i:= 0 step 1 until length2-1 do v[1,no3[0],i]:= vs[1,i];
      E[0,no3[0]]:= 1; E[1,no3[0]]:= length1; E[2,no3[0]]:= length2; E[3,no3[0]]:= length
end; left margin:= 0; type:= if control = 3 then 0 else 2;
print sum(no3[type]);
if temp then sum(Zp[der,no2],i,length,if i<length1×order
                  then Z[0,i-(i÷order)×order,vs[0,i÷order],no2]
                  else Z[1,0,vs[1,i-length1×order],no2])
        else sum(Z[type,der,no3[type],no2],i,length,if i<length1×order
                  then Z[0,i-(i÷order)×order,vs[0,i÷order],no2]
```

```
                    else Z[1,0,vs[1,i-length1×order],no2]);
        if control = 3 then go to again
end else
if control = 3 then
begin   comment construct a new approximation E[i];
        no3[0]:= ioi('E[i],i'); temp:= false; linear comb:= true;
        create E(no3[0],false); sr('new E'); go to new E
end else
if control = 4 then
begin   comment print all parameter defining equations;
        lines(2); name:= 1;
        s('E[0]:= E(0) + error terms. All defining equations are printed below'); lines(2);
        s('c[i,k,n], where k is in(0,...,order-1) and n is in (0,...,im[0]-1)'); lines(1);
        s('comment equations arising from requiring'); lines(1);
        s('E[...]:= u[...](...) + sum(i,0,im[0]-1,b[...]×A(...)[i]);');
        lines(2); for i:= 0 step 1 until e × q do conditions E(i,name); lines(2);
        s('comment Equations arising from E[0,k] - E(0)[k]:= O(h↑(k+'); pi(upper+im[1]+1); s('));');
        print list(i,1,name,Z[0,der,0,no2],i); lines(1);
        i:= -1; for i:= i + 1 while i ≤ period × q do
        begin BO:= true; for n:= i step q × period until e × q do
```

```
      begin if BO ∧ E[-1,n] ≠ nil then
            begin s('comment Equations,which,define,the,undetermined,parameters'); lines(1);
                  s('used,in,the,expansion,of,E['); pi(n); s('];'); lines(1);
                  print list(j1,2,name,
                        Z[if j1 = 0 then -1 else 0,der,if j1 = 0 then n else i,no2],(-1)↑(j1+1));
                  if mode = -1 then BO:= false
            end; if n = e × q then BO:= false
      end
end; lines(2);
comment points that are to be equated;
n:= ioi('number');
if n > 0 then
begin for i:= 0 step 1 until n - 1 do
      begin vs[0,i]:= ioi('name'); vs[1,i]:= ioi('type') end; lines(2);
      s('comment,the,following,approximations,are,equal'); lines(1); field:= 3;
      for i:= 0 step 2 until n - 2 do
      for j:= 0,1 do
      begin s('E['); pi(vs[1,i+j]); s(','); pi(vs[0,i+j]);
            if j = 0 then s(']:=') else begin s(']'); lines(1) end
      end; s(';');
```

```
      for i:= 0 step 2 until n - 2 do
      print list(j,2,name,Z[vs[1,i+j],der,vs[0,i+j],no2],(-1)↑j)
end; lines(4);
s('Define the undetermined parameters b[...] by printing out '); lines(1);
s('the expansions of E[i,k], i in M:=(1,...,exq), k in P:='); lines(1);
s('(0,...,order-1), with respect to these parameters, if mode'); lines(1);
s('=0 then their local origins are the point 0 else'); lines(1);
s('if mode=-1 then they are -i1×h where i1:=i+q');lines(2);
field:= 4;
for i:= 0 step 1 until e × q do if E[-1,i] ≠ nil then
begin for der:= 0 step 1 until order - 1 do
      begin n:= index(i); s('E['); pi(i); s(','); pi(der); s(']:=u[');
            if mode = 0 ∨ i1 = 0 then pi(0) else begin pi(-i1); s('×h') end; s(']([');
            if n = 0 then pi(0) else begin s('B['); pi(n); s(']') end;
            if mode = 0 ∧ i1 ≠ 0 then begin pi(-i1); s('×h') end;
            s(')[']; pi(der); s(']+sum(i,0,'); pi(im[0] - 1); s(',');
            s('b['); pi(2×period×q + ex q + 1 + (if mode= -1 then j+j1 else i)×im[0]
                        ×order + der×im[0]);
            s('+i]');
```

```
                        s('×A('); pi(if mode = 0 then 0 else -i1);
                        if mode ≠ 0 then s('␣×␣h'); s(')[i]'); lines(1)
                    end; lines(1)
                end;
        end of control; go to again;

fin:    lines(4); field:= 6; s('comment');
        s('last:='); pi(last); s('␣␣last1:='); pi(last1); s('␣␣temp0-last:='); pi(temp0-last);
        s('␣␣list␣length-temp0:='); pi(list length-temp0); lines(1); s('next␣free␣parameter␣b['); pi(1); s('];');
        if iob('check␣list') then check list(0,last);
        go to if control = -2 then define problem else if control = -3 then begin else end of computations
end  of the program; lines(2);
end of computations: lines(2); s('␣end␣of␣computations.')
end
end
```

```
integer procedure atom(i); value i; integer i;
comment unpacked;
if abs(i) > 2↑48 then
begin lines(1); s('atom␣too␣large␣:='); pfi(0,i);
      check list(0,last); go to end of computations
end else
begin atom:= last; if i ≠ nil then begin check; V[last]:= i; last:= last + 1 end;
      debug(6,i,nil,nil)
end atom;

integer procedure get atom(father entry,atomic set,atom,B);
integer father entry,atomic set,atom; Boolean B;
comment unpacked;
begin   if B then
        begin if father entry = nil then begin get atom:= atom:= nil; go to end end;
              B:= false; atomic set:= father entry + 1
        end else atomic set:= atomic set + 1;
        get atom:= atom:= if atomic set < V[father entry] then V[atomic set] else nil;
end:    debug(4,father entry, atomic set,atom)
end get atom;
```

RESTRICTIONS AND OMISSIONS TO RKMI

1. The procedure title does not handle line overflow as it should when the line length of the input channel is different from the line length of the output channel.

2. The procedure Bncollection has a limit of 20 as the number of lists it can handle. This limit can be altered by changing the upper bounds of the own variables B11and B01.

3. The procedures normalize, debug, and scheme are presently "dummy" procedures. We have not needed normalize;* debug has been used on the IBM 7094 version, but for the CDC 6400 version, the use of an update program has proved more useful; the procedure scheme exits and can be easily implemented.

4. We have not yet completely checked out the use of the program when using DX. Various parts have been individually checked. Also, we have a slight problem with interpreting the ξ, η, S as presently printed when using the derivative DX. Refer to Definition 10, Chapter III. The program actually operates with this definition. When J_{NL} is never used, it is correct to replace all

$$\xi_i - u(\theta_i) \quad \rightarrow \xi_i$$

$$\eta_i = X(\xi_i) - R_i \rightarrow \eta_i$$

since Condition A (Theorem 10, Chapter III) always holds. RKMI presently does this. If J_{NL} is used, then we must presently interpret its printed output as

$$\xi_i \rightarrow \xi_i - u(\theta_i)$$

$$\eta_i \rightarrow X(\xi_i) - R_i$$

*See the description of the procedure JPn on page 284.

in order that the correct scheme will be written down. This is simply an oversight in printing the output definitions of ξ, η, and S. It does not affect the parameter equations.

5. Note that procedure Ze in reality handles correctly only lists of the form (numerator, denominator), (exponent), (der_1, n_1, ..., der_{max_1}, n_{max_1}) and, thus, will not correctly carry out a multiplication DX · J for a term such as ψ_{61} which needs another son to correctly represent the multiplication process. The short table presented here (Table VI, Appendix I) does not need more than three sons. Substitution only requires three sons and, thus, the oversight which is rather easily corrected.

Appendix III

RESULTS OF THE EXAMPLES OF CHAPTER VII

Appendix III is devoted entirely to the presentation of the results of the examples treated in Chapter VII. Tables I - III deal with the classical RK4 example. The remaining tables and graphs pertain to the examples 1 - 4 presented in that chapter. A full description of these results is given in Chapter VII.

TABLE I DATA INPUT TO RKM1 EXAMPLE RK4

DATA	COMMENT
1,	Control
1,1,1,1,1,3,3,3,1,2,1,1,0, 2,	FORTRAN output
50,70,6000,7000,10,10,10,	Set program parameters
1,1,4,1,1,16,4,	Particularize problem
'FALSE',	No input-output
	Data table input
0,	Derivative harmonics
1,2,1,1, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 'FALSE'	D^2x See Table VI
0, 1,2,1,1, 1,2,1,1, 0,0,0,0,0,0,0,0,0,0,0,0,0, 'FALSE'	D^3x Appendix I
0,0,0, 1,2,1,1, 1,2,3,1, 1,2,1,1, 1,2,1,1, 0,0,0,0,0,0,0,0,0, 'FALSE'	D^4x
0,0,0,0,0,0,0, 1,2,1,1, 1,2,6,1, 1,2,4,1, 1,2,4,1, 1,2,1,1, 1,2,3,1, 1,2,1,1, 1,2,1,1, 1,2,3,1, 'FALSE'	D^5x
2,1,	Substitution harmonics
2,2,1,1,1,1, 2,2,1,2,1,2, 3,2,1,1,1,0,2,0,0, 2,2,1,6,1,3,	0 ψ_i See Table VI 1 2 Appendix I 3

TABLE I CONTINUED

DATA	COMMENT
3,2,1,1,1,1,2,0,0, 3,2,1,1,1,0,2,0,1, 3,2,1,1,1,0,2,0,2, 2,2,1,24,1,4, 3,2,1,2,1,2,2,0,0, 3,2,1,1,1,1,2,0,1, 3,2,1,1,1,1,2,0,2, 3,2,1,1,1,0,2,0,3, 3,2,1,1,1,0,2,0,4, 3,2,1,1,1,0,2,0,5, 3,2,1,1,1,0,2,0,6, 4,2,1,2,1,0,2,0,0,2,0,0, 'FALSE'	4 ψ_i Substitution 5 harmonics 6 continued 8 9 10 11 12 13 14 15 21
1,0,	Translation harmonics
1,2,1,1, 1,4,-1,1,10,1, 1,4,-1,1,10,1, 1,4,1,2,10,2, 1,4,3,2,10,2, 1,4,1,2,10,2, 1,4,1,2,10,2, 1,4,-1,6,10,3, 1,4,-1,1,10,3, 1,4,-2,3,10,3, 1,4,-2,3,10,3, 1,4,-1,6,10,3, 1,4,-1,2,10,3, 1,4,-1,6,10,3, 1,4,-1,6,10,3, 1,4,-1,2,10,3, 'FALSE'	A_o See Table VII Appendix I
0, 1,2,1,1, 0, 1,4,-1,1,10,1, 1,4,-2,1,10,1, 0, 0, 1,4,1,2,10,2, 1,4,5,2,10,2, 1,4,1,1,10,2, 1,4,1,1,10,2, 0, 0, 0, 0, 1,4,1,1,10,2, 'FALSE'	A_1

TABLE I CONTINUED

DATA	COMMENT
0,0, 1,2,1,1, 0, 1,4,-1,1,10,1, 1,4,-1,1,10,1, 1,4,-1,1,10,1, 0, 1,4,1,2,10,2, 1,4,1,1,10,2, 1,4,1,1,10,2, 1,4,1,2,10,2, 1,4,3,2,10,2, 1,4,1,2,10,2, 1,4,1,2,10,2, 1,4,1,2,10,2, 'FALSE'	A_2
0,0,0, 1,2,1,1, 0, 0, 0, 1,4,-1,1,10,1, 1,4,-3,1,10,1, 0, 0, 0, 0, 0, 0, 'FALSE'	A_3
0,0,0,0, 1,2,1,1, 0,0, 0, 1,4,-1,1,10,1, 1,4,-1,1,10,1, 1,4,-1,1,10,1, 0, 0, 0, 0, 1,4,-1,1,10,1, 'FALSE'	A_4
0,0,0,0,0, 1,2,1,1, 0, 0, 0, 1,4,-1,1,10,1, 0,	A_5

TABLE I CONTINUED

DATA	COMMENT
1,4,-1,1,10,1, 1,4,-2,1,10,1, 0, 0, 0, 'FALSE'	A_5
0,0,0,0,0,0, 1,2,1,1, 0, 0, 0, 1,4,-1,1,10,1, 0, 1,4,-1,1,10,1, 1,4,-1,1,10,1, 1,4,-1,1,10,1, 0, 'FALSE'	A_6
0,0,0,0,0,0,0, 1,2,1,1, 0,0,0,0,0,0,0,0, 'FALSE'	A_8
0,0,0,0,0,0,0,0, 1,2,1,1, 0,0,0,0,0,0,0, 'FALSE'	A_9
0,0,0,0,0,0,0,0,0, 1,2,1,1, 0,0,0,0,0,0, 'FALSE'	A_{10}
0,0,0,0,0,0,0,0,0,0, 1,2,1,1, 0,0,0,0,0, 'FALSE'	A_{11}
0,0,0,0,0,0,0,0,0,0,0, 1,2,1,1, 0,0,0,0, 'FALSE'	A_{12}
0,0,0,0,0,0,0,0,0,0,0,0, 1,2,1,1, 0,0,0, 'FALSE'	A_{13}
0,0,0,0,0,0,0,0,0,0,0,0,0, 1,2,1,1, 0,0, 'FALSE'	A_{14}
0,0,0,0,0,0,0,0,0,0,0,0,0,0, 1,2,1,1, 0, 'FALSE'	A_{15}

TABLE I CONTINUED

DATA	COMMENT
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 1,2,1,1, 'FALSE'	A_{21}
-1	End procedure data output
COMMENT THIS IS A CLASSICAL RUNGE KUTTA METHOD OF RANK 4 FOR A SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS.,	
0,-1,	Forward translation
1,'TRUE',4, 1,'TRUE',3, 1,'TRUE',2, 1,'TRUE',1. 3,3,2, 0,4,1,0, 3,2,3, 0,4,1,0,1,1, 3,1,4, 0,4,1,0,1,1,1,2, 3,0,5, 0,4,1,0,1,1,1,2,1,3,	Scheme
4,-1,	Print all parameter equations
-4,'FALSE',	Exit from program

```
TABLE II   RKM1 OUTPUT

COMMENT  GIVEN THE DATA INPUT OF TABLE 1, THE BELOW OUTPUT IS GENERATED
         BY RKM1 SOURCE LISTING 8090 68, 9/20/68.

     CONTROL .=       1

 DATA INPUT THAT SETS COMPUTER VARIABLES

I.=0., 'FOR' PRINT LENGTH(/I/).=  1,  1,  1,  1,  1,  3,  3,  3,  1,  2,
  1,  1,  0, 'DO' I.= I+1.,
     TYPE SET .=       2
     N FOR PRINT .=      50
     LINE LENGTH .=      70
     ORIGIN OF TEMPORARY STORE .=    6000
     LIST LENGTH .=    7000
     MAXIMUM NUMBER OF VECTORS  N OR SUMS  S .=      10
     MAXIMUM LENGTH OF A SUM .=      10
     MAXIMUM LENGTH OF A LIST PRODUCT .=      10

DATA INPUT THAT PARTICULARIZES THE PROBLEM


     ORDER OF THE DIFFERENTIAL EQUATION .=       1
      D'POWER'(ORDER+UPPER)X IS THE FIRST NEGLECTED  TAYLOR TERM, UPPER
 =      1
     NUMBER OF POINTS IN ONE H INTERVAL .=       4
     NUMBER OF H INTERVALS .=       1
     PERIOD OF SCHEME .=       1
     NUMBER OF BASIC FUNCTIONS .=      16
     NUMBER OF DERIVATIVES WE ATTEMPT TO MATCH .=       4
                                    INOUT .= 'FALSE'

DATA TABLE INPUT


DUMMY PRINT SCHEME.  PRINT SCHEME.= 'FALSE'


'COMMENT'  THIS IS A CLASSICAL RUNGE KUTTA METHOD OF RANK 4 FOR A
           SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS.,




 N(/  0/).=  X( E(/  4/))

 N(/  1/).=  X( E(/  3/))

 N(/  2/).=  X( E(/  2/))

 N(/  3/).=  X( E(/  1/))


 E(/  3,  0/).=
SUM(I,0,  0,SUM(J,0,  0,B(/  77 + I *  1 + J/) *  E(/V(/  0,I/), J/)))+
SUM(I,0,  0,B(/  78 + I/) *  N(/V(/  1,I/)/))


WHERE  V(/   0/).=(   4)  AND   V(/   0,I/).= V(/   0/)(/I/)
```

```
WHERE  V(/   1/).=(   0)  AND   V(/   1,I/).= V(/   1/)(/I/)


 E(/  2,  0/).=
SUM(I,0,  0,SUM(J,0,  0,B(/  79 + I *  1 + J/) *  E(/V(/  0,I/), J/)))+
SUM(I,0,  1,B(/  80 + I/) *  N(/V(/  1,I/)/))


WHERE  V(/   0/).=(   4)  AND   V(/   0,I/).= V(/   0/)(/I/)
WHERE  V(/   1/).=(   0,   1)  AND   V(/   1,I/).= V(/   1/)(/I/)


 E(/  1,  0/).=
SUM(I,0,  0,SUM(J,0,  0,B(/  82 + I *  1 + J/) *  E(/V(/  0,I/), J/)))+
SUM(I,0,  2,B(/  83 + I/) *  N(/V(/  1,I/)/))


WHERE  V(/   0/).=(   4)  AND   V(/   0,I/).= V(/   0/)(/I/)
WHERE  V(/   1/).=(   0,   1,   2)  AND   V(/   1,I/).= V(/   1/)(/I/)


 E(/  0,  0/).=
SUM(I,0,  0,SUM(J,0,  0,B(/  86 + I *  1 + J/) *  E(/V(/  0,I/), J/)))+
SUM(I,0,  3,B(/  87 + I/) *  N(/V(/  1,I/)/))


WHERE  V(/   0/).=(   4)  AND   V(/   0,I/).= V(/   0/)(/I/)
WHERE  V(/   1/).=(   0,   1,   2,   3)  AND   V(/   1,I/).= V(/   1/)(/
I/)


 E(/0/).=  E(0) + ERROR TERMS.   ALL DEFINING EQUATIONS ARE PRINTED BELO
W


C(/I,K,N/), WHERE K IS IN (0,...,ORDER-1) AND N IS IN (0,..., IM(/0/)-1)
'COMMENT' EQUATIONS ARISING FROM REQUIRING
 E(/.../).= U(/.../) (...) + SUM(I,0,IM(/0/)-1, B(/.../)* A(/.../)(/I/))
..


      B( 8)=B 0
      B( 9)=B 0 + B( 1)
      B( 10)=B 0 + B( 2)
      B( 11)=B 0 + B( 3)
      B( 12)=B 0 -1 * B( 4)


'COMMENT' CONDITIONS ON  E(/  0/).,


      C(  1,  1,  1)=
     .( 1./ 1.) * B( 86) - ( 1./ 1.)

      C(  1,  1,  2)=
     .( 1./ 1.) * B( 12) * B( 86) + ( 1./ 1.) * B( 87) + ( 1./ 1.)
     . * B( 88) + ( 1./ 1.) * B( 89) + ( 1./ 1.) * B( 90) - ( 1./
     . 1.) * B( 8)


'COMMENT' CONDITIONS ON  E(/  1/).,
```

```
      C(  2,  1,  1)=
     .( 1./ 1.) * B( 82) - ( 1./ 1.)

      C(  2,  1,  2)=
     .( 1./ 1.) * B( 12) * B( 82) + ( 1./ 1.) * B( 83) + ( 1./ 1.)
     . * B( 84) + ( 1./ 1.) * B( 85) - ( 1./ 1.) * B( 9)

'COMMENT' CONDITIONS ON  E(/  2/).,

      C(  3,  1,  1)=
     .( 1./ 1.) * B( 79) - ( 1./ 1.)

      C(  3,  1,  2)=
     .( 1./ 1.) * B( 12) * B( 79) + ( 1./ 1.) * B( 80) + ( 1./ 1.)
     . * B( 81) - ( 1./ 1.) * B( 10)

'COMMENT' CONDITIONS ON  E(/  3/).,

      C(  4,  1,  1)=
     .( 1./ 1.) * B( 77) - ( 1./ 1.)

      C(  4,  1,  2)=
     .( 1./ 1.) * B( 12) * B( 77) + ( 1./ 1.) * B( 78) - ( 1./ 1.)
     . * B( 11)

'COMMENT' EQUATIONS ARISING FROM  E(/0,K/) -  E(0)(/K/).=  O(H*POWER*(K
+  6)).,

      C(  5,  1,  1)=
     . + ( 1./ 2.) * B( 12)** 2 * B( 86)
     . + ( 1./ 1.) * B( 12) * B( 87) + ( 1./ 1.) * B( 11) * B( 88)
     . + ( 1./ 1.) * B( 10) * B( 89) + ( 1./ 1.) * B( 9) * B( 90)

      C(  5,  1,  2)=
     . + ( 1./ 6.) * B( 12)** 3 * B( 86)
     . + ( 1./ 2.) * B( 12)** 2 * B( 87) + ( 1./ 2.) * B( 11)** 2
     . * B( 88) + ( 1./ 2.) * B( 10)** 2 * B( 89) + ( 1./ 2.) * B(
     . 9)** 2 * B( 90)

      C(  5,  1,  3)=
     . + ( 1./ 6.) * B( 12)** 3 * B( 86)
     . + ( 1./ 2.) * B( 12)** 2 * B( 87) + ( 1./ 1.) * B( 61) * B(
     . 88) + ( 1./ 1.) * B( 45) * B( 89) + ( 1./ 1.) * B( 29) * B(
     . 90)

      C(  5,  1,  4)=
     . + ( 1./ 24.) * B( 12)** 4 * B( 86)
     . + ( 1./ 6.) * B( 12)** 3 * B( 87) + ( 1./ 6.) * B( 11)** 3
     . * B( 88) + ( 1./ 6.) * B( 10)** 3 * B( 89) + ( 1./ 6.) * B(
     . 9)** 3 * B( 90)

      C(  5,  1,  5)=
     . + ( 3./ 24.) * B( 12)** 4 * B( 86)
     . + ( 1./ 2.) * B( 12)** 3 * B( 87) + ( 1./ 1.) * B( 11) * B(
     . 61) * B( 88) + ( 1./ 1.) * B( 10) * B( 45) * B( 89) + ( 1./
     . 1.) * B( 9) * B( 29) * B( 90)
```

```
 C(  5,  1,  6)=
. + ( 1./ 24.) * B( 12)** 4 * B( 86)
. + ( 1./ 6.) * B( 12)** 3 * B( 87) + ( 1./ 1.) * B( 62) * B(
. 88) + ( 1./ 1.) * B( 46) * B( 89) + ( 1./ 1.) * B( 30) * B(
. 90)

 C(  5,  1,  7)=
. + ( 1./ 24.) * B( 12)** 4 * B( 86)
. + ( 1./ 6.) * B( 12)** 3 * B( 87) + ( 1./ 1.) * B( 63) * B(
. 88) + ( 1./ 1.) * B( 47) * B( 89) + ( 1./ 1.) * B( 31) * B(
. 90)

 C(  5,  1,  8)=
. + ( 1./ 120.) * B( 12)** 5 * B( 86)
. + ( 1./ 24.) * B( 12)** 4 * B( 87) + ( 1./ 24.) * B( 11)**
. 4 * B( 88) + ( 1./ 24.) * B( 10)** 4 * B( 89) + ( 1./ 24.)
. * B( 9)** 4 * B( 90)

 C(  5,  1,  9)=
. + ( 6./ 120.) * B( 12)** 5 * B( 86)
. + ( 1./ 4.) * B( 12)** 4 * B( 87) + ( 1./ 2.) * B( 11)** 2
. * B( 61) * B( 88) + ( 1./ 2.) * B( 10)** 2 * B( 45) * B(
. 89) + ( 1./ 2.) * B( 9)** 2 * B( 29) * B( 90)

 C(  5,  1, 10)=
. + ( 4./ 120.) * B( 12)** 5 * B( 86)
. + ( 1./ 6.) * B( 12)** 4 * B( 87) + ( 1./ 1.) * B( 11) * B(
. 62) * B( 88) + ( 1./ 1.) * B( 10) * B( 46) * B( 89) + ( 1./
. 1.) * B( 9) * B( 30) * B( 90)

 C(  5,  1, 11)=
. + ( 4./ 120.) * B( 12)** 5 * B( 86)
. + ( 1./ 6.) * B( 12)** 4 * B( 87) + ( 1./ 1.) * B( 11) * B(
. 63) * B( 88) + ( 1./ 1.) * B( 10) * B( 47) * B( 89) + ( 1./
. 1.) * B( 9) * B( 31) * B( 90)

 C(  5,  1, 12)=
. + ( 1./ 120.) * B( 12)** 5 * B( 86)
. + ( 1./ 24.) * B( 12)** 4 * B( 87) + ( 1./ 1.) * B( 64) * B(
. 88) + ( 1./ 1.) * B( 48) * B( 89) + ( 1./ 1.) * B( 32) * B(
. 90)

 C(  5,  1, 13)=
. + ( 3./ 120.) * B( 12)** 5 * B( 86)
. + ( 3./ 24.) * B( 12)** 4 * B( 87) + ( 1./ 1.) * B( 65) * B(
. 88) + ( 1./ 1.) * B( 49) * B( 89) + ( 1./ 1.) * B( 33) * B(
. 90)

 C(  5,  1, 14)=
. + ( 1./ 120.) * B( 12)** 5 * B( 86)
. + ( 1./ 24.) * B( 12)** 4 * B( 87) + ( 1./ 1.) * B( 66) * B(
. 88) + ( 1./ 1.) * B( 50) * B( 89) + ( 1./ 1.) * B( 34) * B(
. 90)

 C(  5,  1, 15)=
. + ( 1./ 120.) * B( 12)** 5 * B( 86)
. + ( 1./ 24.) * B( 12)** 4 * B( 87) + ( 1./ 1.) * B( 67) * B(
. 88) + ( 1./ 1.) * B( 51) * B( 89) + ( 1./ 1.) * B( 35) * B(
. 90)

 C(  5,  1, 16)=
```

```
     . + ( 3./ 120.) * B( 12)** 5 * B( 86)
     . + ( 1./ 8.) * B( 12)** 4 * B( 87) + ( 1./ 2.) * B( 61)** 2
     . * B( 88) + ( 1./ 2.) * B( 45)** 2 * B( 89) + ( 1./ 2.) * B(
     . 29)** 2 * B( 90)


*COMMENT*  EQUATIONS WHICH DEFINE THE UNDETERMINED PARAMETERS
USED IN THE EXPANSION OF  E(/   1/).,

      C(  6,  1,  1)=
     . - ( 1./ 1.) * B( 29)
     . + ( 1./ 2.) * B( 12)** 2 * B( 82) + ( 1./ 1.) * B( 12) * B(
     . 83) + ( 1./ 1.) * B( 11) * B( 84) + ( 1./ 1.) * B( 10) * B(
     . 85)

      C(  6,  1,  2)=
     . - ( 1./ 1.) * B( 30)
     . + ( 1./ 6.) * B( 12)** 3 * B( 82) + ( 1./ 2.) * B( 12)** 2
     . * B( 83) + ( 1./ 2.) * B( 11)** 2 * B( 84) + ( 1./ 2.) * B(
     . 10)** 2 * B( 85)

      C(  6,  1,  3)=
     . - ( 1./ 1.) * B( 31)
     . + ( 1./ 6.) * B( 12)** 3 * B( 82) + ( 1./ 2.) * B( 12)** 2
     . * B( 83) + ( 1./ 1.) * B( 61) * B( 84) + ( 1./ 1.) * B( 45)
     . * B( 85)

      C(  6,  1,  4)=
     . - ( 1./ 1.) * B( 32)
     . + ( 1./ 24.) * B( 12)** 4 * B( 82) + ( 1./ 6.) * B( 12)**
     . 3 * B( 83) + ( 1./ 6.) * B( 11)** 3 * B( 84) + ( 1./ 6.) * B.
     .( 10)** 3 * B( 85)

      C(  6,  1,  5)=
     . - ( 1./ 1.) * B( 33)
     . + ( 3./ 24.) * B( 12)** 4 * B( 82) + ( 1./ 2.) * B( 12)**
     . 3 * B( 83) + ( 1./ 1.) * B( 11) * B( 61) * B( 84) + ( 1./
     . 1.) * B( 10) * B( 45) * B( 85)

      C(  6,  1,  6)=
     . - ( 1./ 1.) * B( 34)
     . + ( 1./ 24.) * B( 12)** 4 * B( 82) + ( 1./ 6.) * B( 12)**
     . 3 * B( 83) + ( 1./ 1.) * B( 62) * B( 84) + ( 1./ 1.) * B(
     . 46) * B( 85)

      C(  6,  1,  7)=
     . - ( 1./ 1.) * B( 35)
     . + ( 1./ 24.) * B( 12)** 4 * B( 82) + ( 1./ 6.) * B( 12)**
     . 3 * B( 83) + ( 1./ 1.) * B( 63) * B( 84) + ( 1./ 1.) * B(
     . 47) * B( 85)

      C(  6,  1,  8)=
     . - ( 1./ 1.) * B( 36)
     . + ( 1./ 120.) * B( 12)** 5 * B( 82) + ( 1./ 24.) * B( 12)**
     . 4 * B( 83) + ( 1./ 24.) * B( 11)** 4 * B( 84) + ( 1./ 24.)
     . * B( 10)** 4 * B( 85)

      C(  6,  1,  9)=
     . - ( 1./ 1.) * B( 37)
     . + ( 6./ 120.) * B( 12)** 5 * B( 82) + ( 1./ 4.) * B( 12)**
```

```
     . 4 * B( 83) + ( 1./ 2.) * B( 11)** 2 * B( 61) * B( 84) + (
     . 1./ 2.) * B( 10)** 2 * B( 45) * B( 85)

      C(  6,  1, 10)=
     . - ( 1./ 1.) * B( 38)
     . + ( 4./ 120.) * B( 12)** 5 * B( 82) + ( 1./ 6.) * B( 12)**
     . 4 * B( 83) + ( 1./ 1.) * B( 11) * B( 62) * B( 84) + ( 1./
     . 1.) * B( 10) * B( 46) * B( 85)

      C(  6,  1, 11)=
     . - ( 1./ 1.) * B( 39)
     . + ( 4./ 120.) * B( 12)** 5 * B( 82) + ( 1./ 6.) * B( 12)**
     . 4 * B( 83) + ( 1./ 1.) * B( 11) * B( 63) * B( 84) + ( 1./
     . 1.) * B( 10) * B( 47) * B( 85)

      C(  6,  1, 12)=
     . - ( 1./ 1.) * B( 40)
     . + ( 1./ 120.) * B( 12)** 5 * B( 82) + ( 1./ 24.) * B( 12)**
     . 4 * B( 83) + ( 1./ 1.) * B( 64) * B( 84) + ( 1./ 1.) * B(
     . 48) * B( 85)

      C(  6,  1, 13)=
     . - ( 1./ 1.) * B( 41)
     . + ( 3./ 120.) * B( 12)** 5 * B( 82) + ( 3./ 24.) * B( 12)**
     . 4 * B( 83) + ( 1./ 1.) * B( 65) * B( 84) + ( 1./ 1.) * B(
     . 49) * B( 85)

      C(  6,  1, 14)=
     . - ( 1./ 1.) * B( 42)
     . + ( 1./ 120.) * B( 12)** 5 * B( 82) + ( 1./ 24.) * B( 12)**
     . 4 * B( 83) + ( 1./ 1.) * B( 66) * B( 84) + ( 1./ 1.) * B(
     . 50) * B( 85)

      C(  6,  1, 15)=
     . - ( 1./ 1.) * B( 43)
     . + ( 1./ 120.) * B( 12)** 5 * B( 82) + ( 1./ 24.) * B( 12)**
     . 4 * B( 83) + ( 1./ 1.) * B( 67) * B( 84) + ( 1./ 1.) * B(
     . 51) * B( 85)

      C(  6,  1, 16)=
     . - ( 1./ 1.) * B( 44)
     . + ( 3./ 120.) * B( 12)** 5 * B( 82) + ( 1./ 8.) * B( 12)**
     . 4 * B( 83) + ( 1./ 2.) * B( 61)** 2 * B( 84) + ( 1./ 2.) * B
     .( 45)** 2 * B( 85)

'COMMENT'  EQUATIONS WHICH DEFINE THE UNDETERMINED PARAMETERS
USED IN THE EXPANSION OF  E(/   2/).,

      C(  7,  1,  1)=
     . - ( 1./ 1.) * B( 45)
     . + ( 1./ 2.) * B( 12)** 2 * B( 79) + ( 1./ 1.) * B( 12) * B(
     . 80) + ( 1./ 1.) * B( 11) * B( 81)

      C(  7,  1,  2)=
     . - ( 1./ 1.) * B( 46)
     . + ( 1./ 6.) * B( 12)** 3 * B( 79) + ( 1./ 2.) * B( 12)** 2
     . * B( 80) + ( 1./ 2.) * B( 11)** 2 * B( 81)

      C(  7,  1,  3)=
     . - ( 1./ 1.) * B( 47)
```

```
     . + ( 1./ 6.) * B( 12)** 3 * B( 79) + ( 1./ 2.) * B( 12)** 2
     . * B( 80) + ( 1./ 1.) * B( 61) * B( 81)

      C(  7,  1,  4)=
     . - ( 1./ 1.) * B( 48)
     . + ( 1./ 24.) * B( 12)** 4 * B( 79) + ( 1./ 6.) * B( 12)**
     . 3 * B( 80) + ( 1./ 6.) * B( 11)** 3 * B( 81)

      C(  7,  1,  5)=
     . - ( 1./ 1.) * B( 49)
     . + ( 3./ 24.) * B( 12)** 4 * B( 79) + ( 1./ 2.) * B( 12)**
     . 3 * B( 80) + ( 1./ 1.) * B( 11) * B( 61) * B( 81)

      C(  7,  1,  6)=
     . - ( 1./ 1.) * B( 50)
     . + ( 1./ 24.) * B( 12)** 4 * B( 79) + ( 1./ 6.) * B( 12)**
     . 3 * B( 80) + ( 1./ 1.) * B( 62) * B( 81)

      C(  7,  1,  7)=
     . - ( 1./ 1.) * B( 51)
     . + ( 1./ 24.) * B( 12)** 4 * B( 79) + ( 1./ 6.) * B( 12)**
     . 3 * B( 80) + ( 1./ 1.) * B( 63) * B( 81)

      C(  7,  1,  8)=
     . - ( 1./ 1.) * B( 52)
     . + ( 1./ 120.) * B( 12)** 5 * B( 79) + ( 1./ 24.) * B( 12)**
     . 4 * B( 80) + ( 1./ 24.) * B( 11)** 4 * B( 81)

      C(  7,  1,  9)=
     . - ( 1./ 1.) * B( 53)
     . + ( 6./ 120.) * B( 12)** 5 * B( 79) + ( 1./ 4.) * B( 12)**
     . 4 * B( 80) + ( 1./ 2.) * B( 11)** 2 * B( 61) * B( 81)

      C(  7,  1, 10)=
     . - ( 1./ 1.) * B( 54)
     . + ( 4./ 120.) * B( 12)** 5 * B( 79) + ( 1./ 6.) * B( 12)**
     . 4 * B( 80) + ( 1./ 1.) * B( 11) * B( 62) * B( 81)

      C(  7,  1, 11)=
     . - ( 1./ 1.) * B( 55)
     . + ( 4./ 120.) * B( 12)** 5 * B( 79) + ( 1./ 6.) * B( 12)**
     . 4 * B( 80) + ( 1./ 1.) * B( 11) * B( 63) * B( 81)

      C(  7,  1, 12)=
     . - ( 1./ 1.) * B( 56)
     . + ( 1./ 120.) * B( 12)** 5 * B( 79) + ( 1./ 24.) * B( 12)**
     . 4 * B( 80) + ( 1./ 1.) * B( 64) * B( 81)

      C(  7,  1, 13)=
     . - ( 1./ 1.) * B( 57)
     . + ( 3./ 120.) * B( 12)** 5 * B( 79) + ( 3./ 24.) * B( 12)**
     . 4 * B( 80) + ( 1./ 1.) * B( 65) * B( 81)

      C(  7,  1, 14)=
     . - ( 1./ 1.) * B( 58)
     . + ( 1./ 120.) * B( 12)** 5 * B( 79) + ( 1./ 24.) * B( 12)**
     . 4 * B( 80) + ( 1./ 1.) * B( 66) * B( 81)

      C(  7,  1, 15)=
     . - ( 1./ 1.) * B( 59)
     . + ( 1./ 120.) * B( 12)** 5 * B( 79) + ( 1./ 24.) * B( 12)**
     . 4 * B( 80) + ( 1./ 1.) * B( 67) * B( 81)
```

```
      C(  7,  1, 16)=
     . - ( 1./ 1.) * B( 60)
     . + ( 3./ 120.) * B( 12)** 5 * B( 79) + ( 1./ 8.) * B( 12)**
     . 4 * B( 80) + ( 1./ 2.) * B( 61)** 2 * B( 81)

'COMMENT'  EQUATIONS WHICH DEFINE THE UNDETERMINED PARAMETERS
USED IN THE EXPANSION OF  E(/  3/).,

      C(  8,  1,  1)=
     . - ( 1./ 1.) * B( 61)
     . + ( 1./ 2.) * B( 12)** 2 * B( 77) + ( 1./ 1.) * B( 12) * B(
     . 78)

      C(  8,  1,  2)=
     . - ( 1./ 1.) * B( 62)
     . + ( 1./ 6.) * B( 12)** 3 * B( 77) + ( 1./ 2.) * B( 12)** 2
     . * B( 78)

      C(  8,  1,  3)=
     . - ( 1./ 1.) * B( 63)
     . + ( 1./ 6.) * B( 12)** 3 * B( 77) + ( 1./ 2.) * B( 12)** 2
     . * B( 78)

      C(  8,  1,  4)=
     . - ( 1./ 1.) * B( 64)
     . + ( 1./ 24.) * B( 12)** 4 * B( 77) + ( 1./ 6.) * B( 12)**
     . 3 * B( 78)

      C(  8,  1,  5)=
     . - ( 1./ 1.) * B( 65)
     . + ( 3./ 24.) * B( 12)** 4 * B( 77) + ( 1./ 2.) * B( 12)**
     . 3 * B( 78)

      C(  8,  1,  6)=
     . - ( 1./ 1.) * B( 66)
     . + ( 1./ 24.) * B( 12)** 4 * B( 77) + ( 1./ 6.) * B( 12)**
     . 3 * B( 78)

      C(  8,  1,  7)=
     . - ( 1./ 1.) * B( 67)
     . + ( 1./ 24.) * B( 12)** 4 * B( 77) + ( 1./ 6.) * B( 12)**
     . 3 * B( 78)

      C(  8,  1,  8)=
     . - ( 1./ 1.) * B( 68)
     . + ( 1./ 120.) * B( 12)** 5 * B( 77) + ( 1./ 24.) * B( 12)**
     . 4 * B( 78)

      C(  8,  1,  9)=
     . - ( 1./ 1.) * B( 69)
     . + ( 6./ 120.) * B( 12)** 5 * B( 77) + ( 1./ 4.) * B( 12)**
     . 4 * B( 78)

      C(  8,  1, 10)=
     . - ( 1./ 1.) * B( 70)
     . + ( 4./ 120.) * B( 12)** 5 * B( 77) + ( 1./ 6.) * B( 12)**
     . 4 * B( 78)

      C(  8,  1, 11)=
```

```
. - ( 1./ 1.) * B( 71)
. + ( 4./ 120.) * B( 12)** 5 * B( 77) + ( 1./ 6.) * B( 12)**
. 4 * B( 78)

 C(  8,  1, 12)=
. - ( 1./ 1.) * B( 72)
. + ( 1./ 120.) * B( 12)** 5 * B( 77) + ( 1./ 24.) * B( 12)**
. 4 * B( 78)

 C(  8,  1, 13)=
. - ( 1./ 1.) * B( 73)
. + ( 3./ 120.) * B( 12)** 5 * B( 77) + ( 3./ 24.) * B( 12)**
. 4 * B( 78)

 C(  8,  1, 14)=
. - ( 1./ 1.) * B( 74)
. + ( 1./ 120.) * B( 12)** 5 * B( 77) + ( 1./ 24.) * B( 12)**
. 4 * B( 78)

 C(  8,  1, 15)=
. - ( 1./ 1.) * B( 75)
. + ( 1./ 120.) * B( 12)** 5 * B( 77) + ( 1./ 24.) * B( 12)**
. 4 * B( 78)

 C(  8,  1, 16)=
. - ( 1./ 1.) * B( 76)
. + ( 3./ 120.) * B( 12)** 5 * B( 77) + ( 1./ 8.) * B( 12)**
. 4 * B( 78)



 DEFINE THE UNDETERMINED PARAMETERS B(/.../) BY PRINTING OUT
THE EXPANSIONS OF  E(/I,K/), I IN  M.= (1,...,E*Q), K IN  P.=
(0,...,ORDER - 1), WITH RESPECT TO THESE PARAMETERS, 'IF' MODE
'EQUAL' 0 'THEN' THEIR LOCAL ORIGINS ARE THE POINT 0 'ELSE'
'IF' MODE 'EQUAL'-1 'THEN' THEY ARE - I1 * H WHERE I1.= I'/' Q


 E(/   1,   0/).= U(/   0/) ( B(/   1/)) (/   0/) + SUM(I,0,  15,B(/  29
 + I/)* A(   0 * H)(/I/))

 E(/   2,   0/).= U(/   0/) ( B(/   2/)) (/   0/) + SUM(I,0,  15,B(/  45
 + I/)* A(   0 * H)(/I/))

 E(/   3,   0/).= U(/   0/) ( B(/   3/)) (/   0/) + SUM(I,0,  15,B(/  61
 + I/)* A(   0 * H)(/I/))


'COMMENT' LAST  1959  LAST1.=     0  TEMPO-LAST.=  4041  LIST LENGTH-TEM
PO.=  1000
NEXT FREE PARAMETER B(/    91/).,

END OF COMPUTATIONS.
```

TABLE III KNOWN COEFFICIENTS

```
COMMENT  THESE ARE RK4 CLASSIC COEFFICIENTS
B(   1)=(                0/          1)= 0.
B(   2)=(               -1/          2)=-5.0000000000000E-01
B(   3)=(               -1/          2)=-5.0000000000000E-01
B(   4)=(                1/          1)= 1.0000000000000E+00
B(  77)=(                1/          1)= 1.0000000000000E+00
B(  78)=(                1/          2)= 5.0000000000000E-01
B(  79)=(                1/          1)= 1.0000000000000E+00
B(  80)=(                0/          1)= 0.
B(  81)=(                1/          2)= 5.0000000000000E-01
B(  82)=(                1/          1)= 1.0000000000000E+00
B(  83)=(                0/          1)= 0.
B(  84)=(                0/          1)= 0.
B(  85)=(                1/          1)= 1.0000000000000E+00
B(  86)=(                1/          1)= 1.0000000000000E+00
B(  87)=(                1/          6)= 1.6666666666667E-01
B(  88)=(                1/          3)= 3.3333333333333E-01
B(  89)=(                1/          3)= 3.3333333333333E-01
B(  90)=(                1/          6)= 1.6666666666667E-01
```

COMMENT FOR THE ABOVE COEFFICIENTS THESE VALUES ARE OBTAINED

C(1, 1, 1)= 0. C(1, 1, 2)= 1.7763568394003E-15		ξ_0	Condition A
C(2, 1, 1)= 0. C(2, 1, 2)= 0.		ξ_1	
C(3, 1, 1)= 0. C(3, 1, 2)= 0.		ξ_2	
C(4, 1, 1)= 0. C(4, 1, 2)= 0.		ξ_3	
C(5, 1, 1)= 0.	h^2	ψ_0	Condition B ξ_0
C(5, 1, 2)= 0.	h^3	1	
C(5, 1, 3)= 0.		2	
C(5, 1, 4)= 2.7755575615629E-17		3	
C(5, 1, 5)= 4.4408920985006E-16	h^4	4	
C(5, 1, 6)= 3.3306690738755E-16		5	
C(5, 1, 7)= 2.7755575615629E-16		6	
C(5, 1, 8)= 3.4722222222222E-04		8	} Principal error term
C(5, 1, 9)= 2.0833333333333E-03		9	
C(5, 1,10)= -2.0833333333334E-03		10	
C(5, 1,11)= 8.3333333333332E-03		11	
C(5, 1,12)= -1.3888888888889E-03		12	
C(5, 1,13)= -4.1666666666667E-03		13	
C(5, 1,14)= 2.0833333333334E-03		14	
C(5, 1,15)= -8.3333333333331E-03		15	
C(5, 1,16)= 6.2500000000000E-03		21	

TABLE III CONTINUED

```
COMMENT THESE ARE RK4 GILL COEFFICIENTS

B(   1)=(                0/                1)= 0.
B(   2)=(               -1/                2)=-5.0000000000000E-01
B(   3)=(               -1/                2)=-5.0000000000000E-01
B(   4)=(                1/                1)= 1.0000000000000E+00
B(  77)=(                1/                1)= 1.0000000000000E+00
B(  78)=(                1/                2)= 5.0000000000000E-01
B(  79)=(                1/                1)= 1.0000000000000E+00
B(  82)=(                1/                1)= 1.0000000000000E+00
B(  83)=(                0/                1)= 0.
B(  86)=(                1/                1)= 1.0000000000000E+00
B(  87)=(                1/                6)= 1.6666666666667E-01
B(  90)=(                1/                6)= 1.6666666666667E-01

      B(80)= (SQRT(2.)-1.)/2.
      B(81)= (2.-SQRT(2.))/2.
      B(84)= -SQRT(2.)/2.
      B(85)= 1.+SQRT(2.)/2.
      B(88)= (2.-SQRT(2.))/6.
      B(89)= (2.+SQRT(2.))/6.
```

COMMENT FOR THE ABOVE COEFFICIENTS THESE VALUES ARE OBTAINED

C(1, 1, 1)= 0. C(1, 1, 2)= 5.3290705182008E-15		ξ_0	Condition A	
C(2, 1, 1)= 0. C(2, 1, 2)= 0.		ξ_1		
C(3, 1, 1)= 0. C(3, 1, 2)= 0.		ξ_2		
C(4, 1, 1)= 0. C(4, 1, 2)= 0.		ξ_3		
C(5, 1, 1)= -1.7763568394003E-15	h^2	ψ_0	Condition B	ξ_0
C(5, 1, 2)= 4.4408920985006E-16	h^3	1		
C(5, 1, 3)= 4.4408920985006E-16		2		
C(5, 1, 4)= 5.5511151231258E-17		3		
C(5, 1, 5)= 2.2204460492503E-16	h^4	4		
C(5, 1, 6)= 4.1633363423443E-16		5		
C(5, 1, 7)= 5.5511151231258E-17		6		
C(5, 1, 8)= 3.4722222222221E-04		8		
C(5, 1, 9)= 2.0833333333334E-03		9		
C(5, 1,10)= -2.0833333333333E-03		10		
C(5, 1,11)= 8.3333333333333E-03		11		
C(5, 1,12)= -1.3888888888887E-03	h^5	12	Principal error term	
C(5, 1,13)= -4.1666666666662E-03		13		
C(5, 1,14)= 2.0833333333335E-03		14		
C(5, 1,15)= -8.3333333333331E-03		15		
C(5, 1,16)= 1.9352753919470E-03		21		

TABLE III CONTINUED

COMMENT THESE ARE RK4 STRACHEY COEFFICIENTS		
B(1)=(0/	1)= 0.	
B(2)=(-1/	2)=-5.0000000000000E-01	
B(3)=(-1/	2)=-5.0000000000000E-01	
B(4)=(1/	1)= 1.0000000000000E+00	
B(77)=(1/	1)= 1.0000000000000E+00	
B(78)=(1/	2)= 5.0000000000000E-01	
B(79)=(1/	1)= 1.0000000000000E+00	
B(80)=(-1/	2)=-5.0000000000000E-01	
B(81)=(1/	1)= 1.0000000000000E+00	
B(82)=(1/	1)= 1.0000000000000E+00	
B(83)=(0/	1)= 0.	
B(84)=(1/	2)= 5.0000000000000E-01	
B(85)=(1/	2)= 5.0000000000000E-01	
B(86)=(1/	1)= 1.0000000000000E+00	
B(87)=(1/	6)= 1.6666666666667E-01	
B(88)=(3/	6)= 5.0000000000000E-01	
B(89)=(1/	6)= 1.6666666666667E-01	
B(90)=(1/	6)= 1.6666666666667E-01	

COMMENT FOR THE ABOVE COEFFICIENTS THESE VALUES ARE OBTAINED			
C(1, 1, 1)= 0. C(1, 1, 2)= 3.5527136788005E-15		ξ_0	Condition A
C(2, 1, 1)= 0. C(2, 1, 2)= 0.		ξ_1	
C(3, 1, 1)= 0. C(3, 1, 2)= 0.		ξ_2	
C(4, 1, 1)= 0. C(4, 1, 2)= 0.		ξ_3	
C(5, 1, 1)= -8.8817841970013E-16	h^2	ψ_0	Condition B ξ_0
C(5, 1, 2)= 2.2204460492503E-16	h^3	1	
C(5, 1, 3)= 0.		2	
C(5, 1, 4)= 1.3877787807814E-17		3	
C(5, 1, 5)= 4.4408920985006E-16	h^4	4	
C(5, 1, 6)= 7.7715611723761E-16		5	
C(5, 1, 7)= 9.4368957093138E-16		6	
C(5, 1, 8)= 3.4722222222222E-04		8	Principal error term
C(5, 1, 9)= 2.0833333333333E-03		9	
C(5, 1,10)= -2.0833333333337E-03		10	
C(5, 1,11)= 8.3333333333329E-03	h^5	11	
C(5, 1,12)= -1.3888888888890E-03		12	
C(5, 1,13)= -4.1666666666664E-03		13	
C(5, 1,14)= 2.0833333333333E-03		14	
C(5, 1,15)= -8.3333333333333E-03		15	
C(5, 1,16)= 1.6666666666667E-02		21	

TABLE III CONTINUED

COMMENT THESE ARE RK4CK COEFFICIENTS	
B(1)=(0/	1)= 0.
B(2)=(-2/	5)=-4.0000000000000E-01
B(3)=(-3/	5)=-6.0000000000000E-01
B(4)=(1/	1)= 1.0000000000000E+00
B(77)=(1/	1)= 1.0000000000000E+00
B(78)=(2/	5)= 4.0000000000000E-01
B(79)=(1/	1)= 1.0000000000000E+00
B(80)=(-3/	20)=-1.5000000000000E-01
B(81)=(3/	4)= 7.5000000000000E-01
B(82)=(1/	1)= 1.0000000000000E+00
B(83)=(19/	44)= 4.3181818181818E-01
B(84)=(-15/	44)=-3.4090909090909E-01
B(85)=(40/	44)= 9.0909090909091E-01
B(86)=(1/	1)= 1.0000000000000E+00
B(87)=(55/	360)= 1.5277777777778E-01
B(88)=(125/	360)= 3.4722222222222E-01
B(89)=(125/	360)= 3.4722222222222E-01
B(90)=(55/	360)= 1.5277777777778E-01

COMMENT FOR THE ABOVE COEFFICIENTS THESE VALUES ARE OBTAINED				
C(1, 1, 1)= 0. C(1, 1, 2)= -8.8817841970013E-16		ξ_0	Condition A	
C(2, 1, 1)= 0. C(2, 1, 2)= 3.5527136788005E-15		ξ_1		
C(3, 1, 1)= 0. C(3, 1, 2)= 0.		ξ_2		
C(4, 1, 1)= 0. C(4, 1, 2)= -3.5527136788005E-15		ξ_3		
C(5, 1, 1)= 1.7763568394003E-15	h^2	ψ_0	Condition B	ξ_0
C(5, 1, 2)= -3.3306690738755E-16	h^3	1		
C(5, 1, 3)= 7.7715611723761E-16		2		
C(5, 1, 4)= 5.1347814888913E-16		3		
C(5, 1, 5)= -1.1102230246252E-16	h^4	4		
C(5, 1, 6)= -5.5511151231258E-17		5		
C(5, 1, 7)= 4.4408920985006E-16		6		
C(5, 1, 8)= 2.7777777777768E-04		8	Principal error term	
C(5, 1, 9)= 0.		9		
C(5, 1,10)= 0.		10		
C(5, 1,11)= 8.3333333333329E-03	h^5	11		
C(5, 1,12)= -1.1111111111110E-03		12		
C(5, 1,13)= 1.1102230246252E-16		13		
C(5, 1,14)= 0.		14		
C(5, 1,15)= -8.3333333333330E-03		15		
C(5, 1,16)= 3.4090909090909E-03		21		

TABLE IV DATA INPUT TO EXAMPLE 2

DATA	COMMENT
1,	Control
1,1,1,1,1,3,3,3,1,2,1,1,0, 2,	FORTRAN Output
50,70,6000,7000,10,10,10,	Set program parameters
1,3,2,2,1,7,3,	Particularize problem
'FALSE',	No input-output
	Data table input
0,	Derivative harmonics
1,2,1,1, 0,0,0,0,0,0, 'FALSE'	D^4x See Table VI
0, 1,2,1,1, 1,2,1,1, 0,0,0,0, 'FALSE'	D^5x Appendix I
0,0,0, 1,2,1,1, 1,2,5,1, 1,2,1,1, 1,2,1,1, 'FALSE'	D^6x
	Substitution harmonics
2,1, 2,2,1,6,1,3, 2,2,1,24,1,4, 3,2,1,1,1,0,2,0,0, 2,2,1,120,1,5, 3,2,1,1,1,1,2,0,0, 3,2,1,1,1,0,2,0,1, 3,2,1,1,1,0,2,0,2, 'FALSE'	See Table VI 0 ψ_i Appendix I 1 2 3 4 5 6
1,0,	Translation harmonics
1,2,1,1, 1,4,-1,1,10,1, 1,4,-1,1,10,1, 1,4,1,2,10,2, 1,4,5,2,10,2, 1,4,1,2,10,2, 1,4,1,2,10,2, 'FALSE'	A_0 See Table VII Appendix I

TABLE IV CONT.

DATA	COMMENT
0, 1,2,1,1, 0, 1,4,-1,1,10,1, 1,4,-4,1,10,1, 0, 0, 'FALSE'	A_1
0,0, 1,2,1,1, 0, 1,4,-1,1,10,1, 1,4,-1,1,10,1, 1,4,-1,1,10,1, 'FALSE'	A_2
0,0,0, 1,2,1,1, 0, 0, 0, 'FALSE'	A_3
0,0,0,0, 1,2,1,1, 0,0, 'FALSE'	A_4
0,0,0,0,0, 1,2,1,1, 0, 'FALSE'	A_5
0,0,0,0,0,0, 1,2,1,1, 'FALSE'	A_6
-1	End data table input
COMMENT EXAMPLE 2, THE RANK IS 2, THE EXTENT IS 2. THIS CORRESPONDS TO A K= 2 FINITE DIFFERENCE SCHEME AND SHOULD BE COMPARED WITH THE K= 2 CASE OF BUTCHER, JACM,12,1,1965, P.130.,	
0,-1,	Forward translation
1,'TRUE',4, 1,'TRUE',3, 1,'TRUE',2, 1,'TRUE',1, 3,1,5, 0,2,0,4,1,0,1,1,1,2, 3,0,6, 0,2,0,4,1,0,1,1,1,2,1,3,	Scheme
4,-1,	Print all parameter equations
-4,'FALSE',	Exit from program

```
TABLE V  OUTPUT FROM RKM1 EXAMPLE 2


     CONTROL .=       1

 DATA INPUT THAT SETS COMPUTER VARIABLES

I.=0., 'FOR' PRINT LENGTH(/I/).=  1,  1,  1,  1,  1,  3,  3,  3,  1,  2,
  1,  1,  0, 'DO' I.= I+1.,
     TYPE SET .=       2
     N FOR PRINT .=      50
     LINE LENGTH .=      70
     ORIGIN OF TEMPORARY STORE .=    6000
     LIST LENGTH .=    7000
     MAXIMUM NUMBER OF VECTORS  N OR SUMS  S .=      10
     MAXIMUM LENGTH OF A SUM .=      10
     MAXIMUM LENGTH OF A LIST PRODUCT .=      10

DATA INPUT THAT PARTICULARIZES THE PROBLEM


     ORDER OF THE DIFFERENTIAL EQUATION .=       1
      D'POWER'(ORDER+UPPER)X IS THE FIRST NEGLECTED  TAYLOR TERM, UPPER
.=       3
     NUMBER OF POINTS IN ONE H INTERVAL .=       2
     NUMBER OF H INTERVALS .=       2
     PERIOD OF SCHEME .=       1
     NUMBER OF BASIC FUNCTIONS .=       7
     NUMBER OF DERIVATIVES WE ATTEMPT TO MATCH .=       3
                                      INOUT .= 'FALSE'

DATA TABLE INPUT


DUMMY PRINT SCHEME.  PRINT SCHEME.= 'FALSE'


COMMENT  EXAMPLE 2,  THE RANK IS 2, THE EXTENT IS 2.  THIS CORRESPONDS TO
         A K= 2 FINITE DIFFERENCE SCHEME AND SHOULD BE COMPARED WITH THE
         K= 2 CASE OF BUTCHER, JACM,12,1,1965,P.130.,



 N(/  0/).=  X( E(/  4/))


 N(/  1/).=  X( E(/  3/))


 N(/  2/).=  X( E(/  2/))


 N(/  3/).=  X( E(/  1/))

 E(/  1,  0/).=
SUM(I,0,  1,SUM(J,0,  0,B(/  23 + I *  1 + J/) *  E(/V(/  0,I/), J/)))+
SUM(I,0,  2,B(/  25 + I/) *  N(/V(/  1,I/)/))
```

```
WHERE  V(/   0/).=(   2,   4)  AND   V(/   0,I/).= V(/   0/)(/I/)
WHERE  V(/   1/).=(   0,   1,   2)  AND   V(/   1,I/).= V(/   1/)(/I/)


 E(/  0,  0/).=
SUM(I,0,  1,SUM(J,0,  0,B(/  28 + I *  1 + J/) *  E(/V(/  0,I/), J/)))+
SUM(I,0,  3,B(/  30 + I/) *  N(/V(/  1,I/)/))


WHERE  V(/   0/).=(   2,   4)  AND   V(/   0,I/).= V(/   0/)(/I/)
WHERE  V(/   1/).=(   0,   1,   2,   3)  AND   V(/   1,I/).= V(/   1/)(/
I/)


 E(/0/).=  E(0) + ERROR TERMS.   ALL DEFINING EQUATIONS ARE PRINTED BELO
W

C(/I,K,N/), WHERE K IS IN (0,...,ORDER-1) AND N IS IN (0,..., IM(/0/)-1)
'COMMENT' EQUATIONS ARISING FROM REQUIRING
 E(/.../).= U(/.../) (...) + SUM(I,0,IM(/0/)-1, B(/.../)* A(/.../)(/I/))
.,


      B( 4)=B 0
      B( 5)=B 0 + B( 1)
      B( 6)=B 0 -1 * B( 2)
      B( 7)=B 0 + B( 1) -1 * B( 2)
      B( 8)=B 0 -2 * B( 2)

'COMMENT' CONDITIONS ON  E(/  0/).,


      C(  1,  1,  1)=
     .( 1./ 1.) * B( 28) + ( 1./ 1.) * B( 29) - ( 1./ 1.)

      C(  1,  1,  2)=
     .( 1./ 1.) * B( 6) * B( 28) + ( 1./ 1.) * B( 8) * B( 29) + (
     . 1./ 1.) * B( 30) + ( 1./ 1.) * B( 31) + ( 1./ 1.) * B( 32)
     . + ( 1./ 1.) * B( 33) - ( 1./ 1.) * B( 4)

      C(  1,  1,  3)=
     .( 1./ 2.) * B( 6)** 2 * B( 28) + ( 1./ 2.) * B( 8)** 2 * B(
     . 29) + ( 1./ 1.) * B( 8) * B( 30) + ( 1./ 1.) * B( 7) * B(
     . 31) + ( 1./ 1.) * B( 6) * B( 32) + ( 1./ 1.) * B( 5) * B(
     . 33) - ( 1./ 2.) * B( 4)** 2

      C(  1,  1,  4)=
     .( 1./ 6.) * B( 6)** 3 * B( 28) + ( 1./ 6.) * B( 8)** 3 * B(
     . 29) + ( 1./ 2.) * B( 8)** 2 * B( 30) + ( 1./ 2.) * B( 7)**
     . 2 * B( 31) + ( 1./ 2.) * B( 6)** 2 * B( 32) + ( 1./ 2.) * B(
     . 5)** 2 * B( 33) - ( 1./ 6.) * B( 4)** 3

'COMMENT' CONDITIONS CN  E(/  1/).,


      C(  2,  1,  1)=
     .( 1./ 1.) * B( 23) + ( 1./ 1.) * B( 24) - ( 1./ 1.)

      C(  2,  1,  2)=
```

```
     .( 1./ 1.) * B( 6) * B( 23) + ( 1./ 1.) * B( 8) * B( 24) + (
     . 1./ 1.) * B( 25) + ( 1./ 1.) * B( 26) + ( 1./ 1.) * B( 27)
     . - ( 1./ 1.) * B( 5)

      C(  2,  1,  3)=
     .( 1./ 2.) * B( 6)** 2 * B( 23) + ( 1./ 2.) * B( 8)** 2 * B(
     . 24) + ( 1./ 1.) * B( 8) * B( 25) + ( 1./ 1.) * B( 7) * B(
     . 26) + ( 1./ 1.) * B( 6) * B( 27) - ( 1./ 2.) * B( 5)** 2

      C(  2,  1,  4)=
     .( 1./ 6.) * B( 6)** 3 * B( 23) + ( 1./ 6.) * B( 8)** 3 * B(
     . 24) + ( 1./ 2.) * B( 8)** 2 * B( 25) + ( 1./ 2.) * B( 7)**
     . 2 * B( 26) + ( 1./ 2.) * B( 6)** 2 * B( 27) - ( 1./ 6.) * B(
     . 5)** 3

'COMMENT'  EQUATIONS ARISING FROM  E(/0,K/) -  E(0)(/K/).=  O(H'POWER'(K
+  7)).,

      C(  3,  1,  1)=
     . + ( 1./ 24.) * B( 6)** 4 * B( 28)
     . + ( 1./ 24.) * B( 8)** 4 * B( 29) + ( 1./ 6.) * B( 8)** 3 *
     .B( 30) + ( 1./ 6.) * B( 7)** 3 * B( 31) + ( 1./ 6.) * B( 6)**
     . 3 * B( 32) + ( 1./ 6.) * B( 5)** 3 * B( 33)

      C(  3,  1,  2)=
     . + ( 1./ 120.) * B( 6)** 5 * B( 28)
     . + ( 1./ 120.) * B( 8)** 5 * B( 29) + ( 1./ 24.) * B( 8)**
     . 4 * B( 30) + ( 1./ 24.) * B( 7)** 4 * B( 31) + ( 1./ 24.) *
     .B( 6)** 4 * B( 32) + ( 1./ 24.) * B( 5)** 4 * B( 33)

      C(  3,  1,  3)=
     . + ( 1./ 120.) * B( 6)** 5 * B( 28)
     . + ( 1./ 120.) * B( 8)** 5 * B( 29) + ( 1./ 24.) * B( 8)**
     . 4 * B( 30) + ( 1./ 1.) * B( 16) * B( 31) + ( 1./ 24.) * B(
     . 6)** 4 * B( 31) + ( 1./ 6.) * B( 1) * B( 6)** 3 * B( 31) + (
     . 1./ 4.) * B( 1)** 2 * B( 6)** 2 * B( 31) + ( 1./ 6.) * B(
     . 1)** 3 * B( 6) * B( 31) + ( 1./ 24.) * B( 6)** 4 * B( 32) +
     .( 1./ 1.) * B( 16) * B( 33)

      C(  3,  1,  4)=
     . + ( 1./ 720.) * B( 6)** 6 * B( 28)
     . + ( 1./ 720.) * B( 8)** 6 * B( 29) + ( 1./ 120.) * B( 8)**
     . 5 * B( 30) + ( 1./ 120.) * B( 7)** 5 * B( 31) + ( 1./ 120.)
     . * B( 6)** 5 * B( 32) + ( 1./ 120.) * B( 5)** 5 * B( 33)

      C(  3,  1,  5)=
     . + ( 5./ 720.) * B( 6)** 6 * B( 28)
     . + ( 5./ 720.) * B( 8)** 6 * B( 29) + ( 1./ 24.) * B( 8)**
     . 5 * B( 30) + ( 1./ 1.) * B( 7) * B( 16) * B( 31) + ( 1./
     . 24.) * B( 6)** 4 * B( 7) * B( 31) + ( 1./ 6.) * B( 1) * B(
     . 6)** 3 * B( 7) * B( 31) + ( 1./ 4.) * B( 1)** 2 * B( 6)**
     . 2 * B( 7) * B( 31) + ( 1./ 6.) * B( 1)** 3 * B( 6) * B( 7)
     . * B( 31) + ( 1./ 24.) * B( 6)** 5 * B( 32) + ( 1./ 1.) * B(
     . 5) * B( 16) * B( 33)

      C(  3,  1,  6)=
     . + ( 1./ 720.) * B( 6)** 6 * B( 28)
     . + ( 1./ 720.) * B( 8)** 6 * B( 29) + ( 1./ 120.) * B( 8)**
     . 5 * B( 30) + ( 1./ 1.) * B( 6) * B( 16) * B( 31) + ( 1./ 1.)
     . * B( 17) * B( 31) + ( 1./ 120.) * B( 6)** 5 * B( 31) + ( 1./
     . 24.) * B( 1) * B( 6)** 4 * B( 31) + ( 1./ 12.) * B( 1)** 2
     . * B( 6)** 3 * B( 31) + ( 1./ 12.) * B( 1)** 3 * B( 6)** 2 *
```

```
.B( 31) + ( 1./ 120.) * B( 6)** 5 * B( 32) + ( 1./ 1.) * B(
. 17) * B( 33)

 C(  3,  1,  7)=
. + ( 1./ 720.) * B( 6)** 6 * B( 28)
. + ( 1./ 720.) * B( 8)** 6 * B( 29) + ( 1./ 120.) * B( 8)**
. 5 * B( 30) + ( 1./ 1.) * B( 6) * B( 16) * B( 31) + ( 1./ 1.)
. * B( 18) * B( 31) + ( 1./ 120.) * B( 6)** 5 * B( 31) + ( 1./
. 24.) * B( 1) * B( 6)** 4 * B( 31) + ( 1./ 12.) * B( 1)** 2
. * B( 6)** 3 * B( 31) + ( 1./ 12.) * B( 1)** 3 * B( 6)** 2 *
.B( 31) + ( 1./ 120.) * B( 6)** 5 * B( 32) + ( 1./ 1.) * B(
. 18) * B( 33)


'COMMENT'  EQUATIONS WHICH DEFINE THE UNDETERMINED PARAMETERS
USED IN THE EXPANSION OF  E(/   1/).,


 C(  4,  1,  1)=
. - ( 1./ 1.) * B( 16)
. + ( 1./ 24.) * B( 6)** 4 * B( 23) + ( 1./ 24.) * B( 8)** 4
. * B( 24) + ( 1./ 6.) * B( 8)** 3 * B( 25) + ( 1./ 6.) * B(
. 7)** 3 * B( 26) + ( 1./ 6.) * B( 6)** 3 * B( 27)

 C(  4,  1,  2)=
. - ( 1./ 1.) * B( 17)
. + ( 1./ 120.) * B( 6)** 5 * B( 23) + ( 1./ 120.) * B( 8)**
. 5 * B( 24) + ( 1./ 24.) * B( 8)** 4 * B( 25) + ( 1./ 24.) *
.B( 7)** 4 * B( 26) + ( 1./ 24.) * B( 6)** 4 * B( 27)

 C(  4,  1,  3)=
. - ( 1./ 1.) * B( 18)
. + ( 1./ 120.) * B( 6)** 5 * B( 23) + ( 1./ 120.) * B( 8)**
. 5 * B( 24) + ( 1./ 24.) * B( 8)** 4 * B( 25) + ( 1./ 1.) * B
.( 16) * B( 26) + ( 1./ 24.) * B( 6)** 4 * B( 26) + ( 1./ 6.)
. * B( 1) * B( 6)** 3 * B( 26) + ( 1./ 4.) * B( 1)** 2 * B(
. 6)** 2 * B( 26) + ( 1./ 6.) * B( 1)** 3 * B( 6) * B( 26) + (
. 1./ 24.) * B( 6)** 4 * B( 27)

 C(  4,  1,  4)=
. - ( 1./ 1.) * B( 19)
. + ( 1./ 720.) * B( 6)** 6 * B( 23) + ( 1./ 720.) * B( 8)**
. 6 * B( 24) + ( 1./ 120.) * B( 8)** 5 * B( 25) + ( 1./ 120.)
. * B( 7)** 5 * B( 26) + ( 1./ 120.) * B( 6)** 5 * B( 27)

 C(  4,  1,  5)=
. - ( 1./ 1.) * B( 20)
. + ( 5./ 720.) * B( 6)** 6 * B( 23) + ( 5./ 720.) * B( 8)**
. 6 * B( 24) + ( 1./ 24.) * B( 8)** 5 * B( 25) + ( 1./ 1.) * B
.( 7) * B( 16) * B( 26) + ( 1./ 24.) * B( 6)** 4 * B( 7) * B(
. 26) + ( 1./ 6.) * B( 1) * B( 6)** 3 * B( 7) * B( 26) + ( 1./
. 4.) * B( 1)** 2 * B( 6)** 2 * B( 7) * B( 26) + ( 1./ 6.) * B
.( 1)** 3 * B( 6) * B( 7) * B( 26) + ( 1./ 24.) * B( 6)** 5 *
.B( 27)

 C(  4,  1,  6)=
. - ( 1./ 1.) * B( 21)
. + ( 1./ 720.) * B( 6)** 6 * B( 23) + ( 1./ 720.) * B( 8)**
. 6 * B( 24) + ( 1./ 120.) * B( 8)** 5 * B( 25) + ( 1./ 1.) *
.B( 6) * B( 16) * B( 26) + ( 1./ 1.) * B( 17) * B( 26) + ( 1./
. 120.) * B( 6)** 5 * B( 26) + ( 1./ 24.) * B( 1) * B( 6)**
```

```
. 4 * B( 26) + ( 1./ 12.) * B( 1)** 2 * B( 6)** 3 * B( 26) + (
. 1./ 12.) * B( 1)** 3 * B( 6)** 2 * B( 26) + ( 1./ 120.) * B(
. 6)** 5 * B( 27)

 C(  4,  1,  7)=
. - ( 1./ 1.) * B( 22)
. + ( 1./ 720.) * B( 6)** 6 * B( 23) + ( 1./ 720.) * B( 8)**
. 6 * B( 24) + ( 1./ 120.) * B( 8)** 5 * B( 25) + ( 1./ 1.) *
.B( 6) * B( 16) * B( 26) + ( 1./ 1.) * B( 18) * B( 26) + ( 1./
. 120.) * B( 6)** 5 * B( 26) + ( 1./ 24.) * B( 1) * B( 6)**
. 4 * B( 26) + ( 1./ 12.) * B( 1)** 2 * B( 6)** 3 * B( 26) + (
. 1./ 12.) * B( 1)** 3 * B( 6)** 2 * B( 26) + ( 1./ 120.) * B(
. 6)** 5 * B( 27)
```

```
 DEFINE THE UNDETERMINED PARAMETERS B(/.../) BY PRINTING OUT
THE EXPANSIONS OF  E(/I,K/), I IN  M.= (1,...,E*Q), K IN  P.=
(0,...,ORDER - 1), WITH RESPECT TO THESE PARAMETERS, 'IF' MODE
'EQUAL' 0 'THEN' THEIR LOCAL ORIGINS ARE THE POINT 0 'ELSE'
'IF' MODE 'EQUAL'-1 'THEN' THEY ARE - I1 * H WHERE I1.= I'/' Q

 E(/   1,   0/).= U(/   0/) ( B(/   1/)) (/   0/) + SUM(I,0,   6,B(/  16
 + I/)* A(   0 * H)(/I/))

 E(/   3,   0/).= U(/  -1 * H/) ( B(/   1/)) (/   0/) + SUM(I,0,   6,B(/
  16 + I/)* A(  -1 * H)(/I/))



'COMMENT' LAST  1154  LAST1.=     0  TEMPO-LAST.=  4846  LIST LENGTH-TEM
PO.=  1000
NEXT FREE PARAMETER B(/    34/).,

END OF COMPUTATIONS.
```

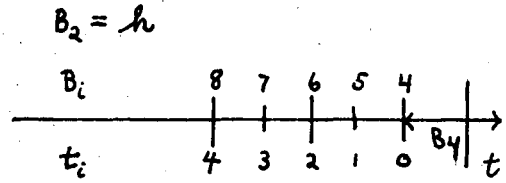
TABLE VI EXAMPLE 1 SCHEME PARAMETERS -35/128

PARAMETER VALUES	COMMENT
B(1)= -2.7343750000000E-01 B(2)= 1.0000000000000E+00 B(4)= 0. B(5)= -2.7343750000000E-01 B(6)= -1.0000000000000E+00 B(7)= -1.2734375000000E+00	$B_2 = h$ B_i 8 7 6 5 4 t_i 4 3 2 1 0 B_4 t
B(16)= 3.7384033203118E-02 B(17)= -1.0341838995613E-01 B(18)= -1.0341838995612E-01 B(19)= 9.1921752008281E-02 B(20)= 2.7576525602483E-01 B(21)= 1.8846137946288E-01 B(22)= 1.8846137946287E-01	0 ψ_i Harmonics of ξ_1 about t_0 1 2 3 4 5 6
B(23)= 1.0000000000000E+00 B(24)= -9.6529017857132E-01 B(25)= 1.6918526785713E+00 B(26)= 1.0000000000000E+00 B(27)= -1.0952380952381E-01 B(28)= 4.6257040450589E-01 B(29)= 6.4695340501791E-01	ξ_i | ξ | X i | 2 | 3 | 2 | 1 1 | 23 | 24 | 25 | 0 | 26 | 27 | 28 | 29

FOR THE ABOVE PARAMETER VALUES THESE RESULTS ARE OBTAINED

C(1, 1, 1)= 0. C(1, 1, 2)= -3.5527136788005E-15	ξ_0	Condition A
C(2, 1, 1)= 4.9737991503207E-14 C(2, 1, 2)= -4.9737991503207E-14	ξ_1	
C(3, 1, 1)= 0.	h^2 ψ_0	Condition B ξ_0
C(3, 1, 2)= 2.2204460492503E-16	h^3 1	
C(3, 1, 3)= -2.6645352591004E-15	2	
C(3, 1, 4)= 6.2730577257364E-05 C(3, 1, 5)= 1.8819173177109E-04 C(3, 1, 6)= -5.3686135912681E-02 C(3, 1, 7)= -5.3686135912679E-02	h^4 3 4 5 6	Principal error term

TABLE VII EXAMPLE 2 SCHEME PARAMETERS

PARAMETER VALUES -7/32	COMMENT
B(1)= -2.1875000000000E-01 B(2)= 1.0000000000000E+00 B(4)= 0. B(5)= -2.1875000000000E-01 B(6)= -1.0000000000000E+00 B(7)= -1.2187500000000E+00 B(8)= -2.0000000000000E+00	
B(16)= 9.5407168007711E-05 B(17)= -1.8593644288477E-02 B(18)= -1.8593644287844E-02 B(19)= 2.3451935087751E-02 B(20)= 1.1725967543796E-01 B(21)= 2.1069520381382E-01 B(22)= 2.1069520380745E-01	0 ψ_i Harmonics of ξ_1 about t_o 1 2 3 4 5 6
B(23)= 8.5435638427522E+00 B(24)= -7.5435638427513E+00 B(25)= -1.8735885620054E+00 B(26)= -1.0072544642836E+01 B(27)= 5.1838193620896E+00 B(28)= 1.3003331746788E+00 B(29)= -3.0033317467873E-01 B(30)= -5.9487796120479E-02 B(31)= -7.7146392874144E-01 B(32)= 9.9055959747060E-01 B(33)= 5.4005895271257E-01	ξ_i | ξ | X i | 2 | 4 | 4 | 3 | 2 | 1 1 | 23 | 24 | 25 | 26 | 27 | 0 | 28 | 29 | 30 | 31 | 32 | 33

FOR THE ABOVE PARAMETER VALUES THESE RESULTS ARE OBTAINED

C(1, 1, 1)= 6.3948846218409E-14 C(1, 1, 2)= -8.1712414612412E-14 C(1, 1, 3)= 4.9293902293357E-14 C(1, 1, 4)= -2.1260770921572E-14		ξ_0	Condition A
C(2, 1, 1)= 9.0949470177293E-13 C(2, 1, 2)= -1.1368683772162E-12 C(2, 1, 3)= 7.3896444519050E-13 C(2, 1, 4)= -4.0738939999230E-13		ξ_1	
C(3, 1, 1)= 7.4766581814600E-15	h^4	ψ_0	Condition B ξ_0
C(3, 1, 2)= -3.0179851673307E-15	h^5	1	
C(3, 1, 3)= 1.2157592640949E-14		2	
C(3, 1, 4)= 2.7488253780019E-06	h^6	3	Principal error term
C(3, 1, 5)= 1.3744126839082E-05		4	
C(3, 1, 6)= 4.3044447371057E-03		5	
C(3, 1, 7)= 4.3044447369593E-03		6	

TABLE VII CONT. EXAMPLE 2 STABILITY CHECK -7/32

I	1	2	3
X(I)	-2.1875E-01		
A(I)	3.0033E-01	-1.3003E+00	1.0000E+00
RHO(I)	3.0033E-01	1.0000E+00	
THETA(I)	0.	0.	

PARAMETER VALUES 177/256	COMMENT
B(1)= 6.9140625000000E-01 B(2)= 1.0000000000000E+00 B(4)= 0. B(5)= 6.9140625000000E-01 B(6)= -1.0000000000000E+00 B(7)= -3.0859375000000E-01 B(8)= -2.0000000000000E+00	See previous case with $t_1 = -7/32$
B(16)= 9.5218637434513E-03 B(17)= -1.2899824826048E-01 B(18)= -1.2899824826040E-01 B(19)= 1.1508940408733E-01 B(20)= 5.7544702043659E-01 B(21)= -3.6947052581091E-01 B(22)= -3.6947052581061E-01	0 ψ_i Harmonics of ξ_0 about t_o 1 2 3 4 5 6
B(23)= 8.8304000415720E+00 B(24)= -7.8304000415719E+00 B(25)= -2.7747193164008E+00 B(26)= 3.7183757821901E+00 B(27)= -7.0826502573614E+00 B(28)= 9.2205493921958E-01 B(29)= 7.7945060780429E-02 B(30)= 1.9005618281631E-02 B(31)= 6.8211932288409E-01 B(32)= 3.7184702411883E-01 B(33)= 4.9730954958851E-03	See previous case with $t_1 = -7/32$

FOR THE ABOVE PARAMETER VALUES THESE RESULTS ARE OBTAINED

C(1, 1, 1)= 0. C(1, 1, 2)= 7.5772721430667E-15 C(1, 1, 3)= -1.2059797604991E-14 C(1, 1, 4)= 5.1417203827953E-15	ξ_0	Condition A
C(2, 1, 1)= 1.1368683772162E-13 C(2, 1, 2)= -2.2737367544323E-13 C(2, 1, 3)= 3.1263880373444E-13 C(2, 1, 4)= -3.1241675912952E-13	ξ_1	

TABLE VII CONT.

C(3, 1, 1)= -1.6930901125534E-15	h^4	ψ_0	Condition A
C(3, 1, 2)= 4.9960036108132E-16	h^5	1	
C(3, 1, 3)= 1.5482840703962E-14		2	
C(3, 1, 4)= 3.2831713151999E-05		3	Principal error term
C(3, 1, 5)= 1.6415856575530E-04	h^6	4	
C(3, 1, 6)= -8.9505577954445E-02		5	
C(3, 1, 7)= -8.9505577954389E-02		6	

EXAMPLE 2 STABILITY CHECK 177/256

I	1	2	3
X(I)	6.9141E-01		
A(I)	-7.7945E-02	-9.2205E-01	1.0000E+00
RHO(I)	7.7945E-02	1.0000E+00	
THETA(I)	1.8000E+02	0.	

TABLE VIII EXAMPLE 3

PARAMETER VALUES -55/256	COMMENT
B(1)= -2.1484375000000E-01 B(2)= 1.0000000000000E+00 B(4)= 0. B(5)= -2.1484375000000E-01 B(6)= -1.0000000000000E+00 B(7)= -1.2148437500000E+00 B(8)= -2.0000000000000E+00 B(10)= -3.0000000000000E+00	$B_2 = h$ B_i 10 9 8 7 6 5 4 t_i 6 5 4 3 2 1 0 B_4 t
B(18)= 1.3658533459804E-07 B(19)= -3.5596361412327E-03 B(20)= -3.5596361433155E-03 B(21)= 6.0425894389404E-03 B(22)= 4.2298126075085E-02 B(23)= 1.1208166814845E-01 B(24)= 1.1208166821050E-01	0 Harmonics of ξ_1 about t_0 1 2 3 4 5 6
B(25)= 3.5976231668263E+01 B(26)= -3.1099918878961E+01 B(27)= -3.8763127892998E+00 B(28)= -9.7817667588986E-01 B(29)= -1.5254413783933E+01 B(30)= -2.9789337836564E+01 B(31)= 7.9545400888260E+00 B(32)= 1.0923319157677E+00 B(33)= -1.2643169976527E-01 B(34)= 3.4099783997631E-02 B(35)= 8.9961593964621E-03 B(36)= 4.7634536908740E-02 B(37)= -6.5949997267148E-01 B(38)= 1.0110443847661E+00 B(39)= 5.3359275983016E-01	ξ_i | ξ | X i | 2 4 6 | 6 4 3 2 1 1 | 25 26 27 | 28 29 30 31 0 | 32 33 34 | 35 36 37 38 39

FOR THE ABOVE PARAMETER VALUES THESE RESULTS ARE OBTAINED

C(1, 1, 1)= 9.9475983006414E-14 C(1, 1, 2)= -1.1368683772162E-13 C(1, 1, 3)= 6.0840221749459E-14 C(1, 1, 4)= -2.1260770921572E-14 C(1, 1, 5)= 7.5876804839226E-15 C(1, 1, 6)= -1.1492543028346E-15	ξ_0	Condition A
C(2, 1, 1)= 2.1032064978499E-12 C(2, 1, 2)= -3.2969182939269E-12 C(2, 1, 3)= 2.5011104298756E-12 C(2, 1, 4)= -1.6674162051089E-12 C(2, 1, 5)= 8.0054539386420E-13 C(2, 1, 6)= -2.9013830183874E-13	ξ_1	
C(3, 1, 1)= -3.9952850056091E-17	h^6 ψ_0	Condition B ξ_0
C(3, 1, 2)= 9.9261673506363E-17 C(3, 1, 3)= -8.6935611941436E-15	h^7 1 2	

TABLE VIII CONT.

C(3, 1, 4)= -2.9878394336950E-05 C(3, 1, 5)= -2.0914876031072E-04 C(3, 1, 6)= 4.1830494316906E-04 C(3, 1, 7)= 4.1830494343131E-04	h^8	3 4 5 6

Principal error term

EXAMPLE 3 STABILITY CHECK -55/256

I	1	2	3
X(I)	-2.1484E-01		
A(I)	-3.4100E-02	1.2643E-01	-1.0923E+00
RHO(I)	1.8466E-01	1.8466E-01	1.0000E+00
THETA(I)	-7.5522E+01	7.5522E+01	0.

I	4
A(I)	1.0000E+00

PARAMETER VALUES 189/256	COMMENT
B(1)= 7.3828125000000E-01 B(2)= 1.0000000000000E+00 B(4)= 0. B(5)= 7.3828125000000E-01 B(6)= -1.0000000000000E+00 B(7)= -2.6171875000000E-01 B(8)= -2.0000000000000E+00 B(10)= -3.0000000000000E+00	See previous case with $t_1 = -55/256$
B(18)= 2.2490470311531E-04 B(19)= -4.9111151787269E-02 B(20)= -4.9111151785584E-02 B(21)= 6.9066665619596E-02 B(22)= 4.8346665933678E-01 B(23)= -2.0539578168642E-01 B(24)= -2.0539578167701E-01	0 ψ_i Harmonics of ξ_0 about t_o 1 2 3 4 5 6
B(25)= 5.6803094838622E+01 B(26)= -3.6292289144468E+01 B(27)= -1.9510805694152E+01 B(28)= -5.4327395775294E+00 B(29)= -4.3689122954257E+01 B(30)= 5.5858992806255E+00 B(31)= -3.0039656031613E+01 B(32)= 6.9220393663531E-01 B(33)= 2.7689156653732E-01 B(34)= 3.0904496827364E-02 B(35)= 7.1917265804906E-03 B(36)= 1.4057619056241E-01 B(37)= 6.1183394188573E-01 B(38)= 5.7737323850185E-01 B(39)= 1.7254626615661E-03	See previous case with $t_1 = -55/256$

TABLE VIII CONT.

FOR THE ABOVE PARAMETER VALUES THESE RESULTS ARE OBTAINED			
C(1, 1, 1)= 0. C(1, 1, 2)= 8.6181062286528E-15 C(1, 1, 3)= -1.2392864512378E-14 C(1, 1, 4)= 1.3163081735712E-14 C(1, 1, 5)= -5.2410333017949E-15 C(1, 1, 6)= 2.7792438489493E-15		ξ_0	Condition A
C(2, 1, 1)= 2.2737367544323E-12 C(2, 1, 2)= -4.2064129956998E-12 C(2, 1, 3)= 5.1159076974727E-12 C(2, 1, 4)= -4.9449333516804E-12 C(2, 1, 5)= 2.4159008127356E-12 C(2, 1, 6)= -1.3045467484041E-12		ξ_1	
C(3, 1, 1)= -6.1640959263948E-16	h^6	ψ_0	Condition B ξ_0
C(3, 1, 2)= 2.6697800871098E-16	h^7	1	
C(3, 1, 3)= 1.8530294486341E-13		2	
C(3, 1, 4)= -1.3235635893351E-06	h^8	3	Principal error term
C(3, 1, 5)= -9.2649451731932E-06		4	
C(3, 1, 6)= -3.0148486496469E-02		5	
C(3, 1, 7)= -3.0148486495435E-02		6	

EXAMPLE 3 STABILITY CHECK 189/256

I	1	2	3
X(I)	7.3828E-01		
A(I)	-3.0904E-02	-2.7689E-01	-6.9220E-01
RHO(I)	1.7580E-01	1.7580E-01	1.0000E+00
THETA(I)	-1.5110E+02	1.5110E+02	0.

I	4
A(I)	1.0000E+00

TABLE IX EXAMPLE 4

PARAMETER VALUES -13/64	COMMENT
B(1)= -2.0312500000000E-01 B(2)= 1.0000000000000E+00 B(4)= 0. B(5)= -2.0312500000000E-01 B(6)= -1.0000000000000E+00 B(7)= -1.2031250000000E+00 B(8)= -2.0000000000000E+00 B(10)= -3.0000000000000E+00 B(12)= -4.0000000000000E+00	$B_2 = h$ B_i 12 11 10 9 8 7 6 5 4 t_i 8 7 6 5 4 3 2 1 0 B_4 t
B(20)= 7.1885178365427E-11 B(21)= -7.5284008171891E-04 B(22)= -7.5284008227385E-04 B(23)= 1.6231158189083E-03 B(24)= 1.4608042370796E-02 B(25)= 4.9377922165884E-02 B(26)= 4.9377922201086E-02	0 ψ_i Harmonics of ξ_0 about t_0 1 2 3 4 5 6
B(27)= 9.0836146607243E+01 B(28)= -5.7707017723012E+01 B(29)= -2.8882857280867E+01 B(30)= -3.2462716033597E+00 B(31)= -7.5385949973605E-01 B(32)= -1.4751932004396E+01 B(33)= -5.6894706452300E+01 B(34)= -6.3432869222755E+01 B(35)= 1.1418695084360E+01 B(36)= 1.7417250703161E-01 B(37)= 3.2180748571091E-01 B(38)= 4.5770073285158E-01 B(39)= 4.6319274405917E-02 B(40)= 1.0452001726922E-02 B(41)= 2.2680839056931E-01 B(42)= 7.6355523090345E-01 B(43)= -2.7778502802622E-01 B(44)= 1.1419380743356E+00 B(45)= 5.1119810512282E-01	ξ_0 | ξ | X 0 | 2 4 6 8 | 8 6 4 3 2 1 1 | 27 28 29 30 | 31 32 33 34 35 0 | 36 37 38 39 | 40 41 42 43 44 45

FOR THE ABOVE PARAMETER VALUES THESE RESULTS ARE OBTAINED

C(1, 1, 1)= 2.1316282072803E-14 C(1, 1, 2)= 5.6843418860808E-14 C(1, 1, 3)= -7.4162898044960E-14 C(1, 1, 4)= 1.0336176359260E-13 C(1, 1, 5)= -5.4678483962789E-14 C(1, 1, 6)= 3.0296294986631E-14 C(1, 1, 7)= -1.4030965245996E-14 C(1, 1, 8)= 5.7578180555519E-15	ξ_0	Condition A
C(2, 1, 1)= 4.4906300900038E-12 C(2, 1, 2)= -7.3328010330442E-12	ξ_1	

TABLE IX CONT.

C(2, 1, 3)= 7.3328010330442E-12		ξ_1	Condition A
C(2, 1, 4)= -7.3801798006201E-12			
C(2, 1, 5)= 4.7156358679024E-12			
C(2, 1, 6)= -2.5848360723970E-12			
C(2, 1, 7)= 1.5251306395059E-12			
C(2, 1, 8)= -7.2241295839177E-13			
C(3, 1, 1)= -3.1584707763270E-15	h^8	ψ_0	Condition B ξ_0
C(3, 1, 2)= 8.3084292100750E-16	h^9	1	
C(3, 1, 3)= 2.8729822594551E-15		2	
C(3, 1, 4)= -6.1688291913145E-06		3	Principal error term
C(3, 1, 5)= -5.5519462718045E-05	h^{10}	4	
C(3, 1, 6)= -1.8189154884590E-04		5	
C(3, 1, 7)= -1.8189154897543E-04		6	

EXAMPLE 4 STABILITY CHECK -13/64

I	1	2	3
X(I)	-2.0313E-01		
A(I)	-4.6319E-02	-4.5770E-01	-3.2181E-01
RHO(I)	1.0872E-01	6.5273E-01	6.5273E-01
THETA(I)	1.8000E+02	-1.2332E+02	1.2332E+02

I	4	5
A(I)	1.7417E-01	1.0000E+00
RHO(I)	1.0000E+00	
THETA(I)	0.	

PARAMETER VALUES 195/256	COMMENT
B(1)= 7.6171875000000E-01	See previous case with
B(2)= 1.0000000000000E+00	
B(4)= 0.	$t_1 = -13/64$
B(5)= 7.6171875000000E-01	
B(6)= -1.0000000000000E+00	
B(7)= -2.3828125000000E-01	
B(8)= -2.0000000000000E+00	
B(10)= -3.0000000000000E+00	
B(12)= -4.0000000000000E+00	
B(20)= 2.8108456175957E-06	0 ψ_i Harmonics of ξ_0 about t_o
B(21)= -1.6757738836915E-02	1
B(22)= -1.6757738819361E-02	2
B(23)= 3.2221594335501E-02	3
B(24)= 2.8999434901404E-01	4
B(25)= -9.8316394112856E-02	5
B(26)= -9.8316393976117E-02	6

TABLE IX CONT.

B(27)= 2.0536983350272E+02 B(28)= 4.3921408186741E+01 B(29)= -2.1197688558192E+02 B(30)= -3.6314356107544E+01 B(31)= -8.8063187484961E+00 B(32)= -1.3314678448849E+02 B(33)= -2.6831348403107E+02 B(34)= 7.7896031546224E+00 B(35)= -8.4736728436299E+01 B(36)= 3.7294517398069E-01 B(37)= 4.1772793391875E-01 B(38)= 1.9551458658608E-01 B(39)= 1.3812305514491E-02 B(40)= 2.9591527068352E-03 B(41)= 8.3992250276174E-02 B(42)= 4.2161843561475E-01 B(43)= 5.7085515817217E-01 B(44)= 7.6986036773435E-01 B(45)= 9.0865913013098E-04	See the previous case with $t_1 = -13/64$

FOR THE ABOVE PARAMETER VALUES THESE RESULTS ARE OBTAINED

C(1, 1, 1)= 7.1054273576010E-15 C(1, 1, 2)= 2.6794538809938E-14 C(1, 1, 3)= -5.1701698478013E-14 C(1, 1, 4)= 2.8249971806282E-14 C(1, 1, 5)= -1.6683703030207E-14 C(1, 1, 6)= 1.0284470757654E-14 C(1, 1, 7)= -4.0702169282550E-15 C(1, 1, 8)= 1.6216987876270E-15		ξ_0	Condition A
C(2, 1, 1)= -4.5474735088646E-12 C(2, 1, 2)= 1.3642420526594E-12 C(2, 1, 3)= 1.7280399333686E-11 C(2, 1, 4)= -3.7516212358923E-11 C(2, 1, 5)= 3.0240754345101E-11 C(2, 1, 6)= -1.8928844602861E-11 C(2, 1, 7)= 1.1777913713762E-11 C(2, 1, 8)= -6.6691679305800E-12		ξ_1	
C(3, 1, 1)= -4.6963466474807E-16	h^8	ψ_0	Condition B ξ_0
C(3, 1, 2)= 2.4597152544462E-16	h^9	1	
C(3, 1, 3)= 1.2887068716132E-12		2	
C(3, 1, 4)= 1.3754480577628E-07		3	
C(3, 1, 5)= 1.2379029477152E-06		4	
C(3, 1, 6)= -9.5814672040689E-03	h^{10}	5	Principal error term
C(3, 1, 7)= -9.5814671940321E-03		6	

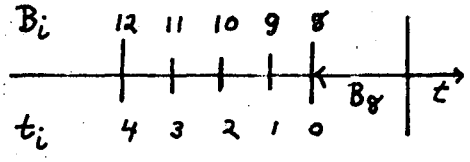
TABLE IX CONT.

EXAMPLE 4 STABILITY CHECK 195/256			
I	1	2	3
X(I)	7.6172E-01		
A(I)	-1.3812E-02	-1.9551E-01	-4.1773E-01
RHO(I)	8.4485E-02	4.0434E-01	4.0434E-01
THETA(I)	1.8000E+02	-1.3214E+02	1.3214E+02
I	4	5	
A(I)	-3.7295E-01	1.0000E 00	
RHO(I)	1.0000E+00		
THETA(I)	0.		

TABLE X EXAMPLE 2 START -7/32

INTERVAL PARAMETER SET 1

```
B(  8)=  7.8125000000000E-01
B(  9)=  7.8124999816152E-01
B( 10)=  5.2083333333333E-01
B( 11)=  2.6041666666666E-01
B( 12)=  0.
```



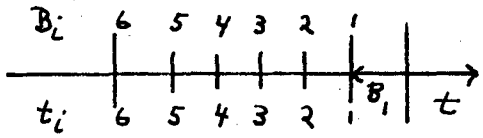
COEFFICIENT SET 1

```
B( 77)=  1.0000000000000E+00
B( 78)=  2.6041666666666E-01
B( 79)=  1.0000000000000E+00
B( 80)= -2.6041666990549E-01
B( 81)=  7.8125000323882E-01
B( 82)=  1.0000000000000E+00
B( 83)=  7.8124999261592E-01
B( 84)= -7.8124998516484E-01
B( 85)=  7.8124999071044E-01
B( 86)=  1.0000000000000E+00
B( 87)=  9.7656249780499E-02
B( 88)=  2.9296875100110E-01
B( 89)=  2.9296874796406E-01
B( 90)=  9.7656251254341E-02
```

ξ_i \ ξ	X				
i	4	4	3	2	1
3	77	78			
2	79	80	81		
1	82	83	84	85	
0	86	87	88	89	90

INTERVAL PARAMETER SET 3

```
B(  1)=  1.0000000000000E+00
B(  2)=  1.0000000000000E+00
B(  3)=  6.6666666666666E-01
B(  4)=  1.9999999999999E-01
B(  5)=  7.8125000000000E-01
B(  6)=  0.
```



COEFFICIENT SET 3

```
B(115)=  1.0000000000000E+00
B(116)=  1.7439999999999E-01
B(117)=  2.5599999999998E-02
B(118)=  1.0000000000000E+00
B(119)= -2.5037037037040E-01
B(120)=  6.6778176025487E-02
B(121)=  8.5025886101158E-01
B(122)=  1.0000000000000E+00
B(123)=  9.3599999999993E-01
B(124)= -4.0051612903217E-01
B(125)= -1.0783410138248E+00
B(126)=  1.5428571428570E+00
B(127)=  1.0000000000000E+00
B(128)=  4.1666666666650E-02
B(129)=  3.6613843203960E-14
B(130)=  3.7202380952384E-01
B(131)=  4.8214285714282E-01
B(132)=  1.0416666666666E-01
```

ξ_i \ ξ	X					
i	6	6	5	4	3	2
4	115	116	117			
3	118	119	120	121		
2	122	123	124	125	126	
1	127	128	129	130	131	132

TABLE X CONT. EXAMPLE 2 START 177/256

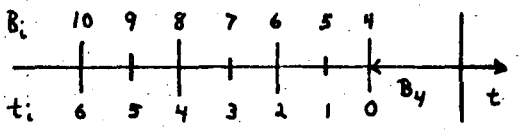
INTERVAL PARAMETER SET 1 B(8)= 1.6914062500000E+00 B(9)= 1.6914062499823E+00 B(10)= 1.1276041666667E+00 B(11)= 5.6380208333333E-01 B(12)= 0.	See previous case with $t_1 = -7/32$
COEFFICIENT SET 1 B(77)= 1.0000000000000E+00 B(78)= 5.6380208333333E-01 B(79)= 1.0000000000000E+00 B(80)= -5.6380208337246E-01 B(81)= 1.6914062500391E+00 B(82)= 1.0000000000000E+00 B(83)= 1.6914062499187E+00 B(84)= -1.6914062498198E+00 B(85)= 1.6914062498834E+00 B(86)= 1.0000000000000E+00 B(87)= 2.1142578124750E-01 B(88)= 6.3427734375933E-01 B(89)= 6.3427734373160E-01 B(90)= 2.1142578126158E-01	See previous case with $t_1 = -7/32$
INTERVAL PARAMETER SET 3 B(1)= 1.0000000000000E+00 B(2)= 1.0000000000000E+00 B(3)= 6.6666666666666E-01 B(4)= 2.0000000000009E-01 B(5)= 1.6914062500000E+00 B(6)= 0.	See previous case with $t_1 = -7/32$
COEFFICIENT SET 3 B(115)= 1.0000000000000E+00 B(116)= 1.8817551963056E-01 B(117)= 1.1824480369526E-02 B(118)= 1.0000000000000E+00 B(119)= -4.5445214267355E-01 B(120)= -1.3420485839614E-03 B(121)= 1.1224608579242E+00 B(122)= 1.0000000000000E+00 B(123)= 1.8314087759795E+00 B(124)= -3.6018519302512E-02 B(125)= -2.3382473995335E+00 B(126)= 1.5428571428564E+00 B(127)= 1.0000000000000E+00 B(128)= 4.1666666666742E-02 B(129)= -1.7551245388849E-15 B(130)= 3.7202380952380E-01 B(131)= 4.8214285714277E-01 B(132)= 1.0416666666670E-01	See previous case with $t_1 = -7/32$

TABLE XI EXAMPLE 3 START

START FOR -55/256

INTERVAL PARAMETER SET 1

```
B(  5)= -2.1484375000000E-01
B(  6)=  0.
B(  7)= -2.1484375000000E-01
B(  8)= -1.0000000000000E+00
B( 10)= -2.0000000000000E+00
```



COEFFICIENT SET 1

```
B( 25)=  8.0769388137674E-01
B( 26)=  1.4709500246681E-01
B( 27)=  4.5211116156451E-02
B( 28)=  1.2699127279347E-02
B( 29)=  1.1549256053058E-01
B( 30)=  0.
B( 31)= -1.0551820303021E-01
B( 32)=  1.0716125982387E+00
B( 33)= -6.6416959663704E-02
B( 34)= -5.1956385750449E-03
B( 35)= -1.2609701702601E-03
B( 36)= -2.6073845525025E-02
B( 37)=  0.
B( 38)= -8.7058630558552E-02
B( 39)= -1.7725854055996E-01
```

ξ_i	ξ			X				
i	2	4	6	6	4	3	2	1
1	25	26	27	27	29	30	31	
0	32	33	34	35	36	37	38	39

STABILITY CHECK OF CORRECTOR

I	1	2	3
X(I)	-2.1484E-01		
A(I)	5.1956E-03	6.6417E-02	-1.0716E+00
RHO(I)	4.4678E-02	1.1629E-01	1.0000E+00
THETA(I)	1.8000E+02	0.	0.

I	4
A(I)	1.0000E+00

TABLE XI CONT. EXAMPLE 3 START FOR 189/256

INTERVAL PARAMETER SET 1	
B(5)= 7.3828125000000E-01 B(6)= 0. B(7)= 7.3828125000000E-01 B(8)= -1.0000000000000E+00 B(10)= -2.0000000000000E+00	See previous set with t = -55/256
COEFFICIENT SET 1	
B(25)= -6.8810803488247E+00 B(26)= 4.0869543144945E+00 B(27)= 3.7941260343302E+00 B(28)= 1.1274617846823E+00 B(29)= 7.1042760544924E+00 B(30)= 0. B(31)= 4.1817497939803E+00 B(32)= -3.5911200522334E-01 B(33)= 1.0244778344542E+00 B(34)= 3.3463417076912E-01 B(35)= 8.9704991856244E-02 B(36)= 8.9041530527373E-01 B(37)= 0. B(38)= 1.2340394364734E+00 B(39)= 2.1786769238911E-01	See previous set with t = -55/256

STABILITY CHECK OF CORRECTOR

I	1	2	3
X(I)	7.3828E-01		
A(I)	-3.3463E-01	-1.0245E+00	3.5911E-01
RHO(I)	3.2296E-01	1.0000E+00	1.0362E+00
THETA(I)	1.8000E+02	0.	1.8000E+02

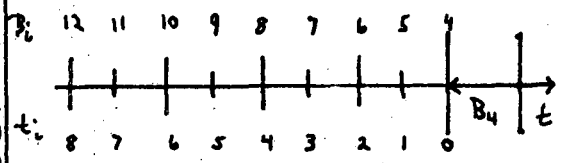
I	4
A(I)	1.0000E+00

TABLE XII EXAMPLE 4 START

START FOR -13/64

INTERVAL PARAMETER SET 1

```
B(  5)= -2.0312500000000E-01
B(  6)=  0.
B(  7)= -2.0312500000000E-01
B(  8)= -1.0000000000000E+00
B( 10)= -2.0000000000000E+00
B( 12)= -3.0000000000000E+00
```



COEFFICIENT SET 1

```
B( 27)=  7.7732624036978E-01
B( 28)=  5.2919044055955E-02
B( 29)=  1.4330664890230E-01
B( 30)=  2.6448066671965E-02
B( 31)=  6.5722405381052E-03
B( 32)=  9.2068517451195E-02
B( 33)=  2.0760548052721E-01
B( 34)=  0.
B( 35)= -9.0494696640064E-02
B( 36)=  1.0929348438486E+00
B( 37)= -7.1409776695964E-02
B( 38)= -1.9655777718015E-02
B( 39)= -1.8692894345904E-03
B( 40)= -4.2660791629733E-04
B( 41)= -9.3021170399682E-03
B( 42)= -4.7297384656830E-02
B( 43)=  0.
B( 44)= -8.0881326187540E-02
B( 45)= -1.8154676463513E-01
```

ξ_i	ξ				X					
i	2	4	6	8	8	6	4	3	2	1
1	27	28	29	30	31	32	33	34	35	
0	36	37	38	39	40	41	42	43	44	45

STABILITY CHECK OF CORRECTOR

I	1	2	3
X(I)	-2.0313E-01		
A(I)	1.8693E-03	1.9656E-02	7.1410E-02
RHO(I)	9.1087E-02	9.1087E-02	2.2530E-01
THETA(I)	-1.3660E+02	1.3660E+02	0.

I	4	5
A(I)	-1.0929E+00	1.0000E+00
RHO(I)	1.0000E+00	
THETA(I)	0.	

TABLE XII CONT.

START FOR 195/256

INTERVAL PARAMETER SET 1

```
B(  5)=  7.6171875000000E-01
B(  6)=  0.
B(  7)=  7.6171875000000E-01
B(  8)= -1.0000000000000E+00
B( 10)= -2.0000000000000E+00
B( 12)= -3.0000000000000E+00
```

See previous set with $t_1 = -13/64$

COEFFICIENT SET 1

```
B( 27)= -1.6683029324881E+01
B( 28)= -1.1924920363082E+01
B( 29)=  2.3964116341398E+01
B( 30)=  5.6438333465649E+00
B( 31)=  1.4351760001959E+00
B( 32)=  1.7593593201836E+01
B( 33)=  2.7580200429486E+01
B( 34)=  0.
B( 35)=  7.0875614778907E+00
B( 36)= -1.9308154367946E+00
B( 37)=  4.2207660908586E-01
B( 38)=  2.1894575618534E+00
B( 39)=  3.1928126585526E-01
B( 40)=  7.6049433303698E-02
B( 41)=  1.2698494619563E+00
B( 42)=  3.1205991376546E+00
B( 43)=  0.
B( 44)=  1.8547241922324E+00
B( 45)=  1.9933205521152E-01
```

See previous set with $t_1 = -13/64$

STABILITY CHECK OF CORRECTOR

I	1	2	3
X(I)	7.6172E-01		
A(I)	-3.1928E-01	-2.1895E+00	-4.2208E-01
RHO(I)	1.4433E+00	1.4433E+00	1.5328E-01
THETA(I)	-1.6421E+02	1.6421E+02	1.8000E+02

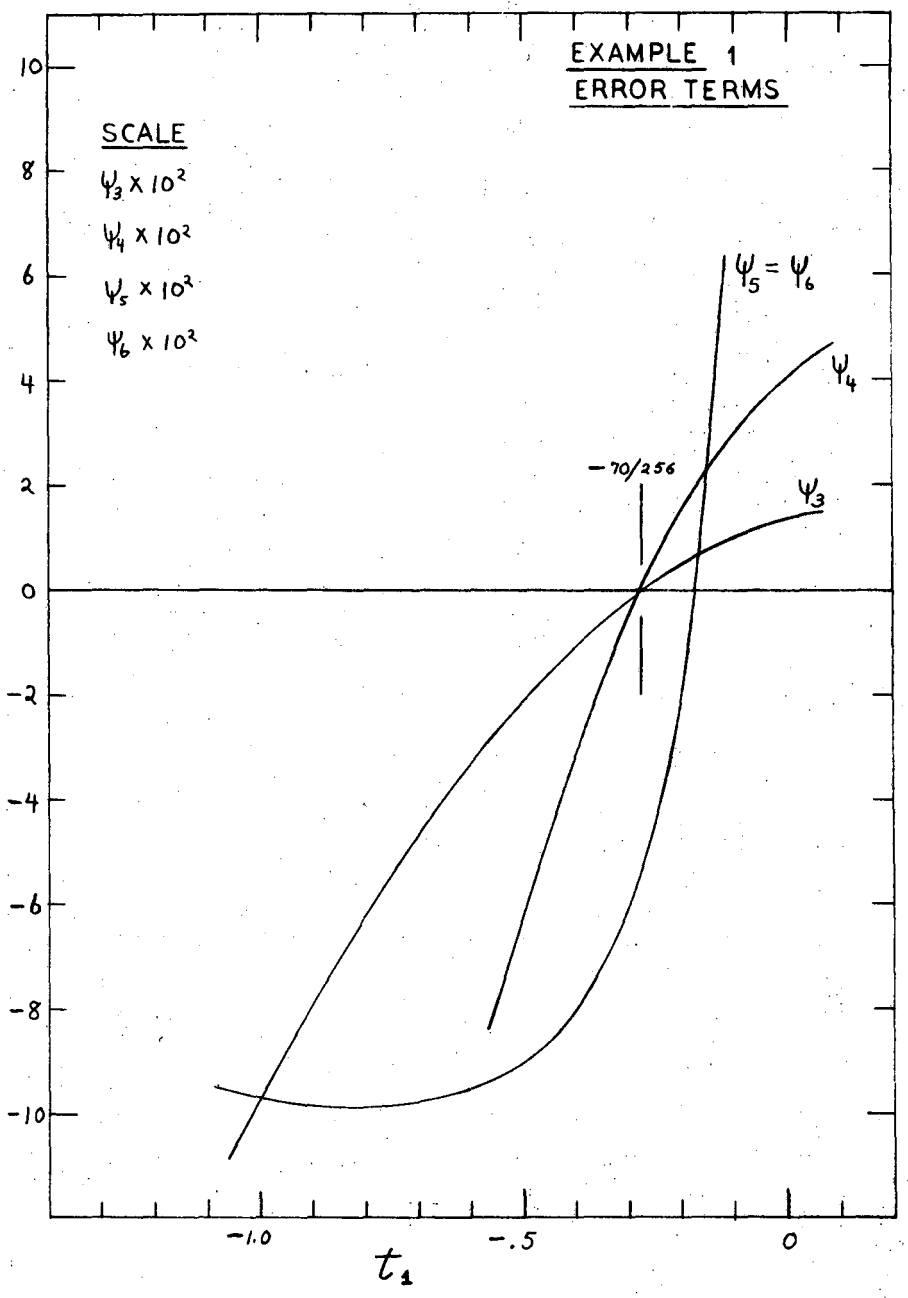
I	4	5
A(I)	1.9308E+00	1.0000E+00
RHO(I)	1.0000E+00	
THETA(I)	0.	

TABLE XIII

ERROR TERMS OF EXAMPLES. ERROR X SCALE

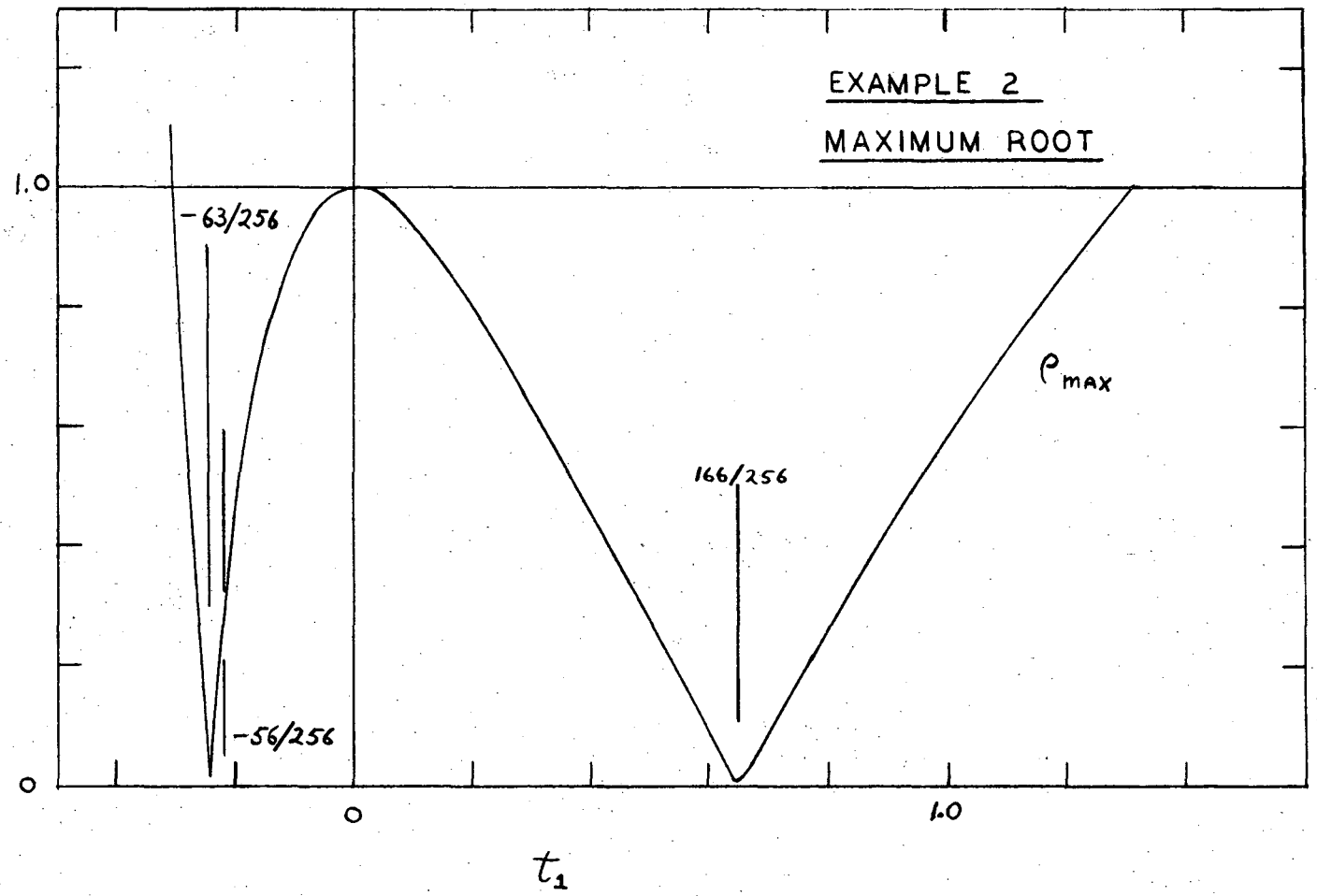
	EX. 7.1	RK3	EX. 7.2	EX. 7.2	JCB	EX. 7.3	EX. 7.3	JCB	EX. 7.4	EX. 7.4	JCB
SCALE	10^4	10^4	10^4	10^4	10^4	10^5	10^5	10^5	10^6	10^6	10^6
t	-35/128		-7/32	177/256	-1/2	-55/256	189/256	-1/2	-13/64	195/256	-1/3
ψ_3	0.627	0	0.027	0.328	1.792	-2.988	-0.123	3.473	-6.169	0.138	-.993
ψ_4	1.822	416.667	0.137	1.642	89.61	-20.91	-0.926	206.6	-55.52	1.238	522.4
ψ_5	-536.86	0	43.04	-895.1	-8.961	41.83	-3020.	-24.31	-181.9	-9581.	8.937
ψ_6	-536.86	-416.667	43.04	-895.1	89.61	41.83	-3020.	-206.6	-181.9	-9581.	-522.4

FIGURE 1



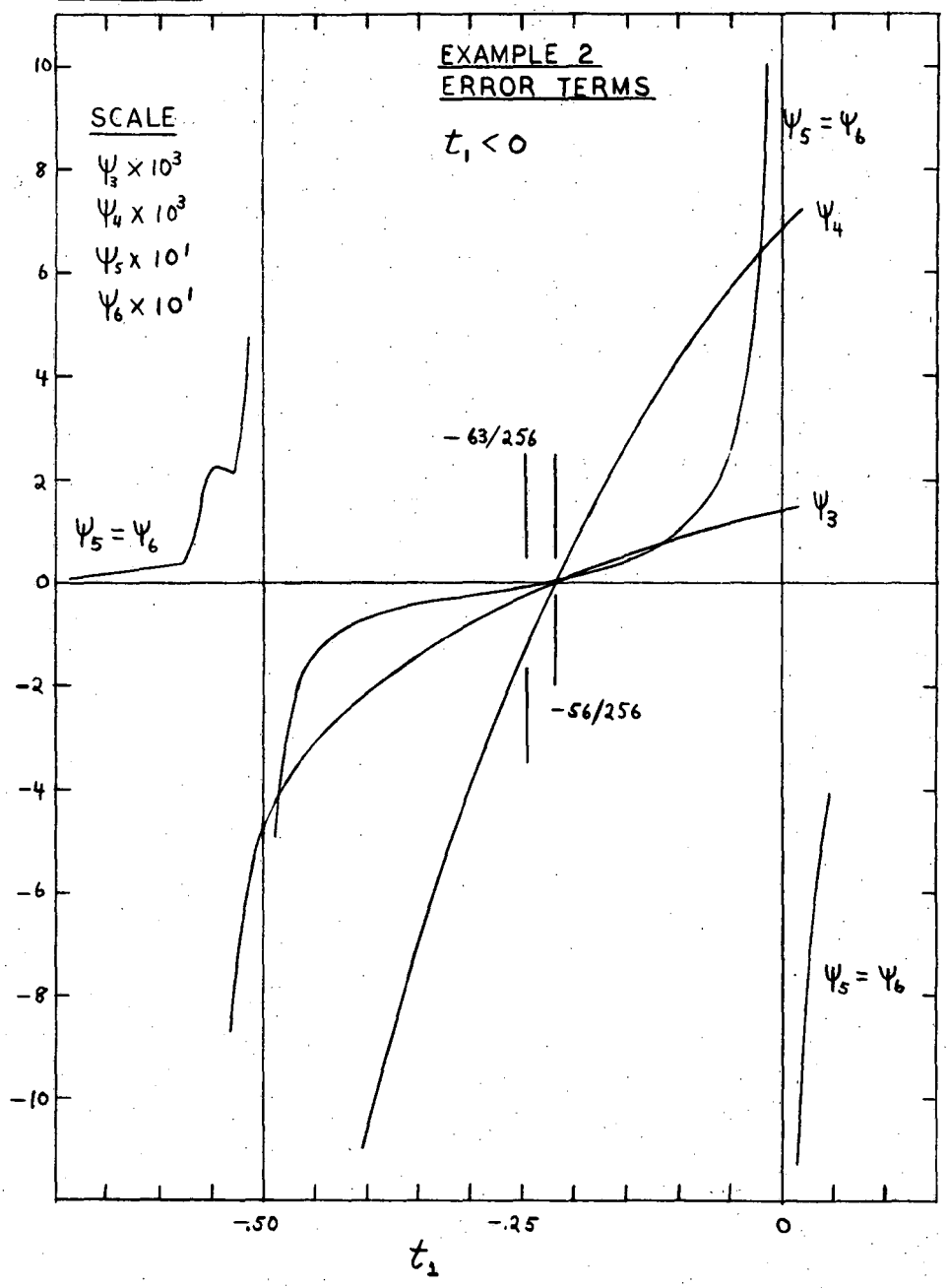
XBL 6810-6071

FIGURE 2



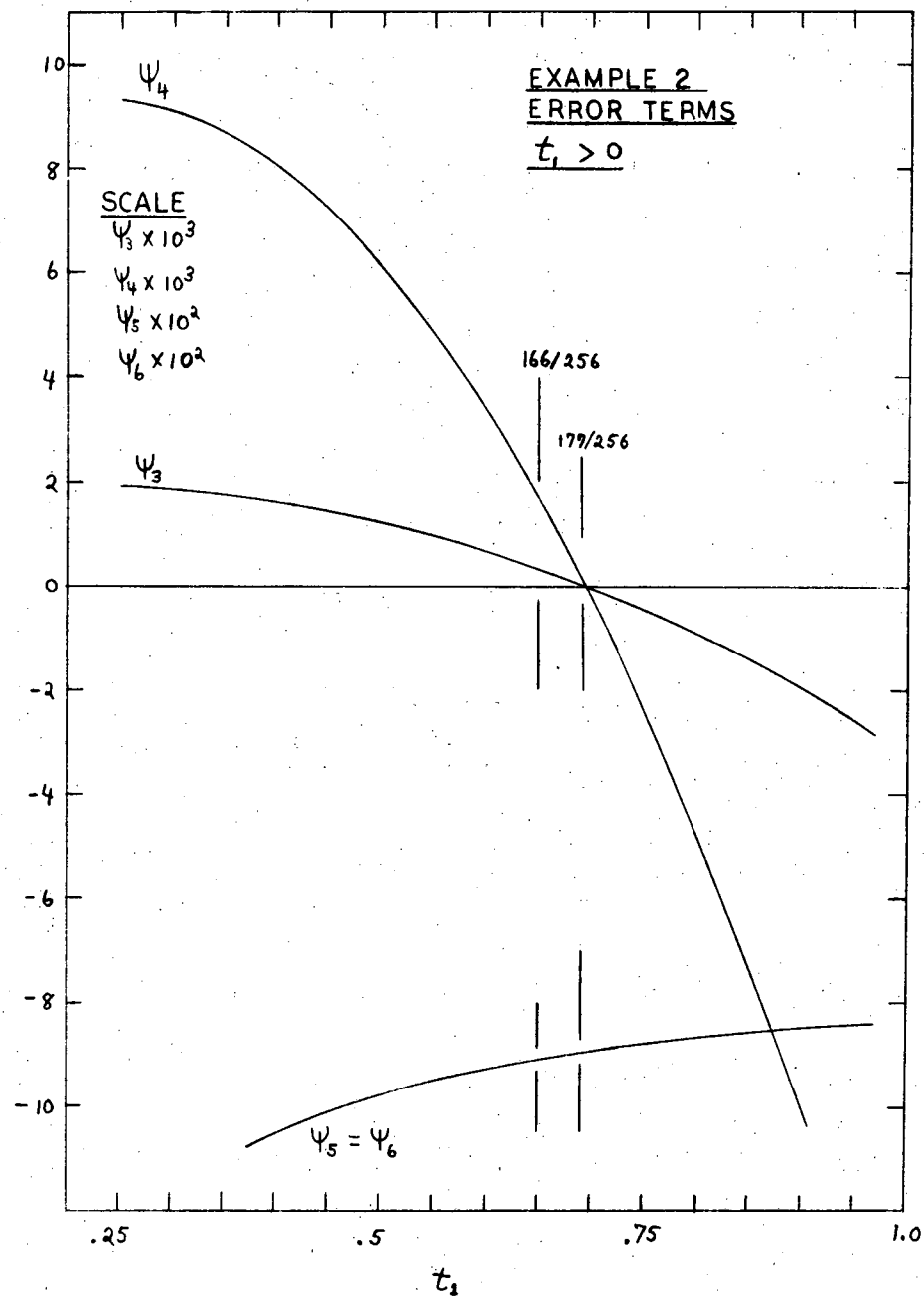
XBL 6810-6070

FIGURE 3



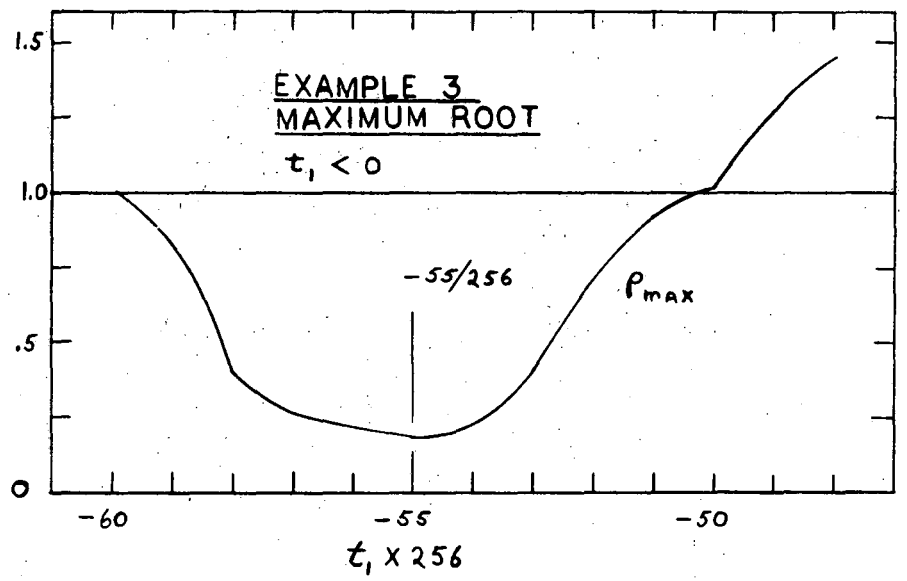
XBL 6810-6072

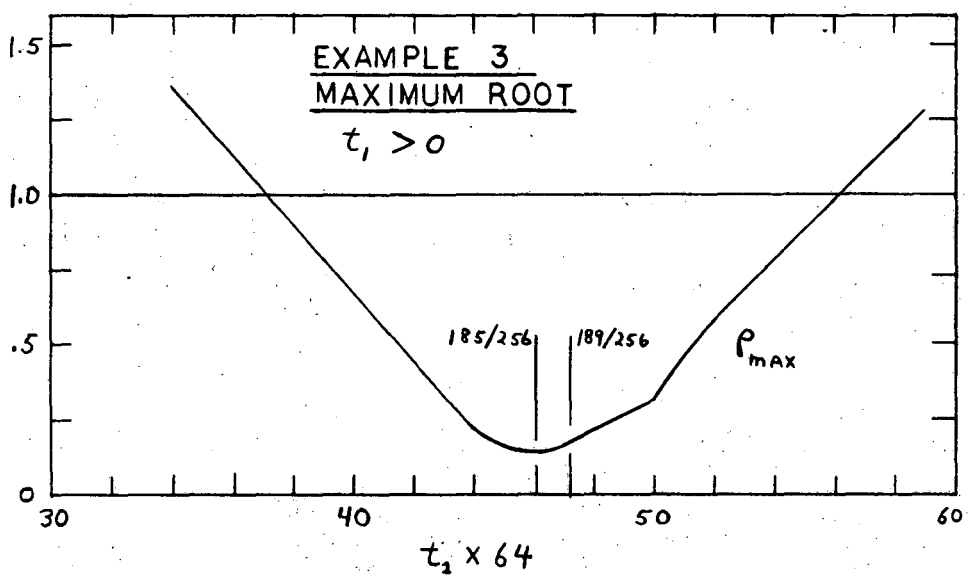
FIGURE 4



XBL 6810-6073

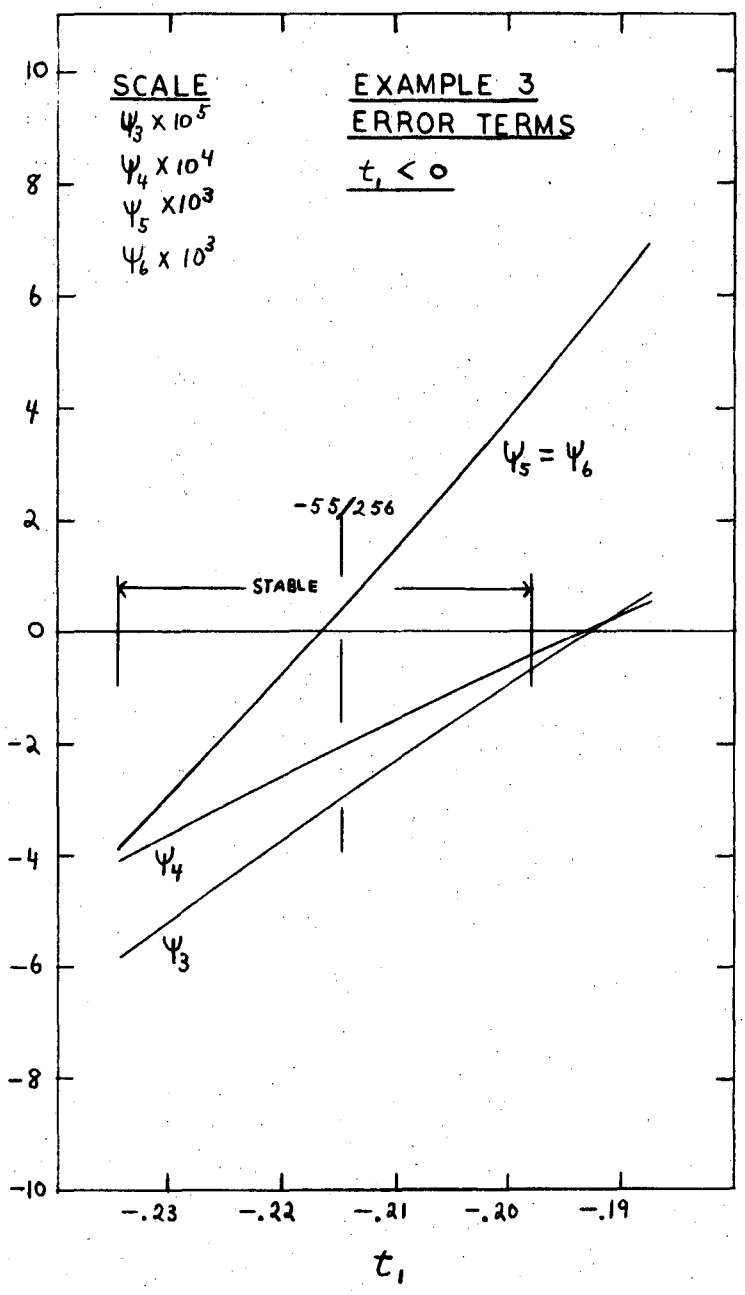
FIGURE 5





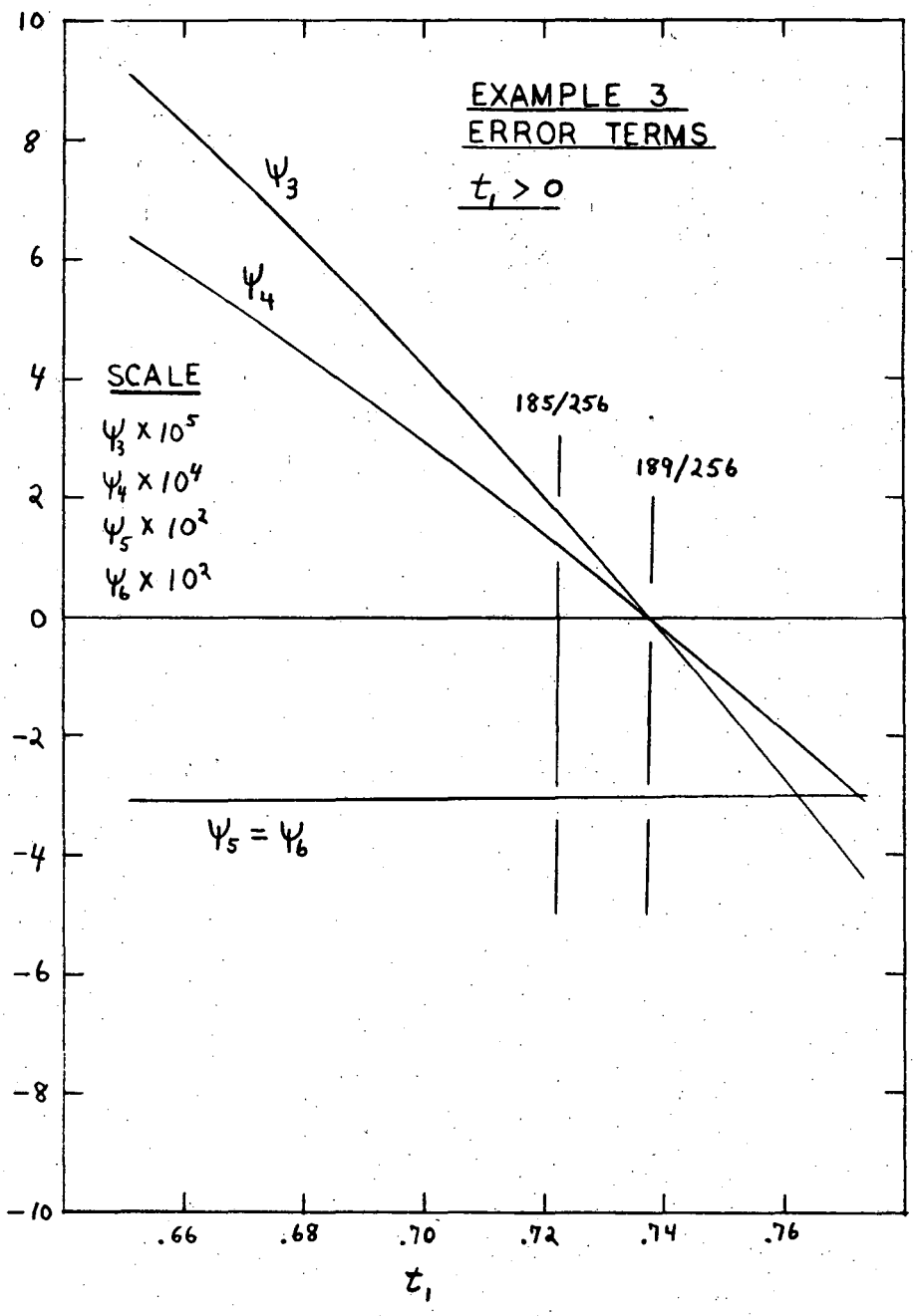
XBL 6810-6074

FIGURE 6



XBL 6810-6075

FIGURE 7

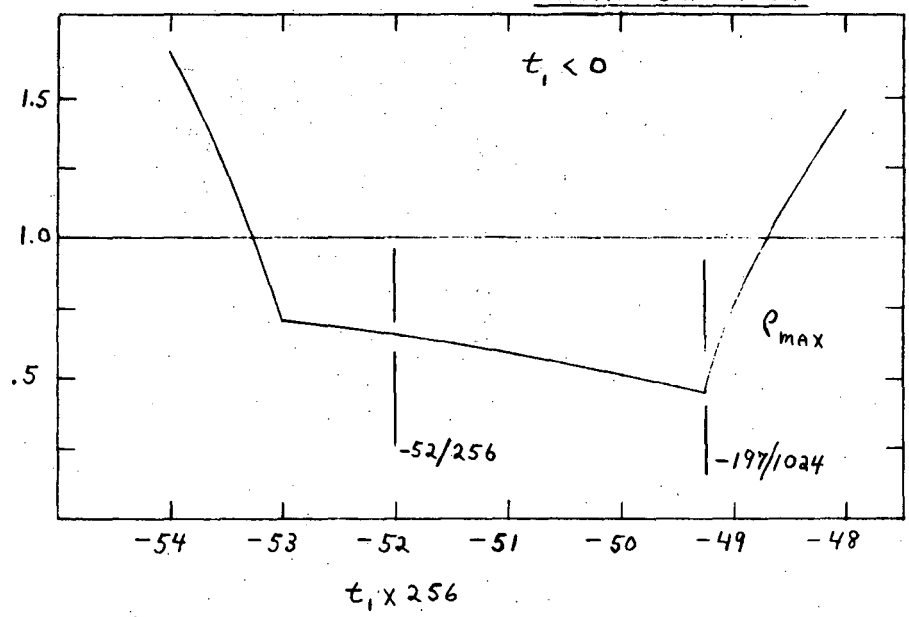


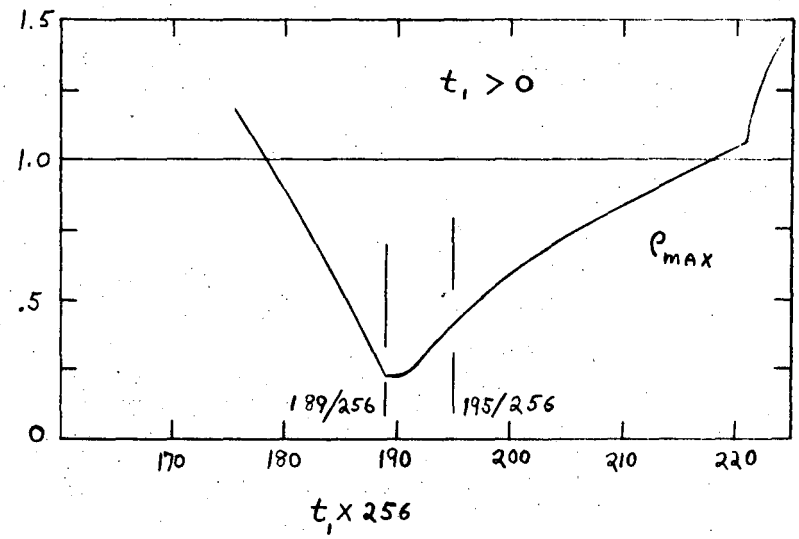
XBL 6810-6076

FIGURE 8

EXAMPLE 4

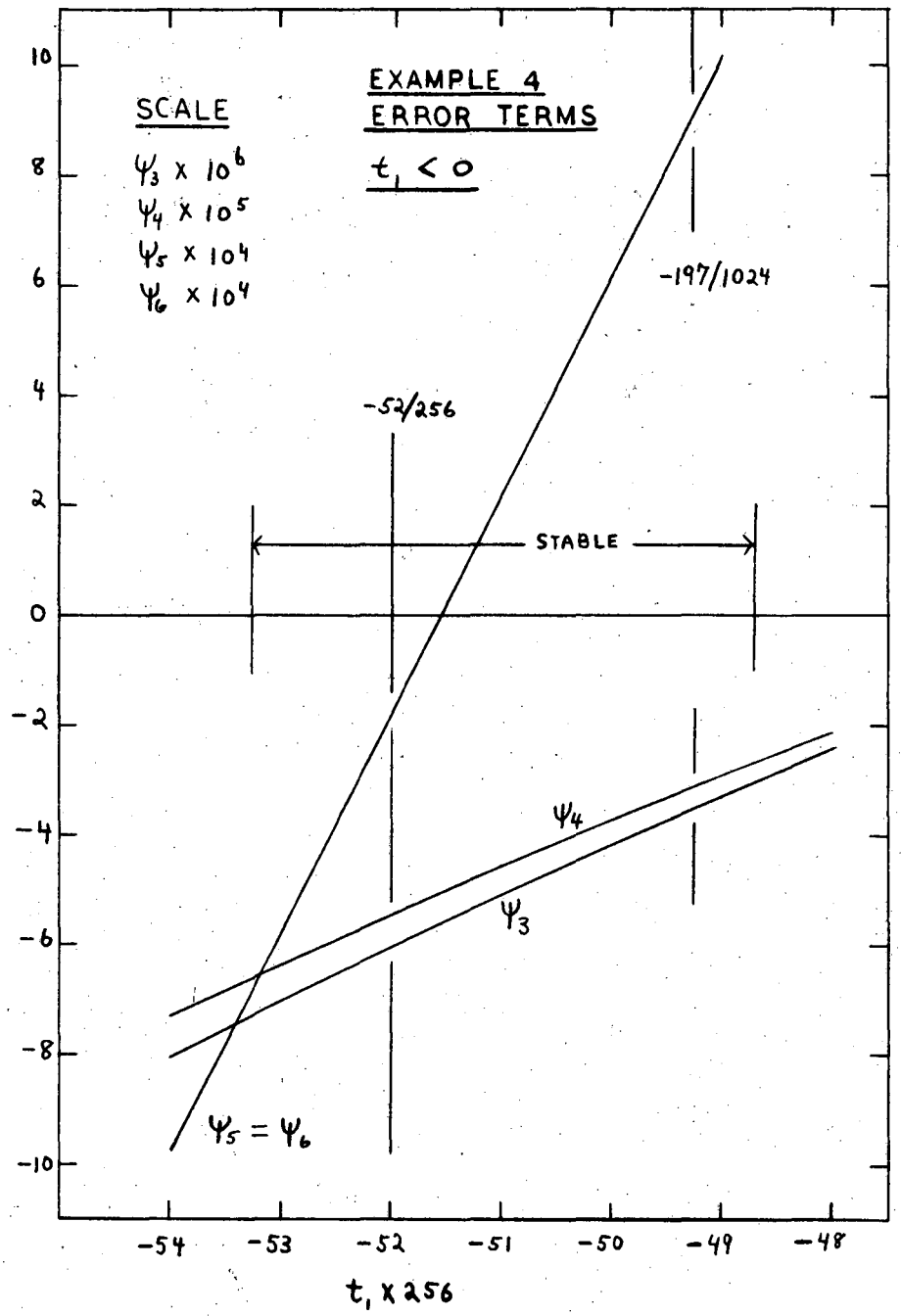
MAXIMUM ROOT





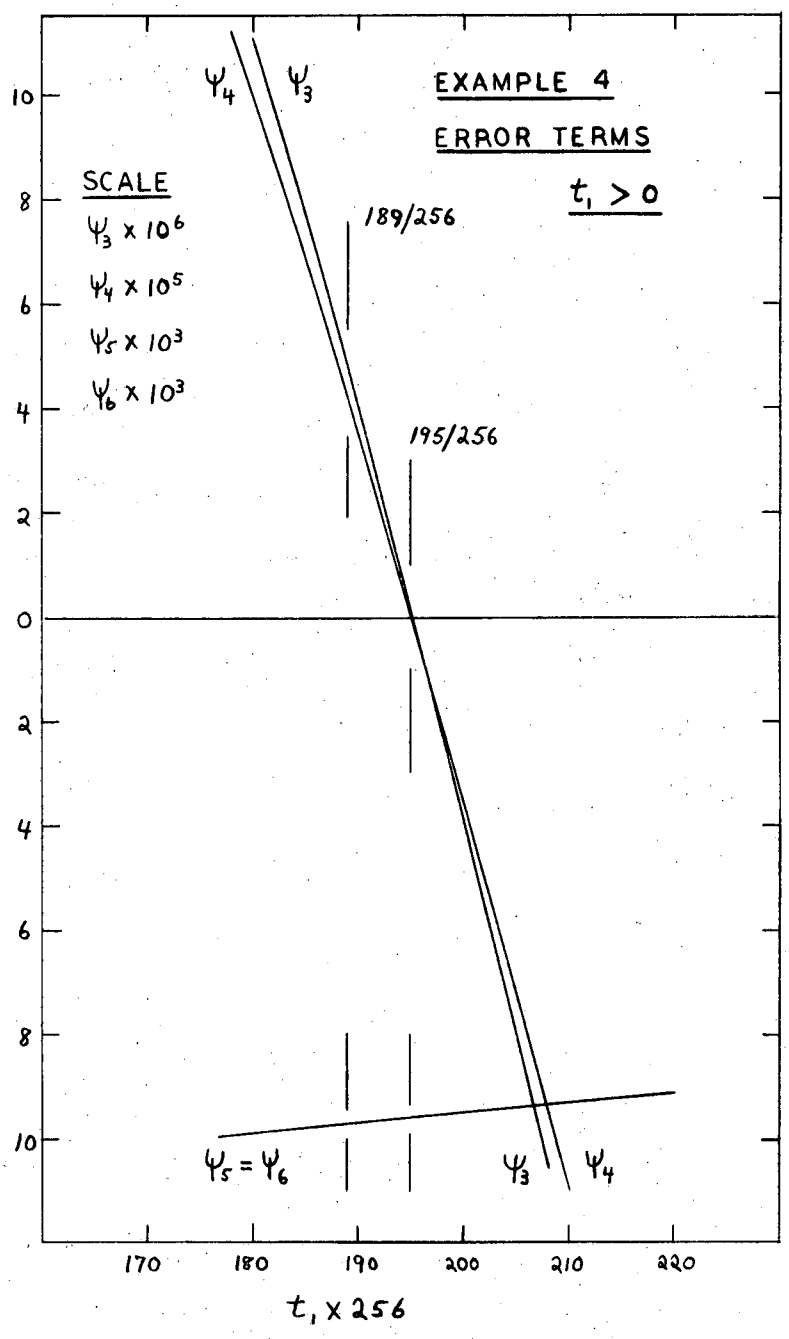
XBL 6810-6077

FIGURE 9



XBL 6810-6078

FIGURE 10



XBL 6810-6079

Appendix IV

USE OF PROCEDURE RKMI

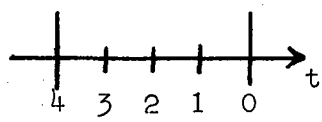
In this appendix, we give information that will enable one to use the procedure RKMI. It is assumed that the reader is familiar with the work of Chapter III and that he thus understands, in principle, how a problem is to be set up. In order to help clarify and fix these ideas, we present here one complete problem work sheet showing how the RK4 classic example was set up, along with a discussion of the program control sections. The reader can find the actual input data in Appendix III where that example appears in full detail.

We also present two source listings that are sufficient to run the program. Source Listing 1 gives the input-output structure of the main program, while Source Listing 2 gives the input-output structure of the data table input procedure data. These two same listings are all that are necessary to run RKMI once an example is understood. Since such is the case and since one usually prefers not to have an excess of information around when preparing data, we shall try and limit our discussion here to the essentials. Remember that there are complete examples given in Appendix III; there are descriptions of examples in Chapter VI; there are descriptions of various parts of the program in Chapter V; there is a description of the problem being solved in Chapter III; and, if disastrous situations arise, there are source listings in Appendix II.

We give first a sample problem set up using the rank 4 Runge-Kutta scheme. We note that we have been running under the CDC Scope System[9] using their ALGOL Compiler.[10] Thus, all the data that we present is exactly that, a data record in the job set up and constitutes the data

to the ALGOL program.

Example RK4



Problem $D^p x = X \circ \xi$

Scheme

$$\eta_0 = X(\xi_4); \quad \xi_3 = \xi_4 + \eta_0$$
$$\eta_1 = X(\xi_3); \quad \xi_2 = \xi_4 + \eta_0 + \eta_1$$
$$\eta_2 = X(\xi_2); \quad \xi_1 = \xi_4 + \eta_0 + \eta_1 + \eta_2$$
$$\eta_3 = X(\xi_1); \quad \xi_0 = \xi_4 + \eta_0 + \eta_1 + \eta_2 + \eta_3$$

The scheme given above is that which one would use to obtain a solution to the differential equation. It is better, however, to generate the scheme parameter equations by doing all the substitutions first; however, we emphasize that this is not necessary and the interested reader can set up the above scheme. The equations obtained will look different from our example, but the results are the same. We thus use instead the

Scheme

$$\eta_0 = X(\xi_4); \ \eta_1 = X(\xi_3); \ \eta_2 = X(\xi_2); \ \eta_3 = X(\xi_1);$$
$$\xi_3 = \xi_4 + \eta_0$$
$$\xi_2 = \xi_4 + \eta_0 + \eta_1$$
$$\xi_1 = \xi_4 + \eta_0 + \eta_1 + \eta_2$$
$$\xi_0 = \xi_4 + \eta_0 + \eta_1 + \eta_2 + \eta_3$$

rank = 4 principal error $\tau = O(h^5, h^5, \ldots, h^{p+3})$

period = 1

extent = 1

$\ell = 1$

We can match h^4 for the case $p = 1$ and since we wish to have the principal error term, we chose not three, but instead four derivatives. Thus, for

RKMI, we have the following parameter values:

order = p = 1	e = extent = 1
upper = ℓ = 1	period = 1
q = rank = 4	im [0] = 16
	im [1] = 4

Using field free format[10], this translates directly to the following data input: A listing of this data is given in Appendix III.

```
1,                                  } Value of control
1,1,1,1,1,3,3,3,1,2,1,1,0,          } A choice of FORTRAN equations
2,                                  }
50,70,6000,7000,10,10,10            } Computer parameters are set
1,1,4,1,1,16,4,                     } Problem parameters set
false,                              } No input-output desired

                                    } Insert data table (im[1],im[0],[k],+h,ℓ)
                                    } which is table (4,16,0,+,1)

comment  This is the classical Runge-Kutta method of rank 4;
0,-1                                } Use an expansion about the local origin and
                                    } translate to the origin
1,true,4,  1,true,3,  1,true,2, 1,true,1,   }
3,3,2,     0,4,1,0,                         }
3,2,3,     0,4,1,0,1,1,                     } Scheme
3,1,4,     0,4,1,0,1,1,1,2,                 }
3,0,5,     0,4,1,0,1,1,1,2,1,3,             }

4,-1,                               } Print results
-4,false,                           } Stop without a list dump
```

The data input to RKMI divides itself naturally into four separate blocks of information.

1. Data input that sets computer variables.

A look at Source Listing 1 will show that these parameters are almost self-explanatory. If a fuller explanation is desired, see the variable list given in Appendix II. It is necessary to have the values of print length readily available so they are given here. The subscripted variable print length [i] gives the length in characters of the character with name i for a given value of type set. Table 1 gives the appropriate values.

Table 1. Values of Print Length

	Type Set		Value of Print Length [i]											
Character		Capital Letter	[	]	(	)	u-u	u+u	u×u	/	↑	:=	,	;
i		0	1	2	3	4	5	6	7	8	9	10	11	12
See the note below.	1	2	2	2	1	1	3	3	3	1	7	2	1	2
	2	1	1	1	1	1	3	3	3	1	2	1	1	0
	3	2	0	0	1	1	3	3	3	1	7	2	0	2
	4	1	0	0	1	1	3	3	3	1	2	1	0	0

Note: Type Set = 1 CDC ALGOL subscripted variable output.

Type Set = 2 FORTRAN subscripted variable output.

Type Set = 3 CDC ALGOL simple variable output.

Type Set = 4 FORTRAN simple variable output.

2. Data input that particularizes the problem.

At the label - define problem - are input, the variables that characterize the differential equation, and the scheme. These include the scheme interval parameters of Chapter III and their meaning is

discussed there. With regard to upper = ℓ, we note that it is assumed that $\xi_i = u(\theta_i) + \Sigma\, \alpha_i A_i$ is an approximation and that $\xi(\theta_i) = u(\theta_i) + v(\theta_i)$ is the true solution. The quantity ℓ gives the first term of v since $v_L[m] = D^{p+\ell}x(0)\,\frac{I^{k+\ell+1}}{(k+\ell+1)!} + \ldots$. This in turn gives the lower bound for the order of accuracy of all approximations and we indicate this by stating that $D^{p+\ell}x(0) = D\uparrow$ (order + upper) x(0) is the first neglected Taylor term.

The number of basic functions and the number of derivatives we attempt to match must be consistent and will come from the tables of Appendix I. We will have as many derivatives as we have orders taken from these tables.

3. Data table input using procedure data

At the label - again1 - we incounter the data input procedure data. We note that the only way tables can be read in, using this procedure, is to pass through the statement last data:=0. Thus, again 1 furnishes an entry point that allows all the names and all the lists, except the data table lists, to be erased.

The tables are characterized by the parameters (im[1], im[0], $k \in \overline{P} \subset P = \{0, \ldots, p-1\}$, $\pm$, ℓ) where $\overline{P}$ is the set of k values that appear explicitly in $X \circ \xi$, ℓ is the value of upper and is a lower bound for the order of accuracy for the minor points, and the choice of + indicates that these tables are written for a forward translation; that is, mode = -1 and - for a backward translation with mode = 0. Note the program determines internally the correct coefficient for + or - translation and either table can be used. This, however, was not always the case and to remain compatible with previously verified tables, the classification is still retained. Which table is used depends on the

problem. For example, the RK4 case requires table (4,16,0,+,1).

In order to make the use of procedure data as easy as possible, we have given in this appendix a schematic source listing of data. Once this listing is understood with the aid of the explanation given below, this source listing should suffice for the inputing of tables to data.

We first note that all input-output can be suppressed from data by setting inout to false before entering data. Whether or not the internal data list is output is controlled by reading a Boolean variable in table, true means that this interval list will be printed out. Since we are inputing the various harmonics tabulated in Appendix I, we shall use these tables to give a short illustrative example for each case encountered in data. The derivative, substitution, and multiplication harmonics come from Table VI of Appendix I. Let us assume im[0] = 3, ℓ = upper = 1. This means that a suitable, though not necessary, choice of im[1] is im[1] = 2.

For derivative harmonics, we have that

$$D^{p+1}x = \frac{1}{1} A[0] + \frac{0}{1} A[1] + \frac{0}{1} A[2] = ((1,1), (0,1), (0,1))$$

$$D^{p+2}x = \frac{0}{1} A[0] + \frac{1}{1} A[1] + \frac{1}{1} A[2] = ((0,1), (1,1), (1,1))$$

where we have given above both the expansion and the list representation.

We recall that for derivative and translation harmonics, these quantities are stored as a normal form list. Data requires two additional pieces of information.

i) How many sons are input

ii) How many atoms are in each son.

The above harmonics would appear as the following data

```
0,                                  }   These are derivative harmonics
1,2,1,1,0,0,false                   }   D^{p+1}
0,  1,2,1,1  1,2,1,1,false          }   D^{p+2}                    (1)
-1,                                 }   No more data input
```

where the data lists are not output since the Boolean is false. Note that for each A[i] there is a son with atoms, zero is input as a zero son; that is, a nil list. For each derivative, there is a separate list terminated by the Boolean true or false depending on whether that data list is or is not to be output.

For substitution and multiplication tables, the list structure is as given in the explanation of Ze given in Chapter VI; that is,

(son 1, son 2, ..., son i, ..., son n) =

((numerator, denominator), (exponent), (der_1, n_1, ..., der_j, n_j, ... der_{max_1}, n_{max_1}), ... (der_i, n_i, ..., der_{max_i}, n_{max_i}),

... (der, n, ..., der_{max}, n_{max})).

The reader should be familiar with the description of Ze and also with Definition 7, Chapter III. The values of the atoms are obtained from Table IV of Appendix I. The number of sons is two more than the number of terms in the sum giving the table entry. Note that for substitution, the number of sons is always three; whereas for multiplication, the higher degree terms, in general, have more. The substitution table entries for our example are $\frac{1}{1}\theta^1$, $\frac{1}{2}\theta^2$, $\frac{1}{1}\alpha_{0,0}$. Notice that this table has $\alpha_{der,n}$. This becomes

```
2,
1,  2,2,1,1,1,1,  2,1,2,1,2,  3,2,1,1,1,0,2,0,0,false              (2)
-1,
```

Note that only the exponent is entered for θ_i^j. The program internally uses the correct θ_i. Also, that a missing θ is input as $\theta^0 = 1$, that is the exponent is 0.

The corresponding multiplication table input is

```
2,
2,  0,0,  3,2,1,1,1,1,2,0,0,false                    (3)
-1,
```

As was previously the case, a missing (zero) quantity is input with 0 sons.

The translation table comes from Table VII of Appendix I. We have that

$$\left.\begin{aligned} A(-h)[0] &= 1A(0)[0] - h\,A(0)[1] - h\,A(0)[2] \\ A(-h)[1] &= 0A(0)[0] + 1\,A(0)[1] + 0\,A(0)[2] \\ A(-h)[2] &= 0A(0)[0] + 0\,A(0)[1] + 1\,A(0)[2] \end{aligned}\right\} \begin{array}{l}\text{Type = 0} \\ \text{+h} \\ \text{mode = -1}\end{array}$$

which is input as

```
1,0,
1,2,1,1,  1,4,-1,1,q,1,  1,4,-1,1,q,1,false
                                                     (4)
0,  1,2,1,1,  0,false,
0,  0,  1,2,1,1,false,
-1,
```

where q can be any value. The program internally uses the rank of the scheme.

The data separately input in (1),(2),(3),(4) can be combined by omitting the -1 from (1),(2),(3). It is suggested that the reader use the above simple examples as a guide to understanding the data input of

the examples given in full detail in Appendix III.

4. Scheme Definition and Program Control.

The rest of the data input used will depend on the values of control read, and these, in turn, will depend on what the users wish to generate as output.

The action of the program for various values of control should be obvious from Source Listing 1 of this appendix. We give below some comments on the section:

a) Control = 0

A selection is made on whether we use forward or backward translation. Which is used is up to the user; the translation tables are internally adjusted. This section must be entered at least once before creating a scheme. A mixture of forward and backward translation is not recommended; it will lead to confusion.

b) Control = 1

The reader should be familiar with Definition 10, Chapter III. In this section, the approximators η are obtained either as a substitution $\eta = X(\xi_i)$ or by multiplying the matrix $D_L X(\xi_i)$ times some previously created sum S. The user first makes the choice of substitution which is true or derivative multiplication which is false. He then chooses the name $\langle i \rangle$ of the vector ξ_i at which evaluation takes place and if a multiplication is to be performed, then the name $\langle j \rangle$ of the sum S[j]. Since the sum can be stored either permanently or temporarily, it is necessary to specify this; true means it was stored in temporary storage, false that it was

stored permanently. Note that if the specified approximation does not exist as an approximation of the scheme, then it is created and translated. This means that temporary storage may be written on starting from its origin tempØ. This will, in effect, destroy a temporary sum S_j. This, however, is no real problem since any ξ_i, $i \in M = \{0, \ldots, e \times q\}$ can be created by performing a substitution $X(\xi_i)$ and the corresponding η need never be used.

As the new approximators and sums are constructed, they are sequentially numbered starting with 0.

As a simple example, we have

i) $\eta_o = X(\xi_2)$ 1,<u>true</u>,2

ii) $\eta_o = DX(\xi_2) \times S_3$ 1,<u>false</u>,2,3,<u>false</u>

where we assumed S_3 is stored permanently.

c. Control = 2

It should be kept in mind that the sums S constructed here are the sums of Definition 10, Chapter III, equation (94). The assignment temp:=<u>true</u> means that the presently constructed sum will be stored in temporary storage and will thus be available only as long as this storage is not reused. The value temp:=<u>false</u> will cause a permanent storage of S. The assignment linear comb:=<u>false</u> means that the coefficients of the sum are the identity; that is, a linear combination is not performed and new parameters are <u>not</u> introduced into the scheme. In particular cases, it may not make sense to take a linear combination; in these cases, a straight sum can be performed.

To actually carry out the summation, we must specify the

vectors to be summed and their type; that is, are they approximations (type = 0) or approximators (type = 1). We illustrate a simple case as

$$S = \xi_0 + \xi_1 + \eta_0 + \eta_2$$

2, false, true, 4, 0, 0, 0, 1, 1, 0, 1, 2,

where a linear combination of four terms has been constructed and permanently stored. What order the terms appear in the sum is of no consequence.

d. Control = 3

Here a new approximation ξ_i is constructed. We recall from Chapter III that ξ_i is itself a sum of the same type as S; however, we attach a special significance to this sum by interpreting it as an approximation to $\xi(t_i)$. This is reflected here in that all ξ_i are stored permanently. Upon specifying the name $i \in M = \{0, \ldots, \epsilon\chi_1\}$ of the approximation being constructed, we simply transfer to the lable new E and form the sum.

Therefore, the input to construct a new ξ_i is almost identical to that of constructing a sum, the difference being that the two Booleans are missing and we must name the ξ_i. As an example, we have

$$\xi_4 = \xi_0 + \eta_0 + \eta_1 + \eta_2$$

3, 4, 4, 0, 0, 1, 0, 1, 1, 1, 2

where the approximators η_0 are assumed already constructed.

It is worth noting that we have been purposefully leaving

out the coefficient matrices. This is quite natural since RKMI will take care of this for us. We need simply tell it what to do with the elements that constitute the scheme.

e. Control = 4

It is here that we ask that all parameter defining equations be printed. Note that the previous input has characterized the problem and defined a scheme for its solution. Once we have defined the scheme, we proceed to this section to obtain the equations. In the layout of the scheme it has always been implicitly assumed that $\xi_o \approx \xi(t_o)$ has been constructed. In general, it is assumed that it is ξ_o that we wish to have as our next value, so ξ_o is compared with $\xi(t_o)$. We have schematically laid out the interval as if t_o were the point furthest to the right, but this is only to have a diagram to talk about. Thus, by an appropriate choice of the value of the interval parameters, the approximation ξ_o can be interpreted as belonging to any value of t.

At the present, we do, however, require that the origin be located at $t_o = 0$. This pertains to the equations representing the harmonics of ξ_o. Note that if ξ_o has never been constructed, by accident or intent, all zeros will be printed since the lists representing the harmonics of ξ_o are nil.

We also implicitly assume that for any ξ_i that was created with undetermined parameters there is also available a constructed representation of either ξ_i or ξ[index(i)]. This is necessary so that there will be equations that define the undetermined parameters.

Once this section is entered, there are printed the parameter defining equations caused by requiring that in

$$\xi_o - \xi(t_o) = \sum_{i=o}^{im[0]-1} (\gamma_i - \beta_i)A_i = \sum_{i=o}^{im[0]-1} c_i A_i$$

the coefficents $c_i = 0$. The number of these equations that can actually be satisfied determines the order of accuracy if ξ_o. Next are printed equations that define the undetermined parameters that were used in the expansion of ξ_i for those ξ_i that had such implicit representation.

There is also in here a section that allows what can be considered a "manual" control of the printing of the equations. We can equate any Z[type, ..., i, ...] with any other Z[type, ..., j, ...] where type is not necessarily the same in both cases. Since any unconstructed quantity has a nil list, we can print out any of the ξ, η, S by equating to an unconstructed quantity, or we can, if it is desired, actually equate any two quantities in the form A - B = 0. These features allow a great flexibility in creating the scheme and also are helpful in constructing special schemes used in checking the tables and internal workings of RKMI.

This essentially completes the discussion of using RKMI. The program is quite flexible and the user will have to experiment with it to have a feeling for what will happen under certain conditions. A word of caution; it is <u>not</u> foolproof; or the contrary, it is easy to obtain meaningless results. A method of checking the results should always be devised.

Schematic Source Listing I

RKM1 program control source listing 8090 68 9/20/68

```
begin
begin comment The following is an outline of the program input.  All input
              is presented here.
begin:  if ioi('control') < 1 then go to end of conputations else
        s('data,input,that,sets,computer,variables');
        for i:= 0 step 1 until 12 do print length[i]:= ioi('');
        type set:= ioi('type,set');
        n for print:= ioi('n,for,print');
        line length:= ioi('line,length');
        temp0:= ioi('origin,of,temporary,store');
        list length:= ioi('list,length');
        height:= ioi('maximum,number,of,vectors,N,or,sums,S');
        height1:= ioi('maximum,length,of,a,sum');
        pmax:= ioi('maximum,length,of,a,list,product');
define problem:
        s('data,input,that,particularizes,the,problem');
        order:= ioi('order,of,the,differential,equation');
        upper:= ioi('D^(order + upper)x,is,the,first,neglected,Taylor,
                     term,,upper');
        q:= ioi('number,of,points,in,one,h,interval');
        e:= ioi('number,of,h,intervals');
        period:= ioi('period,of,the,scheme');
        im[0]:= ioi('number,of,basic,functions');
        im[1]:= ioi('number,of,derivatives,we,attemp,to,match');
        inout:= iob('input,output');
```

Source Listing I Cont.

```
again1: comment The list storage V and all names T,E,Z are initialized
                to nil. The list origin in the storage array V starts
                at last:= last data. Thus all generating tables still
                exist if entry is made through the lable again1;
        if last data = 0 then data; scheme; title;
        comment The procedure data inputs generating tables. See source
                listing II of this appendix to effect this input. The
                procedure scheme ontputs the scheme definition if print
                scheme is true. The procedure title reads an Algol
                comment <text>, thus furnishing a means of identifying
                the output;
again:  control:= ioi('control'); if control < -1 then go to fin;
        if control = -1 then begin print scheme:= true; go to again1 end;
        if control = 0 then
        begin comment Define the undetermined parameters B, if mode = 0 then
                      they are defined in expansions about the origin 0
                      else if mode = -1 then they are defined in expan-
                      sions about the local origin -i1×h where i1:= i÷q,
                      with i the name of the approximation and q the rank
                      of the scheme;
              mode:= ioi('mode,of,memory');
        end else
        if control = 1 then
        begin comment Construct a new approximator B[no3[1]], if substitution
                      then by substitution N[i]:= X(E[j]) of an approximation
                      E[j], j in M = (0,...,e×q) else by multiplication
                      using the Jacobian matrix, J[j]:= DX(E[j]) to create
```

Source Listing I Cont.

```
                N[i]:= J[j] × S[num[2]]. Note that if E[i] = nil
                then E[i] is created with undetermined parameters,
                if temporary then the sum S is stored in temporary
                storage else it is stored in perminant storage. In
                any case, it is assumed that a constructed representa-
                tion of S exists;
            if iob('substitution') then num[0]:= ioi('E[i],i');
            else begin num[0]:= ioi('DX(E[i]),i');
                       num[2]:= ioi('S[j],j');
                       iob('temporary,sum')
                 end
        end else
        if control = 2 then
        begin comment Create a new sum S[no3[2]] consisting of approximations
            E[i], i in M = (0,...,exq), and of existing approximators
            N[j], if E[j] does not exist then it is created with un-
            determined parameters. The order of the input of the
            constituents of the sum is immaterial, they are internally
            normalized to
            S[no3[i]]:= sum(i,0,length1-1,sum(k,order-1,0,B[...] ×
            E[vs[0,i],k])) + sum(i,0,length2-1,B[...] × N[v[1,i]]) +
            sum(i,length2,length-length1-1,B[...] × N[v[1,i]]) where
            (v[0,i] < v[0,j] and v[1,i] < v[1,j]) if i < j. The first
            sum is the sum of approximations E, the second sum is the sum
            of approximators N that have been formed by means of a
            substitution X o E, the third sum is a sum of approximators
```

Source Listing I Cont.

```
                that have been formed by multiplication of the form
                (DX o E) × S.  Any of the sums may be empty;
                s('new,sum');
                temp:= iob('temporary,store');
                linear comb:= iob('linear,combination');
new E:          length:= ioi('length,of,sum');
                for i:= 0 step 1 until length-1 do
                begin type:= ioi('type,of,vector');
                      comment if type = 0 then E else
                              if type = 1 then N;
                      vs[type,num[type]]:= ioi('num[type]')
                end
          end else
          if control = 3 then
          begin comment Construct a new approximation E[i] by specifying i
                        and the vectors that make up the sum,
                        E[i]:= sum(i,0,length1-1,sum(k,order-1,0,B[...] ×
                                   E[vs[0,i],k])) +
                               sum(i,0,length-length1-1,B[...] × N[vs[1,i]])));
                no3[0]:= ioi('E[i],i'); go to new E
          end else
          if control = 4 then
          begin comment Print all parameter defining equations;
                comment Points that are to be equated.  If we input a sequence
                of points a, b, c, d, e, f, ... then the output is a = b,
                c = d, e = f, ... .  Therefore, the number of points must be
```

Source Listing I Cont.

an even number. The quantity printed is the list Z[type, ...,name,...]. Note that names are given sequentially to approximators N and sums S as they are constructed. The names of E lie in the set M = (0,...,exq). A quantity may be printed even though it has never been constructed. This section below furnishes a means by which the user can manipulate the equating of constructed quantities. Note the following short table:

type	quantity
-1	E expanded about the local origin
0	E expanded about the origin
1	N expanded about the origin
2	S expanded about the origin;

```
        n:= ioi('number,of,points');
        if n > 0 then
        for i:= 0 step 1 until n - 1 do
        begin vs[0,i]:= ioi('name');
              vs[0,i]:= ioi('type')
        end
    end of control;
    go to again;
fin: comment A short dump of the list pointers is given here;
    if iob('check,list') then check list(0,last);
    go to if control = -2 then define problem else
```

Source Listing I Cont.

```
                if control = -3 then begin else
                end of conputations
end of program;
end of computations; s('end,of,computations')
end
```

Schematic Source Listing II

Table Input Using The Procedures data and table

```
procedure data;
begin
again:  control:= ioi('control');
        if control = 0 then
        begin sr('harmonics,of,derivatives');
              for i:= 0 step 1 until im[1] - 1 do
              table(no2,im[0],D[i,no2])
        end else
        if control = 1 then
        begin type:= ioi('type'); sr('translation table');
              if type = 0 then
              for i:= 0 step 1 until im[0] - 1 do
              table(no2,im[0],a[type,i,no2])
        end else
        if control = 2 then
        begin type:= ioi('type');
              if type = 1 then sr('substitution,table') else
              if type = 2 then sr('multiplication,table');
              table(no2,im[0],W[type,no2])
        end; if control = -1 then go to again
end data;
```

Source Listing II Cont.

```
procedure table(no2,im[0],name);
begin for no2:= 0 step 1 until im[0] - 1 do
      begin i2:= ioi('number,of,son]);
            if i2 ≠ 0 then
            for no1:= 0 step 1 until i2 - 1 do
            begin i1:= ioi('number,of,atoms');
                  for no:= 0 step 1 until i1 - 1 do
                  ioi('atom')
            end no1
      end no2;
      iob('data,list,out')
end table;
```

Appendix V

NOTATION

The notation that is used throughout this work is standard and is explained as it is introduced. We do, however, use a rather concise representation for the Taylor's series which we shall explain more fully below.

We first note that we consistently use D as the derivative with respect to the single variable $t \in R$ the real line, and $D_N[i]$ as a partial derivative. We also sum on repeated index sets. Thus, we write for $X \in R^m \to R^n$, $u, v \in R \to R^m$

$$X \circ (u + v) = X \circ u + \sum_{s=1}^{\infty} \frac{1}{s!} (D_{L_1 \ldots L_s} X \circ u) v_{L_1} \ldots v_{L_s}$$

which when written in detail becomes

$$X(u(t) + v(t)) = X(u(t)) + \sum_{s=1}^{\infty} \frac{1}{s!} \sum_{m_s \in L_1} \cdots \sum_{m_s \in L_s} D_{L_1 \ldots L_s}[m_1, \ldots, m_s] X(u(t)) \, v_{L_1}(t)[m_1] \ldots v_{L_s}(t)[m_s]$$

where

$$D_{L_1 \ldots L_s}[m_1 \ldots, m_s] X = \frac{\partial^s X}{\partial x_{m_1} \ldots \partial x_{m_s}}$$

and

$$L_i = \{1, \ldots, m\}, \quad i = 1, 2, \ldots, s.$$

In Chapter II, we have $m = n$ and elsewhere we have $m = n \times \mathfrak{d}$.

Throughout this work, we have repeatedly and consistently used

capital subscripts to indicate the index sets (the domains) associated with the quantities. We have also consistently summed on repeated index sets. The style in which this notation has been used along with the use of the matrices $I^{(k)}_{IN}$, which are first used in Chapter III, was introduced to the author by Professor R. DeVogelaere[3]. The use of this notation has proved invaluable in developing this work.

REFERENCES

1. J. C. Butcher, Coefficients for the Study of Runge-Kutta Integration Processes, J. Australian Math. Soc. 3, Part 2, (1963), 185-201.

2. F. Ceschino and J. Kuntzmann, Numerical Solution of Initial Value Problems, (Prentice Hall, Inc., Englewood Cliffs, N. J., 1966).

and

F. Ceschino and J. Kuntzmann, Problèmes Differentiels de Conditions Initiales, (Dunod, Paris, 1963).

3. Professor R. DeVogelaere, Mathematics Dept., Univ. of Calif., Berkeley, Calif., private communication.

4. J. C. Butcher, A Modified Multistep Method for the Numerical Integration of Ordinary Differentials, J. ACM 12, 1 (1963), 124-135.

5. R. DeVogelaere, A Method for the Numerical Integration of Differential Equations of Second Order Without Explicit First Derivatives, J. Research Natl. Bur. Standards 54, 3 (March 1955), 119-125.

6 R. E. Scraton, The Numerical Solution of Second Order Differential Equations, The Computer J. 6, 4, (January, 1964), 368-370.

7. T. Frey, On Improvement of the Runge-Kutta-Nyström Method I, Periodica Polytechnica, Electrical Engineering - Elektrotechnik 2, No. 2, Budapest (1958), 141-165.

8. J. C. Butcher, A Multistep Generalization of Runge-Kutta Methods with Four or Five Stages, J. ACM 14, 1, (January 1967), 84-99.

REFERENCES - contd.

9. Control Data 6400/6500/6600 Computer Systems, Scope 3.1 Reference Manual, Rev. A, Pub. No. 60189400A, Control Data Corporation, Palo Alto, California.

10. ALGOL Generic Reference Manual, Pub. No. 60214900, Control Data Corp. (December 1967), Palo Alto, California.

TECHNICAL INFORMATION DIVISION
LAWRENCE RADIATION LABORATORY
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720