COMPILER SOLUTION OF DIFFERENTIAL EQUATIONS
WITH DIFFERENTIAL ANALYZER-TYPE OUTPUT

By

Lorn Lambier Howard

AN ABSTRACT

Submitted to the School for Advanced Graduate Studies of
Michigan State University of Agriculture and
Applied Science in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

Department of Electrical Engineering

1959

Approved L.W. Von Tersch

Pa

Pag

Pag

Page

Page

Page

CORRECTIONS

1959 Doctoral Thesis in Electrical Engineering: Compiler Solution of Differential Equations with Differential Analyzer-Type Output, Lorn L. Howard

Page 51 - correct order pair 330 as follows:

```
330   40  531F
      26  331F
```

Page 58 - correct order pair 444 as follows:

```
444   L5  592F
      L0  493F
```

Page 72 - next to last line (part 2) - delete the first sentence and the sentence in parenthesis immediately following it. Substitute therefor the following:

"Divide all the terms by the coefficient of the derivative of highest order. (If this yields coefficients whose values are larger than 10,000 the original differential equation must be scaled until the coefficients at this stage are below 10,000 - otherwise the problem will not go into the computer.)"

Page 75 - correct the first two terms in equation (1) to read:

$$0.05\frac{d^4y}{dt^4} + \frac{d^2y}{dt^2}$$

Page 75 - correct equation (2a) to read:

$$\frac{d^4y}{dt^4} + 20\frac{d^2y}{dt^2} - 3900\frac{dy}{dt} + 200y = 2000\sin 2t + 20t^2 + 40$$

Page 76 - correct equation (2b) to read:

$$0.0001\frac{d^4y}{dt^4} + 0.002\frac{d^2y}{dt^2} - 0.39\frac{dy}{dt} + 0.02y = 0.2\sin 2t + 0.002t^2 + 0.004$$

Page 76 - correct equation (3) to read:

```
+0N5  +0001N4  +0N3  +002N2  -39N1  +02N0 =  +2FSIN+0002NT

+002FT2  +004FK/IC  +0  +0  +000001  +000001  +0N/INCR  +1N
```

I am enclosing three copies of corrections which should be made to my 1959 Doctoral Thesis in Electrical Engineering.

Lorn L. Howard

May 3, 1965

Lorn L. Howard

ABSTRACT

Differential analyzer-type output is available from a digital computer using the techniques described in this paper. In addition, this is made possible in such a way that anyone who needs the solution to an ordinary linear constant coefficient differential equation may obtain it without assistance from programmers or previous knowledge of the operation or programming of either type of computer. The user needs only to convert his differential equation directly into a simple code resembling the actual mathematical statement of the equation, punch this code onto computer tape preceded by the compiler routine developed in this paper, and have results immediately after feeding the tape to the computer. The entire process should require at most only a few minutes.

As with the differential analyzer, the output is a simultaneous presentation of the dependent variable and all of its derivatives as a function of time. A major difference, however, is in the greatly improved accuracy of the results over those available from that type of computer. Another desirable feature, of course, is in the large reduction in the time required to obtain the results.

Both the solution and the differential analyzer-type of output are accomplished without the necessity for the reduction of the differential equation to a series of first-order equations, a procedure which is often required. The standard Runge-Kutta integration procedure is used.

The compiler routine developed herein is prepared especially for

the Michigan State University automatic digital computer (MISTIC) but may be used readily where other models of this type of computer are available: Iowa State College , University of Illinois, University of Sydney, Aberdeen Proving Ground. The programming technique, however, is laid out in detail so that the method may be readily adapted to programming for other types of digital computers.

Availability of storage space (1024 positions) limits use of the program to the solution of equations of first through fifth order. A wide variety of combinations of "driving functions" is allowed, however. Provision is made so that experienced programmers may readily modify the routine to add other driving functions as required.

COMPILER SOLUTION OF DIFFERENTIAL EQUATIONS WITH DIFFERENTIAL ANALYZER-TYPE OUTPUT

By

Lorn Lambier Howard

A THESIS

Submitted to the School for Advanced Graduate Studies of
Michigan State University of Agriculture and
Applied Science in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

Department of Electrical Engineering

Approved_______________________________________

G15149
6/19/61

ACKNOWLEDGMENTS

The author wishes, first of all, to thank especially Professor Myril B. Reed for his continual encouragement and understanding throughout the many months of preparation for this thesis.

Professor Lawrence W. VonTersch suggested the problem of programming the digital computer to provide differential analyzer-type output, and his helpfulness is appreciated very much, also.

The author is also very grateful to Professor Gerard P. Weeg for his thoughtfulness, helpful advice, and for his delightful sense of humor.

A graduate student is indebted to many of his contemporaries and to a large number of faculty--both in and out of his field--for his professional outlook and scientific attitude. Dr. Frederick D. Bennett, Professor George B. Clark, Dr. Carl E. Drumheller, and Professor William J. Fry, graduate teachers and advisors, have contributed much in these areas, and contact with them has been a great source of satisfaction.

Finally, the efforts of the author would have fallen far short of this work had it not been for the endless devotion, care, and assistance from his wife, Etha. Also, her many capabilities in typing, tape preparation, and in making lengthy calculations for verifying results have sped the completion time immeasurably.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

I. INTRODUCTION

It has been possible to obtain the solution to differential equations with the aid of electronic equipment ever since the development of the first all-electronic type digital computer, ENIAC, at the University of Pennsylvania around 1942 (1). About five years later, another electronic device became available for this purpose: the electronic differential analyzer or analog computer (2).

The differential analyzer is frequently found to be faster, more convenient, and more satisfactory in many problems, but the need is often felt for an accuracy and a kind of flexibility obtainable only on the digital machine. This has inspired considerable effort toward the production of either a machine or machine-program which would combine the advantages of both types of computer.

The digital differential analyzer was one of the earliest of the "machine" efforts. It was developed by a group of engineers from the Northrop Aircraft Corporation (3), and was first discussed by Sprague (4) in 1952. This digital-type computer is composed mainly of a set of units which perform an integrating function. These units are analogous to the integrators in the typical electronic analog machine. The accuracy of this computer appears to be considerably less than that of the usual digital computer; however, it is sometimes approximately that of the ordinary differential analyzer. It is slower than the differential analyzer.

In 1955 Selfridge (5) described a system of programming a digital computer using a scheme very similar to that employed in the coding of an analog computer. His method employs a very simple integration process in which the increment consists solely of the sum of the inputs multiplied by the mesh size of the independent variable. Encoding a differential equations problem for solution in this manner on a digital computer is a great simplification; however, extremely small mesh size is required in order to obtain appreciable accuracy. This has the disadvantage of requiring much more time for the solution. Some problems do not appear to be readily adaptable to this technique.

The Selfridge method allows the use of normal digital computer coding. A different type of coding, using "pseudo-code," was developed by Lesh and Curl (6,7) in 1957 for use with their "interpretive" digital computer routine simulating differential analyzer operations. This coding depends upon an interpretive routine (previously fed to the computer) to deduce the analog computer component structure and sequence of operations from it and to produce the differential equations' solution therefrom. The system, called DEPI (differential equations pseudo-code interpreter), is an aid to users familiar with analog computer operations but who are unfamiliar with digital methods since rapid, accurate digital solutions to differential equation problems may be obtained without the necessity for learning digital techniques. In comparative performance at similar accuracies the DEPI program is eight times slower than an analog computer solution (6). Even at this speed, DEPI performance is much faster than a digital differential analyzer. At slower speeds (reduced increment size), DEPI accuracy increases to that appropriate for normal digital computer output.

Recently (1959) Stein, Rose, and Parker (8) developed for a digital computer a compiler routine (a program whose sole purpose is to assemble another program to carry out a specific function) which makes use of "analog-oriented" input information. Input to the compiler consists of the encoded description of an analog computer set-up diagram. This system differs from either of the two previous programming techniques, first, in that no effort is made to simulate the functional structure of the analog computer. Secondly, the compiler does most of the programming for the digital machine. The balance is accomplished by Fortran, an automatic coding system developed by the International Business Machines Corporation, which accepts statements resembling mathematical language. The compiler output is Fortran input, and the entire operation is handled by an IBM 704 digital computer. Common usage of the analog computer set-up as a fundamental "problem-source" led the authors to begin their programming at this point rather than at the point of mathematical description[1]. Deduction of differential equations from analog computer set-up diagrams represents work done earlier by Stein and Rose (9) and forms the basis for use of a code acceptable to the compiler. Preliminary experience in use of this compiler indicates a speed four times slower than a test analog computer on a similar problem at a comparable accuracy. This is two times as fast as the experience reported with DEPI, and such a gain in computing speed was predicted by Lesh and Curl (6).

The general purpose of the present work was to obtain a type of program for the digital computer which would enable it to yield rapid, accurate, differential analyzer-type output from extremely simple, yet very flexible input,--input which could be written readily as a mathe-

[1] Personal communication from Mr. Rose.

matical expression by users having no familiarity with either type of computer. Some conclusions were drawn from preliminary studies concerning the general direction such an effort should take, and programming was completed (within storage limits of the computer available)in fulfillment of this aim.

In particular, a compiler routine has been written for digital computers of the MISTIC type (ILLIAC, SILLIAC, and ORDVAC) to provide differential analyzer-type output from simply-encoded differential equation input. The differential equation may be of any order up to and including the fifth. One-point boundary conditions must be available for all except the highest order derivative. "Driving-functions" may consist of a constant plus any additive combination of the following functions multiplied by their respective coefficients: sin k_1t, cos k_2t, ln k_3t, e^{k_4t}, t, t^2, t^3, and $t^{1/2}$ or $t^{1/3}$ or $t^{1/4}$ or $t^{1/5}$. Each function may be used only once; however, instructions for easy modification of the compiler to add other driving functions and still remain within the storage capacity of the computer are given later. Also discussed are outlines for extension of the present routine to include simultaneous equations and equations of higher order (possible with the availability of more storage).

All previous effort to combine advantages of both types of computer has presumed knowledge of the programming of at least one of these machines. Use of the compiler routine developed herein requires no such previous knowledge, and its programming for the digital computer yields almost-simultaneous information on the independent variable together with the dependent variable and all of its derivatives.

Aside from advantages which accrue in obtaining a composite of the benefits of both types of computer, there is an economic urgency in the development of compilers which is often pointed out by Hopper (9,10, 11). This fact obtains at installations of computers of the MISTIC type mentioned previously as well as in industry. Insofar as is known to this writer, however, there has been no compiler development for solving differential equations on any of these machines, even though the physicist, chemist, engineer, or researcher there should be able to get this "bread-and-butter" job done as readily as his counterpart in industry where compilers are commonplace.

On the following pages is described the preliminary study leading to the first programming efforts, assembly of the compiler routine with a discussion of limitations, and final testing. The complete compiler routine is then given, together with instructions for its use. An example is also prepared in detail.

II. DIFFERENTIAL ANALYZER SIMULATION

Most of the attempts to simulate the differential analyzer have sought its speed, ease of programming, flexibility, and economy. The first attempts (4) aimed at duplicating the physical action in an integrating circuit by amassing a stored quantity at a programmed rate. Some ease of programming and flexibility were gained, perhaps, but at a loss of speed and accuracy for some problems. Further developments have made some improvement in these areas. Later, Lesh and Curl's interpretive routine (6) imitated only the structure of the analog program. This routine made marked progress in achieving some of each of the desirable attributes of the analog machine. Its authors pointed out, however, that the analog structure of their program appeared to be artificial and that improvement could probably be made by its elimination. They also suggested a compiler routine for increased speed, noting however, that it would be much more difficult to write and at the same time keep flexible.

It was with the development of their modification ideas in view that the present work was begun. There are several considerations which make this type of compiler seem promising. First of all, both the Selfridge routine (5) and the interpretive routine require sequential calculation. This, in itself, precludes an output speed equal to that of the analog device. Further, it appears likely that so long as digital computers are sequential devices similar to present-day types, there is little promise of completely duplicating the speed of the analog computer. The compiler-type program, however, represents an improvement over the relatively slow interpretive routine. Secondly, the other desirable characteristics principally involve the input and output of the machine, and it would seem reasonable to expect that, although the time and effort required might be appreciable, both the input and output of a digital computer could be tailored to provide much of the flexibility, ease of programming, and type of output found on the differential analyzer. And then in particular, real economy of time might be realized by Hopper's "layman" (11), or inexperienced computer user, if such a compiler were available. Finally, the solution of differential equations need not depend upon the integrating processes nor the component configuration inherent in the differential analyzer, but could be obtained more readily by using a suitably-programmed numerical method. Both the first and last of these considerations have been utilized in a recent compiler program (8,9).

The idea of simulating the differential analyzer as such then was abandoned, and in its place was planned a compiler program which would retain all the desirable features common to the analyzer as a differential equation solver except some of its speed. Even in this area, it was planned to choose and provide routines to allow as close an approach as possible to analog speed.

III. ORGANIZATION OF THE COMPILER ROUTINE

General Description

The Compiler is a complete routine in itself, designed to be put on tape and fed into the computer just ahead of a small amount of coding (also on a tape) describing the differential equation to be solved. The coding is discussed later, but it is the job of the Compiler to bring this code into the computer, to obtain, and then to output the solution to the differential equation represented thereon.

The Compiler must necessarily contain a number of subroutines designed to do specific jobs if the calculations are to be obtained efficiently. The routines are listed below in the order in which they appear on the Compiler tape. (Their memory locations are given at the end of the Compiler Routine and in the Appendix.)

1. Input the balance of the Compiler (Decimal Order Input)
2. Differential Equation (including "Driving Function" Routine)
3. Assembly
4. Fast Sine-Cosine
5. Integral Root
6. Exponential
7. Logarithm
8. Decimal Fraction Input

9. Decimal Fraction Print

This list comprises everything in the Compiler with the exception of special control orders. The complete Compiler program, except for standard library routines noted in the following discussion, is given order by order in part V.

In operation, the Decimal Order Input brings in the rest of the Compiler. Control is then transferred to the Assembly routine which proceeds to bring in the encoded differential equation. As this equation code is being brought in, the Assembly routine makes choices and sets counters to organize a program to solve the differential equation. Program control is transferred to the differential equation-solving routine at the end of this read-in.

That routine then proceeds to carry out a program to evaluate the differential equation. Control is often transferred out of the routine and into subroutines for frequently-repeated operations such as printing or punching out information or calculating the driving function. In the case of the latter process, program control frequently leaves its subroutine also to go to other subroutines such as the exponential, sine, logarithm, et cetera, finally returning to the Driving Function routine, and then later to the main routine. One increment of each of the variables after another is calculated and output. The machine will continue to run until stopped or until hang-up occurs due to overflow.

Since the computer operates with fractional quantities, the program is designed to carry out calculations at a value of the variables which is at least 0.0001 of their actual value (see part VI) in order to allow for considerable growth of the variables before overflow or hang-up. This implies that in determining the range of allowable computations the user must consider that normal unscaled values in the solution may not exceed 9,999 (when a factor of 0.0001 only is used). In fact, they must be considerably less than this if the calculation is to proceed usefully for very long. Scaling must also be considered in the use of the various driving function subroutines such as those listed (limits are discussed later on in this part).

Fixed-point programming is used throughout. This does not seem to limit seriously most problems of the usual engineering type encountered. A decision to provide floating-point programming would have allowed considerably less storage space for essential operations.

The routine to input the balance of the Compiler is the standard Decimal Order Input routine available at any of the MISTIC-type computer installations.

Differential Equation Routine

The Numerical Method

The Differential Equation routine is prepared especially for this compiler. Its purpose is to carry out the numerical solution of the differential equation using the Runge-Kutta method (four-step). It is desirable to consider some reasons for such a choice.

Numerical methods for solving differential equations on digital computers have been studied extensively since 1942 and a partial list of the work reported in the literature is given in the bibliography (13-20). A recent comprehensive study was made by Williams (20). He found that the best accuracy obtained in a comparison including several four-point methods, a series method, the Runge-Kutta-Gill technique, and the Wilf method came from use of the Runge-Kutta-Gill procedure. The price for this accuracy is a somewhat reduced speed, however. Gill himself points out that his modification of the Runge-Kutta process is slower than the original (21).

Actually, this general process (Runge-Kutta) has been chosen by several authors as the outstanding method for machine solution. The earliest seems to have been Froberg (14) in 1950. Also, it was used by Lesh and Curl (6), and by Stein, Rose and Parker (8). It is essentially a refinement of what may be called averaging methods, and has the very desirable characteristic that it requires no special formulas to get the solution started. Further, for purposes of this work, it lends itself readily to programming without the annoying necessity for reducing equations of order greater than one down to the first order. In addition, it is easy to obtain the usual values one expects to find at the output of a differential analyzer, i.e., y, $\dot{y}$, $\ddot{y}$, et cetera, in passing normally through the calculation procedure.

The Runge-Kutta method has no check on accuracy, and the error cannot be determined although it is near the order of the fifth power of the increment of the independent variable (22). Improvement in accuracy can be obtained by taking smaller increments—up to a point. Such a decrease always reduces the speed and increases the possible round-off error. Another method, such as Milne's, might be added to the Runge-Kutta method after starting in order to provide for a regular check on the accuracy. It is felt that this would require excessive storage—already in short supply for the present program—and it has not been done.

Runge-Kutta Equations

The unmodified verson of the Runge-Kutta method was chosen after preliminary testing (see part IV) demonstrated its suitability insofar as speed and accuracy were concerned. The equations (22), including all steps necessary to make calculations for two increments of all the dependent variables in an ordinary linear constant coefficient fifth order differential equation, are given on pages 13 and 14. Note that the equations for two increments are given; also, that all steps necessary for the calculation of a fifth order equation are included. The assembly routine decides the order of the equation being input and makes a choice as to whether all or part of these equations are used--depending upon the order of the equation. The major modification of the steps for an equation of order less than five requires calculating the highest derivative of the equation as a function of the other terms in the equation--in a manner similar to that shown for the fifth order equation--rather than as shown for that derivative in the chart. As for the fifth derivative, the function is evaluated using the values of the variables corresponding to the step in which the highest derivative is being evaluated.

Consider the procedure for obtaining the solution to a fifth order differential equation using the steps shown in the chart. The first line (except for the highest derivative) at the top of the two pages consists of initial conditions, and the first calculation requires the use of these in evaluating the highest derivative $\overset{.....}{y}_{11}$. The next calculation is of $\overset{....}{y}_{12}$, and then the calculating proceeds to the left until t_{12} is calculated. Following this, $\overset{.....}{y}_{12}$ is calculated using the values obtained in the calculations which proceeded leftward along the second

$t_{11} = t_o$ $t_{12} = t_{11} + \frac{\Delta t}{2}$ $t_{13} = t_{11} + \frac{\Delta t}{2}$ $t_{14} = t_{11} + \Delta t$	$y_{11} = y_o$ $y_{12} = y_{11} + \dot y_{11} \frac{\Delta t}{2}$ $y_{13} = y_{11} + \dot y_{12} \frac{\Delta t}{2}$ $y_{14} = y_{11} + \dot y_{13} \Delta t$
$\ddot y_{11} = \ddot y_o$ $\ddot y_{12} = \ddot y_{11} + \dddot y_{11} \frac{\Delta t}{2}$ $\ddot y_{13} = \ddot y_{11} + \dddot y_{12} \frac{\Delta t}{2}$ $\ddot y_{14} = \ddot y_{11} + \dddot y_{13} \Delta t$	$\dddot y_{11} = \dddot y_o$ $\dddot y_{12} = \dddot y_{11} + \ddddot y_{11} \frac{\Delta t}{2}$ $\dddot y_{13} = \dddot y_{11} + \ddddot y_{12} \frac{\Delta t}{2}$ $\dddot y_{14} = \dddot y_{11} + \ddddot y_{13} \Delta t$
$\Delta \ddot y_1 = \frac{\Delta t}{6}(\dddot y_{11} + 2 \dddot y_{12} + 2 \dddot y_{13} + \dddot y_{14})$	$\Delta \dddot y_1 = \frac{\Delta t}{6}(\ddddot y_{11} + 2 \ddddot y_{12} + 2 \ddddot y_{13} + \ddddot y_{14})$
$t_{21} = t_{11} + \Delta t$ $t_{22} = t_{21} + \frac{\Delta t}{2}$ $t_{23} = t_{21} + \frac{\Delta t}{2}$ $t_{24} = t_{21} + \Delta t$	$y_{21} = y_{11} + \Delta y_1$ $y_{22} = y_{21} + \dot y_{21} \frac{\Delta t}{2}$ $y_{23} = y_{21} + \dot y_{22} \frac{\Delta t}{2}$ $y_{24} = y_{21} + \dot y_{23} \Delta t$
$\ddot y_{21} = \ddot y_{11} + \Delta \ddot y_1$ $\ddot y_{22} = \ddot y_{21} + \dddot y_{21} \frac{\Delta t}{2}$ $\ddot y_{23} = \ddot y_{21} + \dddot y_{22} \frac{\Delta t}{2}$ $\ddot y_{24} = \ddot y_{21} + \dddot y_{23} \Delta t$	$\dddot y_{21} = \dddot y_{11} + \Delta \dddot y_1$ $\dddot y_{22} = \dddot y_{21} + \ddddot y_{21} \frac{\Delta t}{2}$ $\dddot y_{23} = \dddot y_{21} + \ddddot y_{22} \frac{\Delta t}{2}$ $\dddot y_{24} = \dddot y_{21} + \ddddot y_{23} \Delta t$
$\Delta \ddot y_2 = \frac{\Delta t}{6}(\dddot y_{21} + 2 \dddot y_{22} + 2 \dddot y_{23} + \dddot y_{24})$	$\Delta \dddot y_2 = \frac{\Delta t}{6}(\ddddot y_{21} + 2 \ddddot y_{22} + 2 \ddddot y_{23} + \ddddot y_{24})$

The Runge-Kutta Equations for Calculating the First Two Incre-

$\dot{y}_{11} = \dot{y}_0$ $\dot{y}_{12} = \dot{y}_{11} + \ddot{y}_{11}\frac{\Delta t}{2}$ $\dot{y}_{13} = \dot{y}_{11} + \ddot{y}_{12}\frac{\Delta t}{2}$ $\dot{y}_{14} = \dot{y}_{11} + \ddot{y}_{13}\,\Delta t$	$\ddot{y}_{11} = \ddot{y}_0$ $\ddot{y}_{12} = \ddot{y}_{11} + \dddot{y}_{11}\frac{\Delta t}{2}$ $\ddot{y}_{13} = \ddot{y}_{11} + \dddot{y}_{12}\frac{\Delta t}{2}$ $\ddot{y}_{14} = \ddot{y}_{11} + \dddot{y}_{13}\,\Delta t$
$\Delta \dot{y}_1 = \frac{\Delta t}{6}(\dot{y}_{11} + 2\dot{y}_{12} + 2\dot{y}_{13} + \dot{y}_{14})$	$\Delta \ddot{y}_1 = \frac{\Delta t}{6}(\ddot{y}_{11} + 2\ddot{y}_{12} + 2\ddot{y}_{13} + \ddot{y}_{14})$
$\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{11} = f(t_0, y_0, \dot{y}_0, \ddot{y}_0, \dddot{y}_0, \ddddot{y}_0)$ $\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{12} = f(t_{12}, y_{12}, \dot{y}_{12}, \ddot{y}_{12}, \dddot{y}_{12}, \ddddot{y}_{12})$ $\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{13} = f(t_{13}, y_{13}, \dot{y}_{13}, \ddot{y}_{13}, \dddot{y}_{13}, \ddddot{y}_{13})$ $\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{14} = f(t_{14}, y_{14}, \dot{y}_{14}, \ddot{y}_{14}, \dddot{y}_{14}, \ddddot{y}_{14})$	
$\Delta \overset{\cdot\cdot\cdot\cdot\cdot}{y}_1 = \frac{\Delta t}{6}(\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{11} + 2\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{12} + 2\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{13} + \overset{\cdot\cdot\cdot\cdot\cdot}{y}_{14})$	
$\dot{y}_{21} = \dot{y}_0 + \Delta \dot{y}_1$ $\dot{y}_{22} = \dot{y}_{21} + \ddot{y}_{21}\frac{\Delta t}{2}$ $\dot{y}_{23} = \dot{y}_{21} + \ddot{y}_{22}\frac{\Delta t}{2}$ $\dot{y}_{24} = \dot{y}_{21} + \ddot{y}_{23}\,\Delta t$	$\ddot{y}_{21} = \ddot{y}_0 + \Delta \ddot{y}_1$ $\ddot{y}_{22} = \ddot{y}_{21} + \dddot{y}_{21}\frac{\Delta t}{2}$ $\ddot{y}_{23} = \ddot{y}_{21} + \dddot{y}_{22}\frac{\Delta t}{2}$ $\ddot{y}_{24} = \ddot{y}_{21} + \dddot{y}_{23}\,\Delta t$
$\Delta \dot{y}_2 = \frac{\Delta t}{6}(\dot{y}_{21} + 2\dot{y}_{22} + 2\dot{y}_{23} + \dot{y}_{24})$	$\Delta \ddot{y}_2 = \frac{\Delta t}{6}(\ddot{y}_{21} + 2\ddot{y}_{22} + 2\ddot{y}_{23} + \ddot{y}_{24})$
$\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{21} = f(t_{21}, y_{21}, \dot{y}_{21}, \ddot{y}_{21}, \dddot{y}_{21}, \ddddot{y}_{21})$ $\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{22} = f(t_{22}, y_{22}, \dot{y}_{22}, \ddot{y}_{22}, \dddot{y}_{22}, \ddddot{y}_{22})$ $\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{23} = f(t_{23}, y_{23}, \dot{y}_{23}, \ddot{y}_{23}, \dddot{y}_{23}, \ddddot{y}_{23})$ $\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{24} = f(t_{24}, y_{24}, \dot{y}_{24}, \ddot{y}_{24}, \dddot{y}_{24}, \ddddot{y}_{24})$	
$\Delta \overset{\cdot\cdot\cdot\cdot\cdot}{y}_2 = \frac{\Delta t}{6}(\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{21} + 2\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{22} + 2\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{23} + \overset{\cdot\cdot\cdot\cdot\cdot}{y}_{24})$	

ments in the Solution of a Fifth Order Differential Equation line. A similar pattern is followed in making the calculations along the third line. After completing the fourth line in the same fashion the increments $\Delta \ddddot{y}_1$, $\Delta \dddot{y}_1$, $\Delta \ddot{y}_1$, $\Delta \dot{y}_1$, Δy_1, and Δt are calculated in that order. This completes calculations in the first increment group. One may then proceed to the beginning of calculations for the first step in the second increment group and obtain t_{21}. Next, y_{21} is obtained, and so on. After the fifth derivative is calculated, the balance of the computation proceeds in the same manner as that in the first increment group. Calculations in successive increment groups proceed in the same manner.

Differential analyzer-type output is desired, so in actual computer operation each of the variables in the first step of each increment calculation group is output as soon as its value is available.

Programming the Equations

Figure 1a shows the flow diagram for programming the first half of the calculations in any increment group. The flow diagram for the second half is shown in Figure 1b. The block notation is that given by McCracken (23); however, the functions of all of the blocks in the diagram are largely self-evident. Lines leading to encircled letters make connection with other lines at points where there are identical letters.

Some discussion of the contents of blocks in the flow diagram is desirable. In the upper left-hand corner of Figure 1a is a box indicating calculation and storage of the driving function, step 1. This operation is actually a subroutine in itself and its flow diagram may be found in Figure 2. The program goes into this calculation four times during the calculation of one increment—once for each step therein.

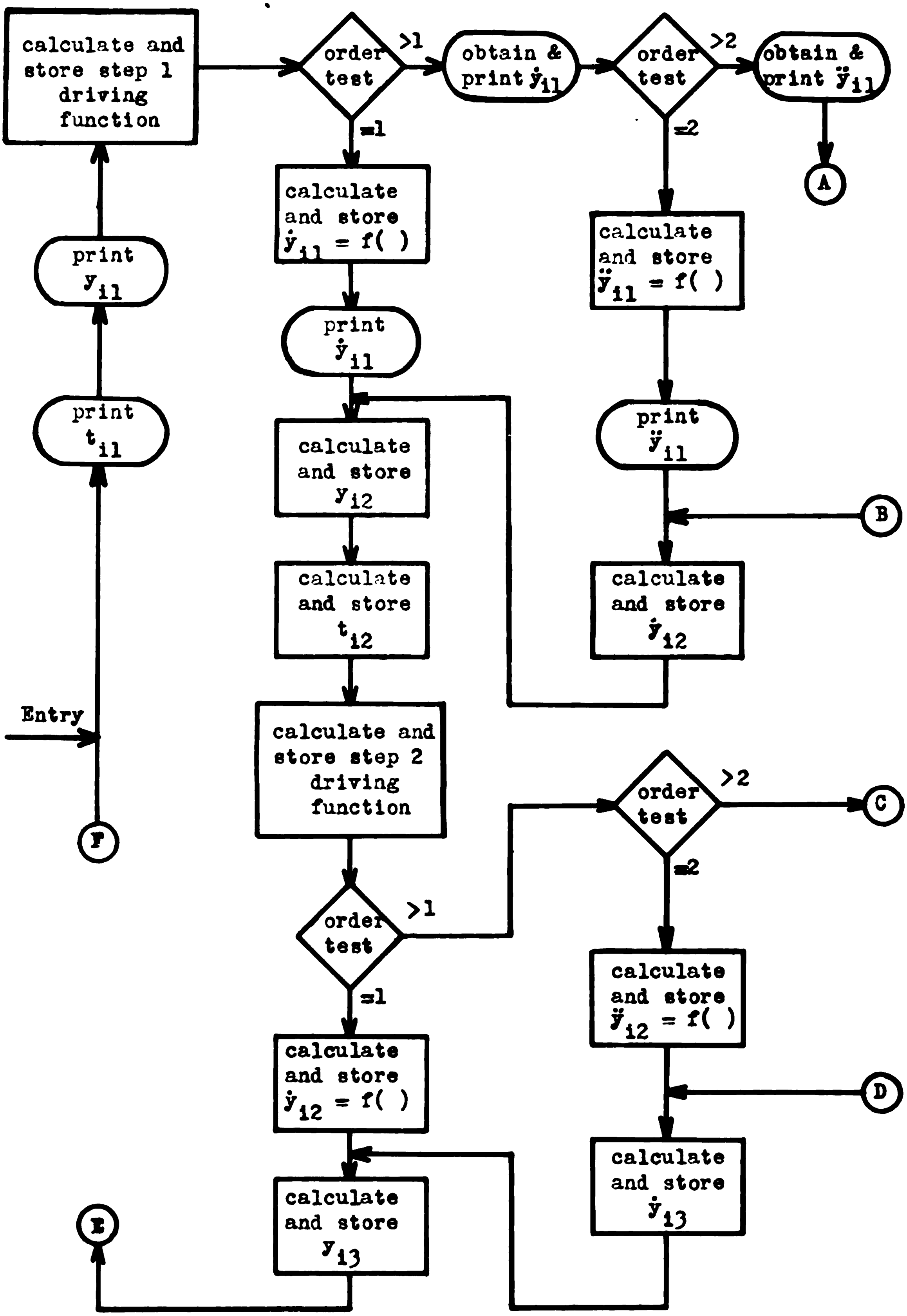
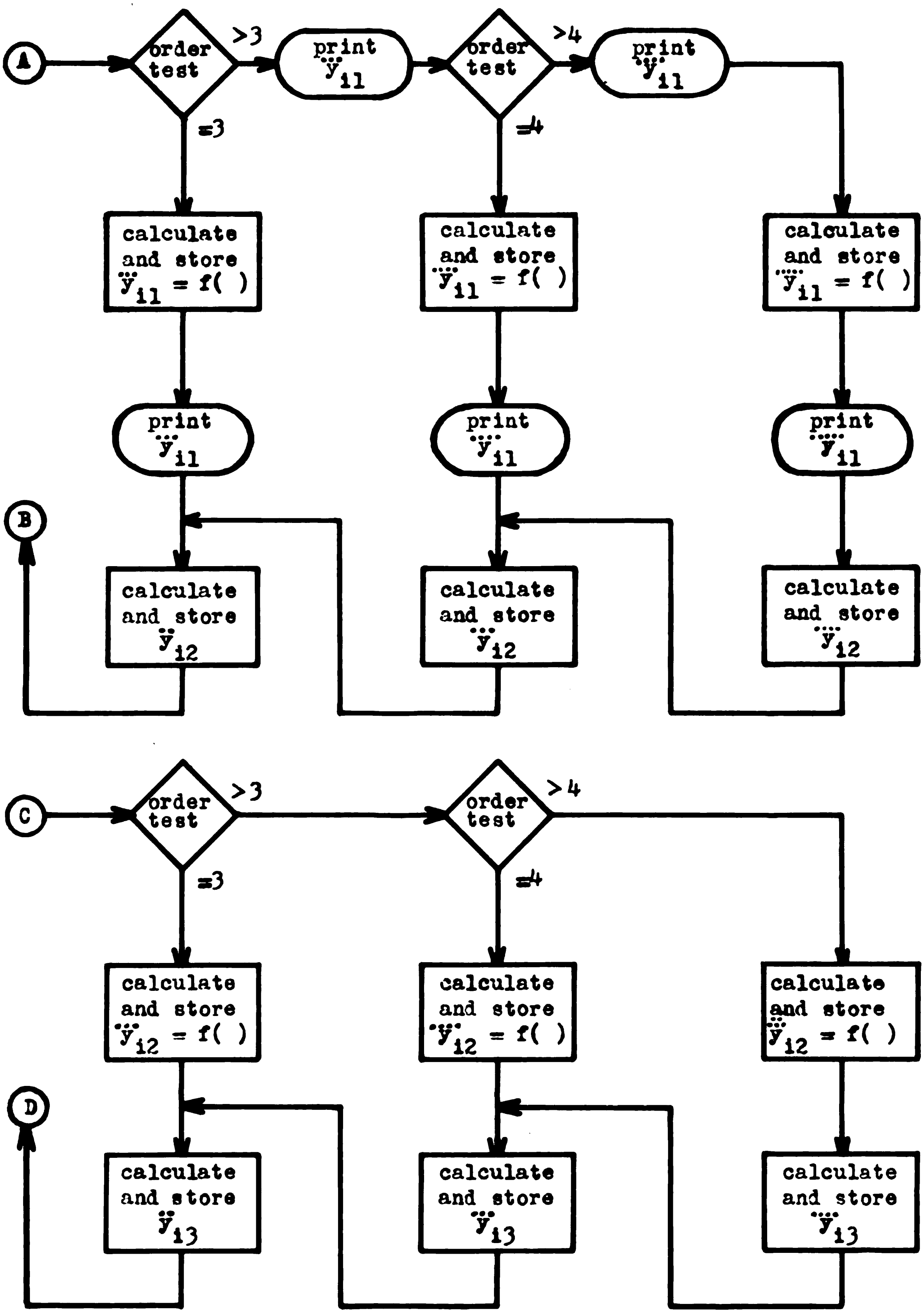

calculate and store step 1 driving function
order test
>1
obtain & print $\dot{y}_{i1}$
order test
>2
obtain & print $\ddot{y}_{i1}$
A
=1
=2
calculate and store $\dot{y}_{i1} = f(\ )$
calculate and store $\ddot{y}_{i1} = f(\ )$
print y_{i1}
print $\dot{y}_{i1}$
print $\ddot{y}_{i1}$
print t_{i1}
calculate and store y_{i2}
B
calculate and store t_{i2}
calculate and store $\dot{y}_{i2}$
Entry
calculate and store step 2 driving function
order test
>2
C
F
=2
order test
>1
calculate and store $\ddot{y}_{i2} = f(\ )$
=1
calculate and store $\dot{y}_{i2} = f(\ )$
D
calculate and store $\dot{y}_{i3}$
E
calculate and store y_{i3}

Fig. 1a.—Flow Diagram of the Programming for the First Half of the


A
order test
>3
print $\dddot{y}_{i1}$
order test
>4
print $\ddddot{y}_{i1}$
=3
=4
calculate and store $\dddot{y}_{i1} = f(\ )$
calculate and store $\ddddot{y}_{i1} = f(\ )$
calculate and store $\ddddot{y}_{i1} = f(\ )$
print $\dddot{y}_{i1}$
print $\ddddot{y}_{i1}$
print $\ddddot{y}_{i1}$
B
calculate and store $\ddot{y}_{i2}$
calculate and store $\dddot{y}_{i2}$
calculate and store $\ddddot{y}_{i2}$
C
order test
>3
order test
>4
=3
=4
calculate and store $\dddot{y}_{i2} = f(\ )$
calculate and store $\ddddot{y}_{i2} = f(\ )$
calculate and store $\ddddot{y}_{i2} = f(\ )$
D
calculate and store $\ddot{y}_{i3}$
calculate and store $\dddot{y}_{i3}$
calculate and store $\ddddot{y}_{i3}$

Differential Equation Routine. (Continued on the following pages.)

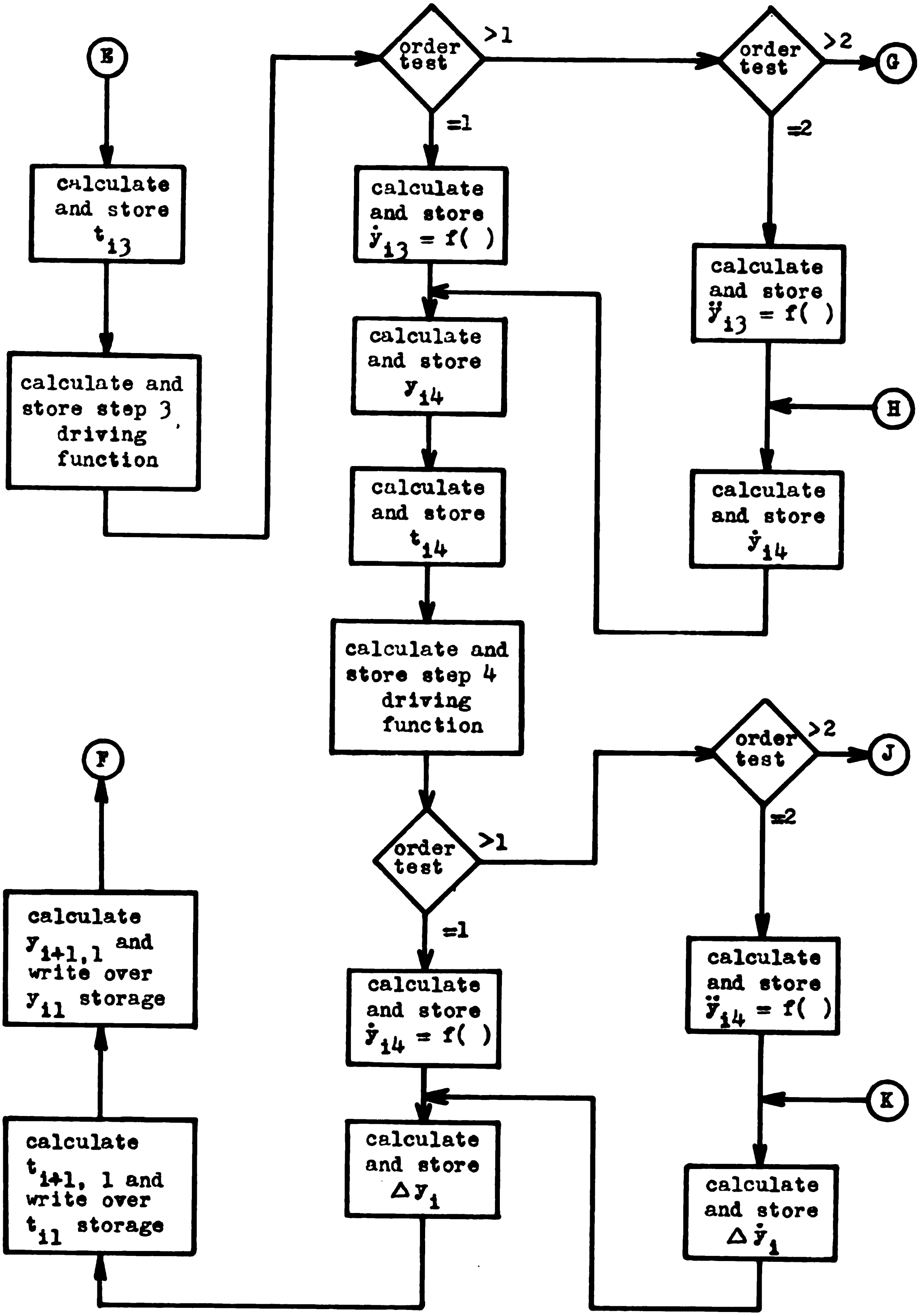
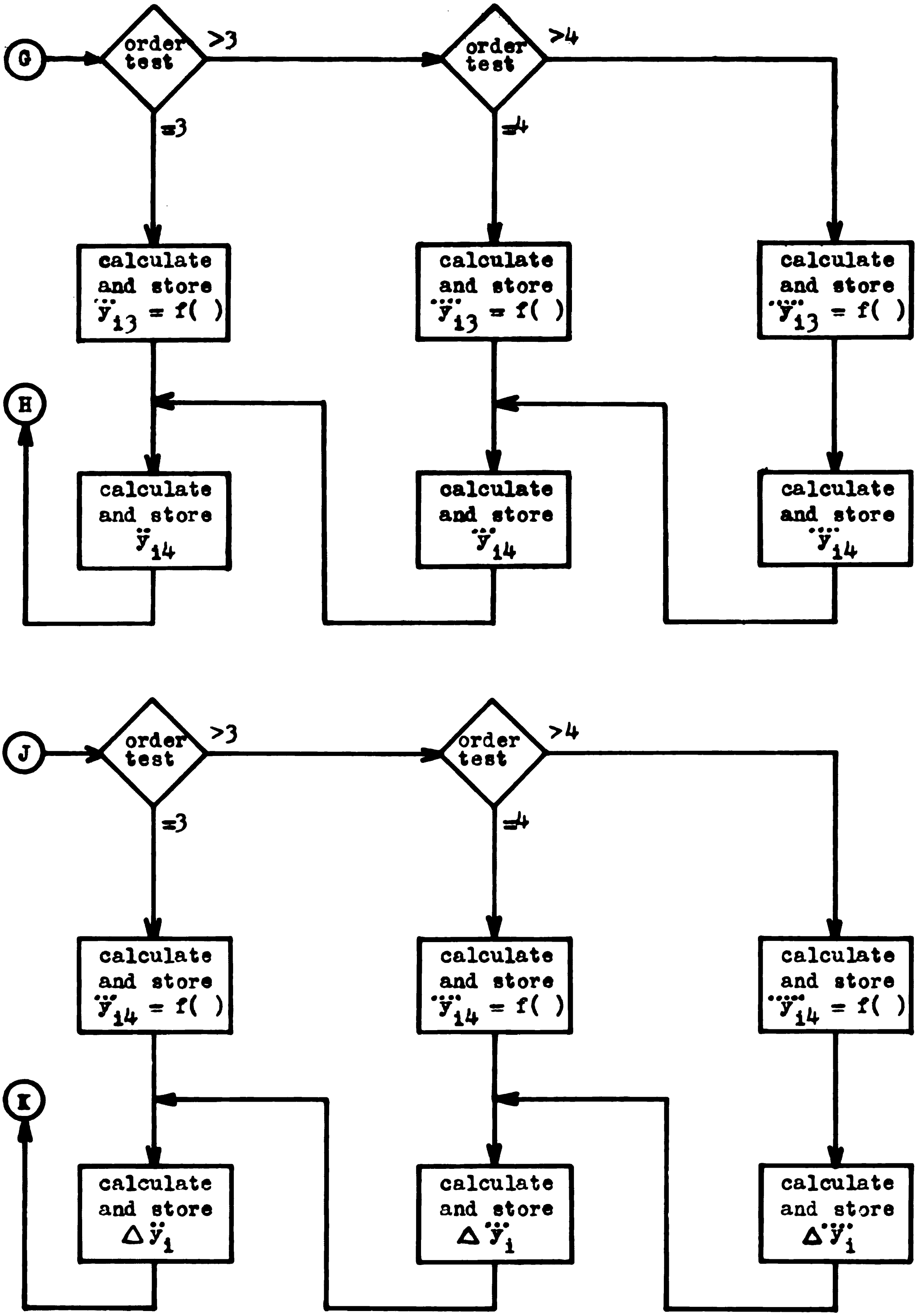

E
calculate and store t_{i3}
calculate and store step 3 driving function
order test
>1
=1
calculate and store $\dot{y}_{i3} = f(\ )$
calculate and store y_{i4}
calculate and store t_{i4}
calculate and store step 4 driving function
order test
>2
G
=2
calculate and store $\ddot{y}_{i3} = f(\ )$
H
calculate and store $\dot{y}_{i4}$
order test
>1
=1
calculate and store $\dot{y}_{i4} = f(\ )$
calculate and store Δy_i
order test
>2
J
=2
calculate and store $\ddot{y}_{i4} = f(\ )$
K
calculate and store $\Delta \dot{y}_i$
F
calculate $y_{i+1,1}$ and write over y_{i1} storage
calculate $t_{i+1,\ 1}$ and write over t_{i1} storage

Fig. 1b.—Flow Diagram of the Programming for the Second Half of the


G
order test
>3
=3
order test
>4
=4
calculate and store $\dddot{y}_{i3} = f(\ )$
calculate and store $\ddddot{y}_{i3} = f(\ )$
calculate and store $\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{i3} = f(\ )$
H
calculate and store $\ddot{y}_{i4}$
calculate and store $\dddot{y}_{i4}$
calculate and store $\ddddot{y}_{i4}$
J
order test
>3
=3
order test
>4
=4
calculate and store $\dddot{y}_{i4} = f(\ )$
calculate and store $\ddddot{y}_{i4} = f(\ )$
calculate and store $\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{i4} = f(\ )$
K
calculate and store $\Delta\ddot{y}_i$
calculate and store $\Delta\dddot{y}_i$
calculate and store $\Delta\ddddot{y}_i$

Differential Equation Routine. (Continued from preceding pages.)

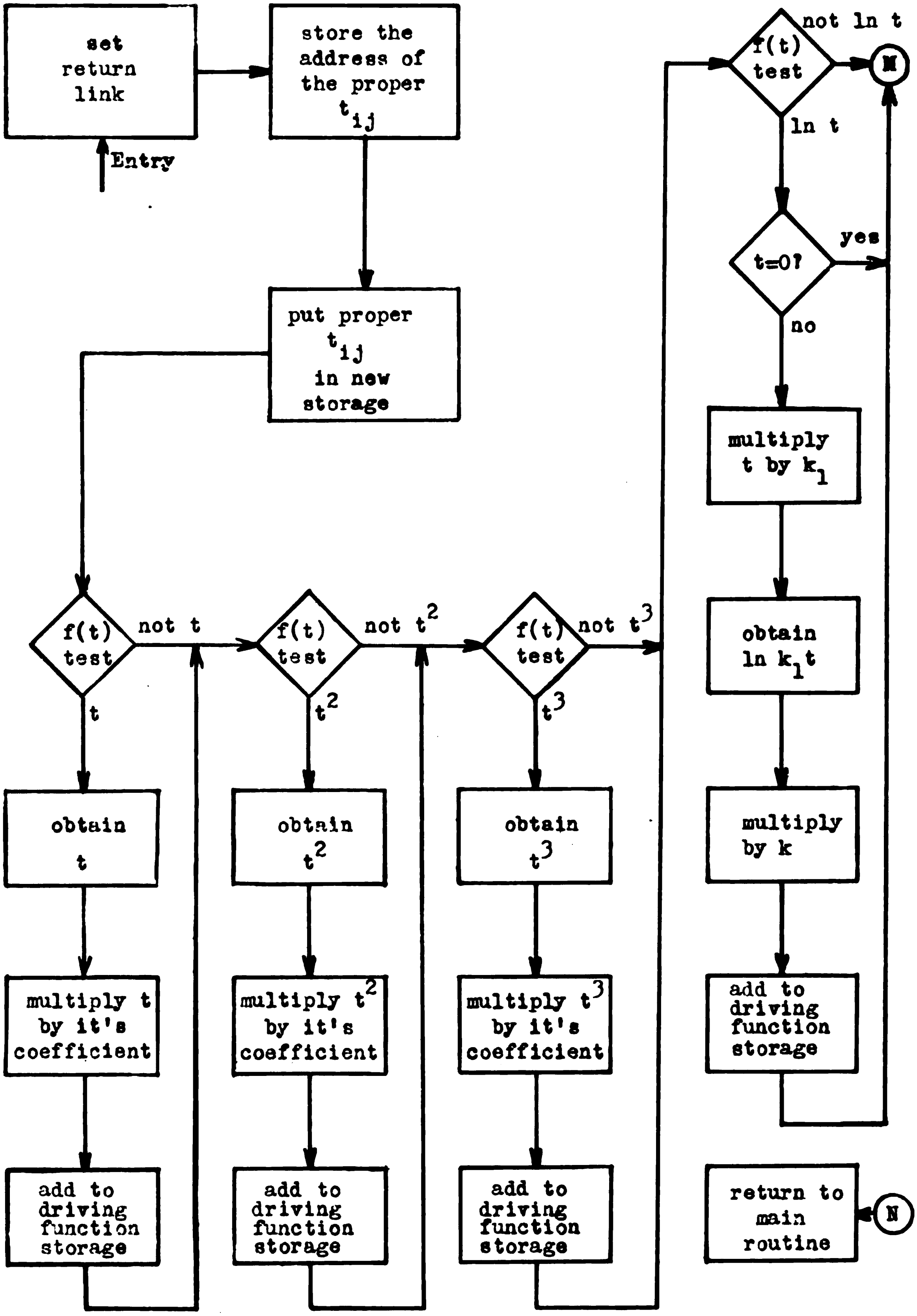
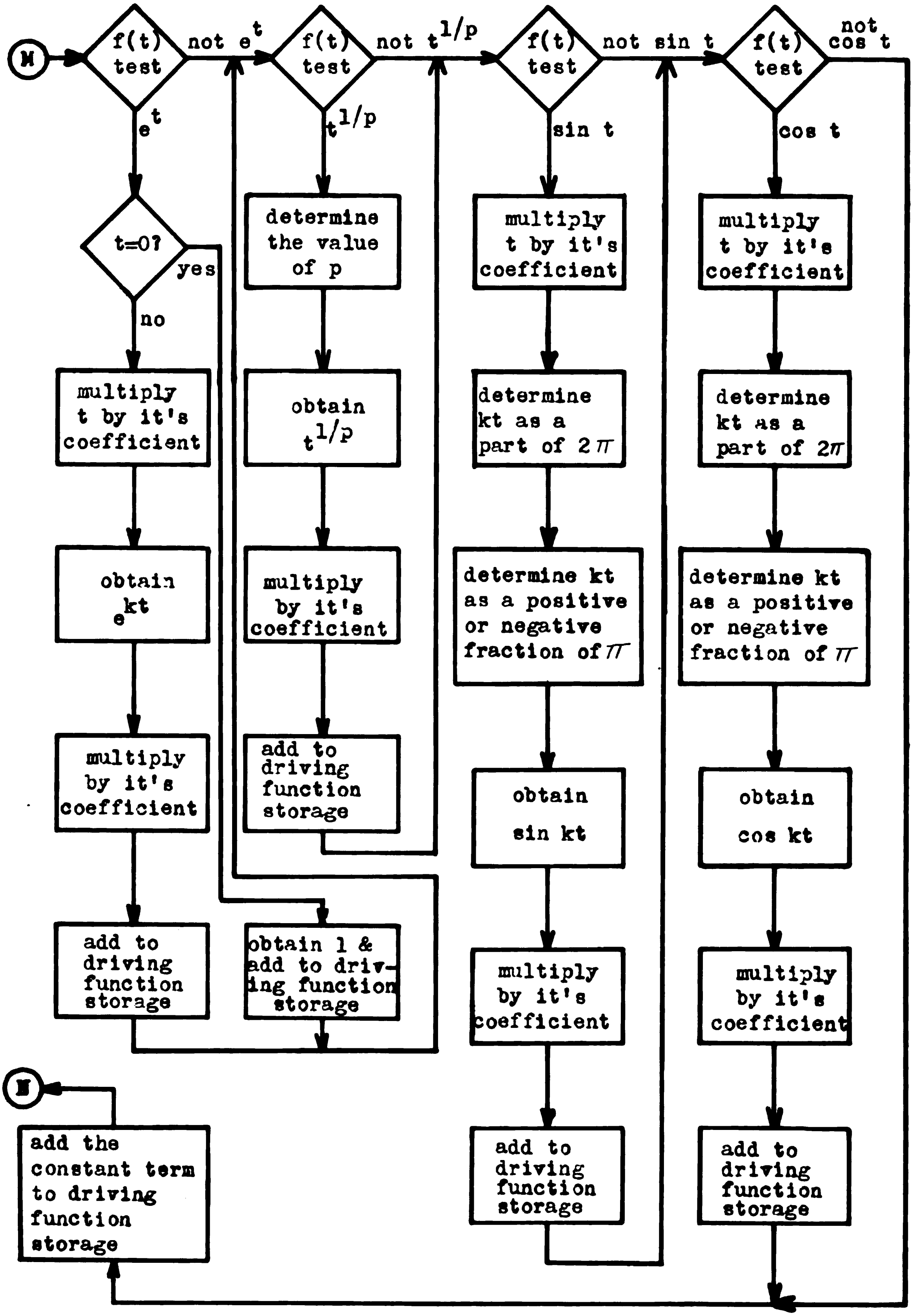

set return link
Entry
store the address of the proper t_{ij}
put proper t_{ij} in new storage
f(t) test
not t
t
obtain t
multiply t by it's coefficient
add to driving function storage
f(t) test
not t^2
t^2
obtain t^2
multiply t^2 by it's coefficient
add to driving function storage
f(t) test
not t^3
t^3
obtain t^3
multiply t^3 by it's coefficient
add to driving function storage
f(t) test
not ln t
ln t
N
t=0?
yes
no
multiply t by k_1
obtain ln k_1t
multiply by k
add to driving function storage
return to main routine
N

Fig. 2.—Flow Diagram for Programming Calculation of the


M
f(t) test
not e^t
e^t
t=0?
yes
no
multiply t by it's coefficient
obtain e^{kt}
multiply by it's coefficient
add to driving function storage
f(t) test
not $t^{1/p}$
$t^{1/p}$
determine the value of p
obtain $t^{1/p}$
multiply by it's coefficient
add to driving function storage
obtain 1 & add to driving function storage
f(t) test
not sin t
sin t
multiply t by it's coefficient
determine kt as a part of 2 π
determine kt as a positive or negative fraction of π
obtain sin kt
multiply by it's coefficient
add to driving function storage
f(t) test
not cos t
cos t
multiply t by it's coefficient
determine kt as a part of 2π
determine kt as a positive or negative fraction of π
obtain cos kt
multiply by it's coefficient
add to driving function storage
add the constant term to driving function storage
N

Driving Function in the Differential Equation Routine

The value of the driving function is obtained here for later use in calculation of the highest derivative. An "order test" box indicates the location of a programmed test to ascertain the order of the differential equation being solved. "Print" boxes indicate the position in the program at which information is being output, and, depending upon the computer, this may come out as punches on tape, printed numbers (see part VI for interpretation) or plotted points. Blocks representing calculation and location of special increment constants, and blocks giving instructions for output have been omitted for simplicity.

Calculation of the Driving Function

The flow diagram of Figure 2 indicates the pattern used for obtaining the value of the driving function. A test to determine whether or not one of the allowed functions is contained in the driving function is indicated by "f(t) test." The routine frequently requires that program control leave it for calculation in other routines specifically designed to obtain the values of certain types of functions. A brief discussion of all the routines and their limitations follows.

The calculation of t, t^2, and t^3 is straightforward, and there is no restriction on the values which t may have. The computer will, however, give incorrect output when either of these powers of t multiplied by its coefficient exceeds 9,999 when unscaled. The same rule applies to any of the succeeding driving functions and its coefficient.

The Logarithm subroutine requires that the number for which it computes the logarithm lie between zero and 1 (not inclusive). Of course, the coefficient of t, k_1 (in ln k_1t) must be less than 10,000 in order to be able to scale it to fractional size according to instructions in part VI and then get it in the computer. If it is less than this number and can be input, then computation of the ln k_1t may proceed as long as the product of scaled k_1 and the unscaled value of t (t is normally carried in the computer at 0.0001 its actual value) is less than unity.

The Exponential routine has a similar requirement except that the product k_1t must lie between -1 and 0. The sign is taken care of in the program so the user need only place the same restrictions on the product of the coefficient and t as in the previous routine.

In calculating the integral root, it should be emphasized that only one of the three "roots" may be calculated by the driving function in the solution of any one differential equation. No other restrictions are necessary.

In using the Fast Sine-Cosine routine, the same restrictions apply as were necessary for the Logarithm routine: the product of the scaled coefficient of t and unscaled t must be less than unity.

Insofar as the constant is concerned, it must be less than unity when scaled for input. This applies, as well, to all coefficients.

It may be seen by inspection of Figure 2 that additional driving functions may be added readily with very little additional programming. Actually, the entire Differential Equation routine may be easily lifted out of the Compiler and used by itself with no modification when additional driving functions are required for differential equation solving.

Assembly Routine

The Assembly routine is shown in block diagram form in Figure 3. The programming of this routine is begun with order pairs at location

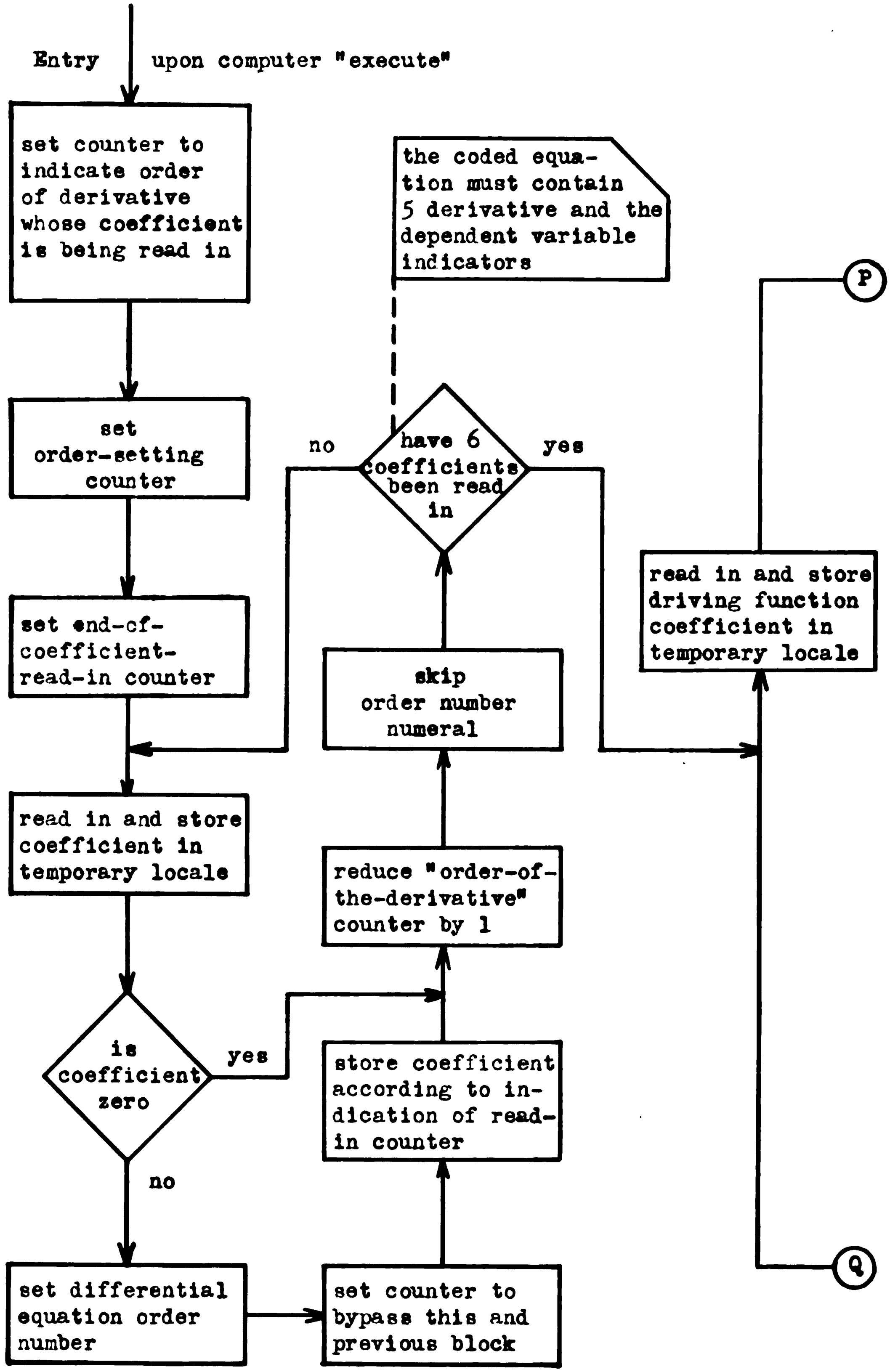
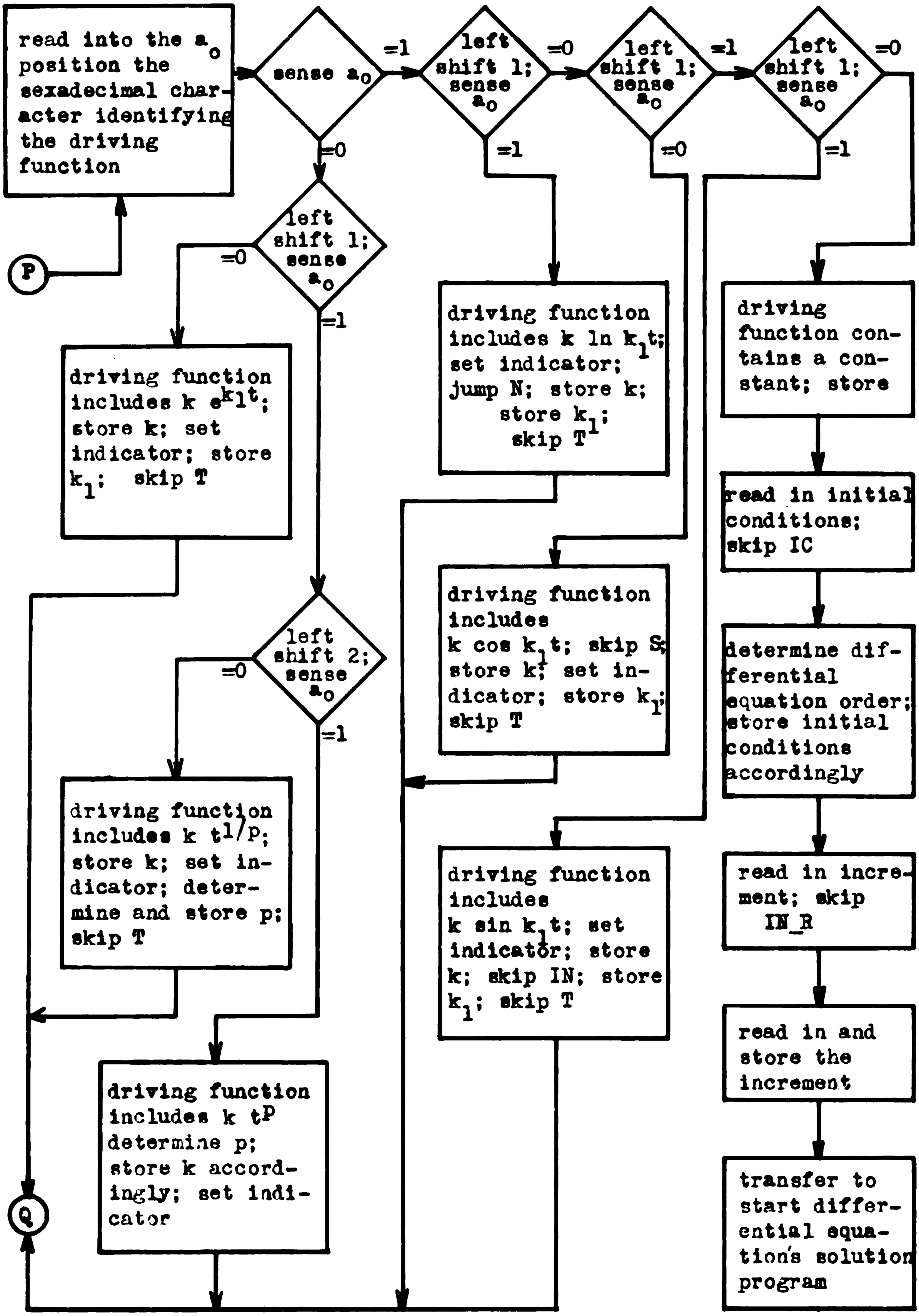

Entry
upon computer "execute"
set counter to indicate order of derivative whose coefficient is being read in
set order-setting counter
set end-of-coefficient-read-in counter
read in and store coefficient in temporary locale
is coefficient zero
yes
no
set differential equation order number
set counter to bypass this and previous block
store coefficient according to indication of read-in counter
reduce "order-of-the-derivative" counter by 1
skip order number numeral
have 6 coefficients been read in
no
yes
the coded equation must contain 5 derivative and the dependent variable indicators
read in and store driving function coefficient in temporary locale
P
Q

Fig. 3.--Flow Diagram for Pro-


read into the a_o position the sexadecimal character identifying the driving function
P
sense a_o
=1
=0
left shift 1; sense a_o
=0
=1
left shift 1; sense a_o
=1
=0
left shift 1; sense a_o
=0
=1
left shift 1; sense a_o
=0
=1
driving function includes k $e^{k_1 t}$; store k; set indicator; store k_1; skip T
driving function includes k ln k_1t; set indicator; jump N; store k; store k_1; skip T
driving function contains a constant; store
read in initial conditions; skip IC
left shift 2; sense a_o
=0
=1
driving function includes k cos k_1t; skip S; store k; set indicator; store k_1; skip T
determine differential equation order; store initial conditions accordingly
driving function includes k $t^{1/p}$; store k; set indicator; determine and store p; skip T
driving function includes k sin k_1t; set indicator; store k; skip IN; store k_1; skip T
read in increment; skip IN_R
read in and store the increment
driving function includes k t^p determine p; store k accordingly; set indicator
Q
transfer to start differential equation's solution program

gramming the Compiler Routine

670 in the memory (see part V). Its sequence of operations begins with reading in and storing each of the coefficients of the derivatives. As soon as a non-zero coefficient of a derivative is sensed, the order of the differential equation is available and a constant is set to indicate this order. After all coefficients are input and stored properly, the Assembly routine begins sensing more of the differential equation code to determine the nature of the driving functions, initial conditions, and increment, and set indicators accordingly. The actual code used is given in part VI together with instructions for its use. The code is also listed below together with the characters of each code group which is sensed and the binary representation of the sensed characters. The letters "IC" and "INCR" representing respectively "initial conditions and "increment" are also equation code but are merely indicators and are not sensed for directions; therefore, they are not listed. (Lower-case k and k_1 are constants.)

Driving Function	Code	Character(s) Sensed	Binary Representation(s)
$k \ln k_1 t$	$kFLNk_1NT$	L	1111
$k \sin k_1 t$	$kFSINk_1NT$	S	1011
$k \cos k_1 t$	$kFCOSk_1NT$	O	1001
constant, k	kFK	K	1010
k t	kFT1	T1	1001 0001
$k\ t^2$	kFT2	T2	1001 0010
$k\ t^3$	kFT3	T3	1001 0011
$k\ t^{1/2}$	kFR2T	R2	0100 0010
$k\ t^{1/3}$	kFR3T	R3	0100 0011
$k\ t^{1/4}$	kFR4T	R4	0100 0100
$k\ t^{1/5}$	kFR5T	R5	0100 0101

Driving Function	Code	Character(s) Sensed	Binary Representation(s)
$k\ e^{k_1 t}$	$kFEk_1T$	E	0011

Inspection of the coding will show that the coefficient of the driving function can be easily read in with the input routine (which is stopped by the "F")-after which the next character in the code can be sensed in the a_0 position of the accumulator. This holds true for all except the cosine code. In practice it holds there, also, because the "C" is a fifth hole character and as such is skipped by the read-in process. When the character following the "F" is sensed, function indicators are set accordingly. This is the general plan of the driving function sensing, and it includes the setting of indicators for as many functions as are included in the equation.

Initial conditions are read in after the differential equation driving functions are determined. They are stored according to information sensed on the order of the equation. Finally, the increment is read in and stored. Control is then transferred to the beginning of the Differential Equation routine.

Other Routines

The function of each of the other routines listed in the general description of the Compiler is indicated by its title--the last two routines being responsible for all input and output operations. Finally, a brief increment-constant calculating routine precedes the Differential Equation routine and would need to be included with it in any attempt to use it apart from the Compiler.

IV. EXPERIMENTAL PROCEDURE

The first experimental work was planned to test the speed, ease of programming, and accuracy of the Runge-Kutta equations using a second order differential equation with only a simple driving function, t, and an increment of t = 0.1. The results were good in each case: speed was such that a complete increment was calculated and output for each variable in approximately one second; the programming was readily carried out; and the results were accurate well beyond normal three-to-four-place engineering requirements. This accuracy remained even when the program was run for a great many cycles and when it might be expected that round-off error would become appreciable. This was evident in a test in which three values of the increment of the independent variable , t, 0.01, 0.05, and 0.1 were used in three separate program runs of the same second order equation described above. Six or seven significant figure agreement between the values of the different variables calculated for each of these values of Δ t was evident after t = 45 seconds. Increments of 0.3, 0.6, and 0.9 were also tested; however, the error for these values grew excessive rapidly. The increment of 0.1 was used in all further testing.

A general fifth order differential equation solution program (solving any order up to and including the fifth) was written thereafter and was tested with the same second order equation used above. After it worked successfully and minor changes were made to improve accuracy, equations of first, third, fourth, and fifth orders were tested and run successfully with the simple driving function, t, used above.

After the main differential equation solving routine appeared to run successfully for all equation orders, testing was carried out on expanding its driving function calculating subroutine to include the calculation of all the other allowed (see part VI) functions. The testing of the calculation of each of the other functions was performed with a first order equation having the driving function under test. The test was considered successful when the results corresponded with values obtained from an analytical solution. The end of these tests marked the completion of the Differential Equation routine.

The Compiler Assembly routine was begun next and was tested first to read in properly all the coefficients of derivatives. Thereafter, each test included coefficient read-in and the proper read-in of another driving function code. After each driving function code had been checked, several combinations of those codes in typically-encoded differential equations were further checked. Each checking involved an examination of computer post-mortem print-outs indicating the storage in certain locations since the ability of the Differential Equation routine to run properly depends upon proper storage of data from the encoded equation. Finally, each of the equations which had been run earlier in testing the Differential Equation routine were encoded and fed to the computer after the Compiler—now complete with the addition of the Assembly routine. All ran successfully and the Compiler programming and testing was considered complete.

The Compiler might have been written to punch out on tape the completely assembled program which it prepares in the computer storage. It was decided that there was little justification for this when it would only involve the loss of time required to output and then input the same information again before starting the solution. For this reason, computer control is transferred immediately to the Differential Equation routine for the beginning of the solution just as soon as the Assembly routine brings in the encoded equation.

Some operating times are of interest. It requires approximately fifty-two seconds to input the Compiler. Program assembly is accomplished in negligible time. The time required to obtain and output all of the increments for all of the variables in a given equation is called the "time per cycle." Typical values follow:

Order of Equation	Driving Function	Time per Cycle (Sec.)
1	t	0.93
2	t	1.25
3	t	1.6
4	t	2.0
5	t	2.4
1	0	0.94
1	t^2	0.96
1	t^3	0.96
1	$\ln t$	1.05
1	$t^{1/2}$	1.25
1	$t^{1/3}$	1.4
1	$t^{1/4}$	1.7
1	$t^{1/5}$	1.7
1	e^t	8.2
1	$\cos t$	1.0

Order of Equation	Driving Function	Time per Cycle (sec.)
1	sin t	1.0

V. THE COMPLETE COMPILER PROGRAM

Order pairs for the complete Compiler program are given in the pages which follow. No attempt has been made to "tighten-up" the program. In fact, room has been left in the program for easy modification of sections where, for example, it might be desirable to add a driving function or modify the calculation of an existing one.

This program, without additions or modifications, represents the entire code needed for preceding the encoded differential equation tape discussed in part VI.

Some special notation includes:

f( ) - The value of the highest order derivative for any step in the four-step process is the explicit function of all the other variables and the constant in the differential equation evaluated with the values these variables have at any given step. This function is represented thus.

order test - Test to determine the order of the differential equation.

Letters such at t_{i1}, y_{i1}, et cetera, are those in the Runge-Kutta relations on pages 13 and 14.

LOCATION	ORDER	NOTES
	Library Routine X1	Decimal Order Input
	00 20K	
20	22 20F	
	L5 519F	Calculate and store $\frac{\Delta t}{2}$
21	10 1F	
	40 520F	
22	50 519F	Calculate and store $\frac{\Delta t}{6}$
	7J 490F	
23	40 521F	
	50 519F	Calculate and store $\Delta t(x\ 10^{-4})$
24	7J 491F	
	40 517F	
25	L5 517F	Calculate and store $\frac{\Delta t}{2}(x\ 10^{-4})$
	10 1F	
26	40 518F	
	26 27F	
27	92 131F	Carriage return and line feed
	92 131F	
28	92 515F	Delay
	L5 516F	Print out t_{i1}
29	52 114F	
	50 29F	
30	26 963F	
	92 963F	2 carriage spaces
31	92 963F	
	26 33F	
32	26 33F	
	26 33F	
33	22 33F	
	L5 515F	Print out y_{i1}
34	52 114F	
	50 34F	
35	26 963F	
	92 963F	2 spaces
36	92 963F	

```
LOCATION         ORDER                       NOTES
            Library Routine X1          Decimal Order Input

                 00 20K
   20            22 20F
                 L5 519F
   21            10 1F                  Calculate and store Δt/2
                 40 520F
   22            50 519F
                 7J 490F                Calculate and store Δt/6
   23            40 521F
                 50 519F
   24            7J 491F                Calculate and store Δt(x 10^-4)
                 40 517F
   25            L5 517F
                 10 1F                  Calculate and store Δt/2(x 10^-4)
   26            40 518F
                 26 27F
   27            92 131F                Carriage return and line feed
                 92 131F
   28            92 515F                Delay
                 L5 516F
   29            52 114F                Print out t_i1
                 50 29F
   30            26 963F
                 92 963F                2 carriage spaces
   31            92 963F
                 26 33F
   32            26 33F
                 26 33F
   33            22 33F
                 L5 515F
   34            52 114F                Print out y_i1
                 50 34F
   35            26 963F
                 92 963F                2 spaces
   36            92 963F
```

LOCATION	ORDER	NOTES
	26 38F	
37	26 38F	
	26 38F	
38	50 516F	(516) is t_{i1}
	50 38F	Transfer to a subroutine to calculate the driving function of the first calculation of f( )
39	26 373F	
	26 40F	
40	L5 493F	
	L0 492F	Order test No. 1
41	36 42F	(differential equation order = 1)
	26 91F	(differential equation order > 1)
42	50 505F	Control to print $\dot{y}_{i1}$
	71 515F	
43	66 491F	
	S5 F	Calculate and store $\dot{y}_{i1} = f( )$
44	L4 566F	
	40 514F	
45	22 45F	
	L5 514F	
46	52 114F	Print $\dot{y}_{i1} = f( )$
	50 46F	
47	26 963F	
	26 50F	
48	26 50F	
	26 50F	
49	26 50F	
	26 50F	
50	50 520F	
	7J 514F	Calculate and store y_{i2}
51	L4 515F	
	40 536F	
52	L5 518F	
	L4 516F	Calculate and store t_{i2}
53	40 537F	

LOCATION	ORDER	NOTES
	26 54F	
54	50 537F	(537) is t_{12}
	50 54F	Transfer to a subroutine and calculate the driving function of the second calculation of f( )
55	26 373F	
	26 56F	
56	L5 493F	Order test No. 2
	L0 492F	
57	36 58F	(differential equation order = 1)
	26 110F	(differential equation order > 1) Control to order test No. 6
58	50 505F	
	71 536F	
59	66 491F	
	S5 F	Calculate and store $\dot{y}_{12}$ = f( )
60	L4 566F	
	40 535F	
61	50 520F	
	7J 535F	Calculate and store y_{13}
62	L4 515F	
	40 543F	
63	L5 537F	Obtain and store t_{13}
	40 544F	
64	50 544F	Transfer to subroutine to calculate the driving function of the third calculation of f( )
	50 64F	
65	26 373F	
	26 66F	
66	L5 493F	Order test No. 3
	L0 492F	
67	36 68F	(Differential equation order = 1)
	22 120F	(Differential equation order > 1) Control to order test No. 7
68	50 505F	
	71 543F	
69	66 491F	Calculate and store $\dot{y}_{13}$ = f( )
	S5 F	
70	L5 566F	

LOCATION	ORDER	NOTES
	40 542F	
71	50 519F	
	7J 542F	Calculate and store y_{i4}
72	L4 515F	
	40 550F	
73	L5 517F	
	L4 516F	Calculate and store t_{i4}
74	40 551F	
	26 75F	
75	50 551F	
	50 75F	Transfer to subroutine to calculate the driving function of the fourth calculation of f( )
76	26 373F	
	L5 493F	
77	LO 492F	Order test No. 4
	36 79F	(differential equation order = 1)
78	26 131F	(differential equation order > 1) Control to order test No. 8
	26 79F	
79	50 505F	
	71 550F	
80	66 491F	Calculate and store $\dot{y}_{i4}$ = f( )
	S5 F	
81	L4 566F	
	40 549F	
82	22 82F	
	11 39F	
83	L5 535F	
	L4 542F	
84	00 1F	
	L4 514F	Calculate and store Δy_i
85	L4 549F	
	40 556F	
86	50 556F	
	7J 521F	

LOCATION	ORDER	NOTES
87	40 561F	
	L5 517F	
88	L4 516F	Calculate and store $t_{i+1,1}$
	40 516F	
89	L5 561F	
	L4 515F	Calculate and store $y_{i+1,1}$
90	40 515F	
	26 27F	Control to 27 to begin calculation of values at the next increment
91	L5 514F	
	L4 560F	
92	40 514F	
	L5 514F	Obtain, store and print $\dot{y}_{i1}$
93	52 114F	
	50 93F	
94	26 963F	
	92 963F	2 spaces
95	92 963F	
	26 97F	
96	26 97F	
	26 97F	
97	L5 494F	
	L0 492F	Order test No. 5
98	36 99F	(differential equation order = 2)
	22 144F	(differential equation order > 2)
99	50 505F	Control to print $\ddot{y}_{i1}$
	71 515F	
100	66 491F	
	S5 F	
101	40 526F	
	50 504F	Obtain and store $\ddot{y}_{i1} = f(\ )$
102	71 514F	
	66 491F	
103	S5 F	

LOCATION	ORDER	NOTES
	L4 526F	
104	L4 566F	
	40 513F	
105	22 105F	
	L5 513F	
106	52 114F	Print $\ddot{y}_{11}$ = f( )
	50 106F	
107	26 963F	
	50 520F	
108	7J 513F	
	L4 514F	Calculate and store $\dot{y}_{12}$
109	40 535F	
	26 50F	Control to calculate y_{12}
110	L5 494F	
	L0 492F	Order test No. 6
111	36 112F	(differential equation order = 2)
	26 164F	(differential equation order > 2) Control to order test No. 10
112	50 505F	
	71 536F	
113	66 491F	
	S5 F	
114	40 526F	
	50 504F	Calculate and store $\ddot{y}_{12}$ = f( )
115	71 535F	
	66 491F	
116	S5 F	
	L4 526F	
117	L4 566F	
	40 534F	
118	50 520F	
	7J 534F	Calculate and store $\dot{y}_{13}$
119	L4 514F	
	40 542F	

LOCATION	ORDER	NOTES
120	26 61F	Control to calculate and store y_{13}
	L5 494F	
121	L0 492F	Order test No. 7
	32 122F	(differential equation order = 2)
122	22 177F	(differential equation order > 2) Control to order test No. 11
	50 505F	
123	71 543F	
	66 491F	
124	S5 F	
	40 526F	
125	50 504F	Calculate and store $\ddot{y}_{13} = f(\ )$
	71 542F	
126	66 491F	
	S5 F	
127	L4 526F	
	L4 566F	
128	40 541F	
	50 519F	
129	7J 541F	Calculate and store $\dot{y}_{14}$
	L4 514F	
130	40 549F	
	26 71F	Control to calculate and store y_{14}
131	L5 494F	
	L0 492F	Order test No. 8
132	36 133F	(differential equation order = 2)
	26 191F	(differential equation order > 2) Control to order test No. 12
133	50 505F	
	71 550F	
134	66 491F	
	S5 F	
135	40 526F	
	50 504F	
136	71 549F	
	66 491F	

LOCATION	ORDER	NOTES
137	S5 F	Calculate and store $\ddot{y}_{i4} = f(\ )$
	L4 526F	
138	L4 566F	
	40 548F	
139	11 39F	
	L5 534F	
140	L4 541F	
	00 1F	
141	L4 513F	Calculate and store $\Delta\dot{y}_i$
	L4 548F	
142	40 555F	
	50 555F	
143	7J 521F	
	40 560F	
144	22 82F	Control to calculate and store Δy_i
	L5 513F	
145	L4 559F	
	40 513F	Obtain, store and print $\ddot{y}_{i1}$
146	52 114F	
	50 146F	
147	26 963F	
	92 963F	2 spaces
148	92 963F	
	L5 495F	
149	L0 492F	Order test No. 9
	32 150F	(Differential equation order = 3)
150	26 208F	(Differential equation order > 3)
	50 505F	Control to print $\dddot{y}_{i1}$
151	71 515F	
	66 491F	
152	S5 F	
	40 526F	
153	50 504F	

LOCATION	ORDER	NOTES
	71 514F	
154	66 491F	
	S5 F	Calculate and store $\dddot{y}_{i1}$ = f( )
155	40 525F	
	50 503F	
156	71 513F	
	66 491F	
157	S5 F	
	L4 526F	
158	L4 525F	
	L4 566F	
159	40 512F	
	L5 512F	
160	52 114F	Print $\dddot{y}_{i1}$ = f( )
	50 160F	
161	26 963F	
	50 520F	
162	7J 512F	Calculate and store $\ddot{y}_{i2}$
	L4 513F	
163	40 534F	
	22 107F	Control to calculate and store $\dot{y}_{i2}$
164	L5 495F	
	L0 492F	Order test No. 10
165	36 166F	(differential equation order = 3)
	22 234F	(differential equation order > 3) Control to order test No. 14
166	50 505F	
	71 536F	
167	66 491F	
	S5 F	
168	40 526F	
	50 504F	
169	71 535F	
	66 491F	

LOCATION	ORDER	NOTES
170	S5 F	
	40 525F	Calculate and store $\dddot{y}_{12} = f(\ )$
171	50 503F	
	71 534F	
172	66 491F	
	S5 F	
173	L4 526F	
	L4 525F	
174	L4 566F	
	40 533F	
175	50 520F	
	7J 533F	Calculate and store $\ddot{y}_{13}$
176	L4 513F	
	40 541F	
177	26 118F	Control to calculate and store $\dot{y}_{13}$
	L5 495F	
178	L0 492F	Order test No. 11
	32 179F	(differential equation order = 3)
179	26 252F	(differential equation order > 3)
	50 505F	Control to order test No. 15
180	71 543F	
	66 491F	
181	S5 F	
	40 526F	
182	50 504F	
	71 542F	
183	66 491F	
	S5 F	Calculate and store $\dddot{y}_{13} = f(\ )$
184	40 525F	
	50 503F	
185	71 541F	
	66 491F	
186	S5 F	

LOCATION	ORDER	NOTES
	L4 526F	
187	L4 525F	
	L4 566F	
188	40 540F	
	50 519F	
189	7J 540F	Calculate and store $\ddot{y}_{i4}$
	L4 513F	
190	40 548F	
	22 128F	Control to calculate and store $\dot{y}_{i4}$
191	L5 495F	
	L0 492F	Order test No. 12
192	36 193F	(differential equation order = 3)
	26 269F	(differential equation order > 3) Control to order test No. 16
193	50 505F	
	71 550F	
194	66 491F	
	S5 F	
195	40 526F	
	50 504F	
196	71 549F	
	66 491F	
197	S5 F	Calculate and store $\dddot{y}_{i4}$ = f( )
	40 525F	
198	50 503F	
	71 548F	
199	66 491F	
	S5 F	
200	L4 526F	
	L4 525F	
201	L4 566F	
	40 547F	
202	11 39F	
	L5 533F	

LOCATION	ORDER	NOTES
203	L4 540F	
	00 1F	
204	L4 512F	Calculate and store $\Delta\ddot{y}_i$
	L4 547F	
205	40 554F	
	50 554F	
206	7J 521F	
	40 559F	
207	26 139F	Control to calculate and store $\Delta\dot{y}_i$
	26 208F	
208	L5 512F	
	L4 558F	
209	40 512F	
	L5 512F	Obtain and print $\dddot{y}_{i1}$
210	52 114F	
	50 210F	
211	26 963F	
	26 212F	
212	92 131F	Carriage return, line feed
	92 515F	Delay
213	92 67F	Tab
	92 515F	Delay
214	92 67F	Tab
	92 515F	Delay
215	L5 496F	
	L0 492F	Order test No. 13
216	36 217F	(differential equation order = 4)
	26 289F	(differential equation order > 4)
217	50 505F	Control to print $\ddddot{y}_{i1}$
	71 515F	
218	66 491F	
	S5 F	
219	40 526F	
	50 504F	

LOCATION	ORDER	NOTES
220	71 514F	
	66 491F	
221	S5 F	
	40 525F	
222	50 503F	
	71 513F	Calculate and store $\ddddot{y}_{11}$ = f( )
223	66 491F	
	S5 F	
224	40 524F	
	50 502F	
225	71 512F	
	66 491F	
226	S5 F	
	L4 526F	
227	L4 525F	
	L4 524F	
228	L4 566F	
	40 511F	
229	22 229F	
	L5 511F	
230	52 114F	Print $\ddddot{y}_{11}$ = f( )
	50 230F	
231	26 963F	
	50 520F	
232	7J 511F	
	L4 512F	Calculate and store $\dddot{y}_{12}$
233	40 533F	
	22 161F	Control to calculate and store $\ddot{y}_{12}$
234	22 234F	
	L5 496F	
235	L0 492F	Order test No. 14
	32 236F	(differential equation order = 4)
236	22 315F	(differential equation order > 4)
	50 505F	Control to calculate and store $\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{12}$

LOCATION	ORDER	NOTES
237	71 536F	
	66 491F	
238	S5 F	
	40 526F	
239	50 504F	
	71 535F	
240	66 491F	
	S5 F	
241	40 525F	
	50 503F	
242	71 534F	Calculate and store $\ddddot{y}_{i2} = f(\ )$
	66 491F	
243	S5 F	
	40 524F	
244	50 502F	
	71 533F	
245	66 491F	
	S5 F	
246	L4 526F	
	L4 525F	
247	L4 524F	
	L4 566F	
248	40 532F	
	26 249F	
249	50 520F	
	7J 532F	Calculate and store $\dddot{y}_{i3}$
250	L4 512F	
	40 540F	
251	26 175F	Control to calculate and store $\ddot{y}_{i3}$
	26 252F	
252	L5 496F	
	L0 492F	Order test No. 15
253	36 254F	(differential equation order = 4)
	26 334F	(differential equation order > 4) Control to calculate and store $\ddddot{y}_{i3}$

LOCATION	ORDER	NOTES
254	50 505F	
	71 543F	
255	66 491F	
	S5 F	
256	40 526F	
	50 504F	
257	71 542F	
	66 491F	
258	S5 F	
	40 525F	
259	50 503F	
	71 541F	Calculate and store $\ddddot{y}_{13}$ = f( )
260	66 491F	
	S5 F	
261	40 524F	
	50 502F	
262	71 540F	
	66 491F	
263	S5 F	
	L4 526F	
264	L4 525F	
	L4 524F	
265	L4 566F	
	40 539F	
266	50 519F	
	7J 539F	Calculate and store $\dddot{y}_{14}$
267	L4 512F	
	40 547F	
268	22 188F	Control to calculate and store $\ddot{y}_{14}$
	26 269F	
269	L5 496F	
	L0 492F	Order test No. 16
270	36 271F	(differential equation order = 4)

LOCATION	ORDER	NOTES
	26 352F	(differential equation order > 4)
271	50 505F	Control to calculate and store $\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{i4}$
	71 550F	
272	66 491F	
	S5 F	
273	40 526F	
	50 504F	
274	71 549F	
	66 491F	
275	S5 F	
	40 525F	
276	50 503F	
	71 548F	Calculate and store $\ddddot{y}_{i4} = f(\ )$
277	66 491F	
	S5 F	
278	40 524F	
	50 502F	
279	71 547F	
	66 491F	
280	S5 F	
	L4 526F	
281	L4 525F	
	L4 524F	
282	L4 566F	
	40 546F	
283	11 39F	
	L5 532F	
284	L4 539F	
	00 1F	
285	L4 511F	Calculate and store $\Delta\dddot{y}_i$
	L4 546F	
286	40 553F	
	50 553F	
287	7J 521F	

LOCATION	ORDER	NOTES
	40 558F	
288	26 202F	Control to calculate and store $\Delta\ddot{y}_i$
	26 289F	
289	L5 511F	
	L4 557F	
290	40 511F	
	L5 511F	Calculate, store and print $\ddddot{y}_{i1}$
291	52 114F	
	50 291F	
292	26 963F	
	92 963F	
293	92 963F	
	92 963F	5 spaces
294	92 963F	
	92 963F	
295	50 505F	
	71 515F	
296	66 491F	
	S5 F	
297	40 526F	
	50 504F	
298	71 514F	
	66 491F	
299	S5 F	
	40 525F	
300	50 503F	
	71 513F	
301	66 491F	Calculate and store $\ddddot{y}_{i1} = f(\ )$
	S5 F	
302	40 524F	
	50 502F	
303	71 512F	
	66 491F	

LOCATION	ORDER	NOTES
304	S5 F	
	40 523F	
305	50 501F	
	71 511F	
306	66 491F	
	S5 F	
307	L4 526F	
	L4 525F	
308	L4 524F	
	L4 523F	
309	L4 566F	
	40 567F	
310	22 310F	
	L5 567F	
311	52 114F	Print $\overset{.....}{y}_{i1}$ = f( )
	50 311F	
312	26 963F	
	50 520F	
313	7J 567F	Calculate and store $\ddddot{y}_{i2}$
	L4 511F	
314	40 532F	
	26 315F	
315	22 231F	Control to calculate and store $\dddot{y}_{i2}$
	50 505F	
316	71 536F	
	66 491F	
317	S5 F	
	40 526F	
318	50 504F	
	71 535F	
319	66 491F	
	S5 F	
320	40 425F	

LOCATION	ORDER	NOTES
	50 503F	
321	71 534F	
	66 491F	
322	S5 F	Calculate and store $\ddddot{y}_{12}$ = f( )
	40 524F	
323	50 502F	
	71 533F	
324	66 491F	
	S5 F	
325	40 523F	
	50 501F	
326	71 532F	
	66 491F	
327	S5 F	
	14 526F	
328	14 525F	
	14 524F	
329	14 523F	
	14 566F	
330	40 531F	
	26 231F	
331	50 520F	
	7J 531F	Calculate and store $\ddddot{y}_{13}$
332	14 511F	
	40 539F	
333	26 249F	Control to calculate and store $\dddot{y}_{13}$
	26 334F	
334	50 505F	
	71 543F	
335	66 491F	
	S5 F	
336	40 526F	
	50 504F	

LOCATION	ORDER	NOTES
337	71 542F	
	66 491F	
338	S5 F	
	40 525F	
339	50 503F	
	71 541F	
340	66 491F	
	S5 F	
341	40 524F	
	50 502F	
342	71 540F	
	66 491F	Calculate and store $\ddddot{y}_{13} = f(\ )$
343	S5 F	
	40 523F	
344	50 501F	
	71 539F	
345	66 491F	
	S5 F	
346	L4 526F	
	L4 525F	
347	L4 524F	
	L4 523F	
348	L4 566F	
	40 538F	
349	50 519F	Calculate and store $\ddddot{y}_{14}$
	7J 538F	
350	L5 511F	
	40 546F	
351	26 266F	Control to calculate and store $\dddot{y}_{14}$
	26 352F	
352	50 505F	
	71 550F	
353	66 491F	
	S5 F	

LOCATION	ORDER	NOTES
354	40 526F	
	50 504F	
355	71 549F	
	66 491F	
356	S5 F	
	40 525F	
357	50 503F	
	71 548F	
358	66 491F	Calculate and store $\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{14}$ = f( )
	S5 F	
359	40 524F	
	50 502F	
360	71 547F	
	66 491F	
361	S5 F	
	40 523F	
362	50 501F	
	71 546F	
363	66 491F	
	S5 F	
364	L4 526F	
	L4 525F	
365	L4 524F	
	L4 523F	
366	L4 566F	
	40 545F	
367	11 39F	
	L5 531F	
368	L4 538F	
	00 1F	
369	L4 567F	Calculate and store $\Delta\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{1}$
	L4 545F	
370	40 552F	
	50 552F	

LOCATION	ORDER	NOTES
371	7J 521F	
	40 557F	
372	26 283F	Control to calculate and store $\Delta\dddot{y}_i$
	26 373F	
373	S5 F	Begin subroutine to calculate the driving function
	46 375F	
374	L4 493F	
	42 651F	Set link
375	L5 ( )F	Store t_{ij} in location for driving function calculation
	40 527F	
376	41 566F	Clear driving function storage
	26 377F	
377	L5 568F	Is t included in the driving function?
	L0 493F	
378	36 379F	(Yes)
	22 381F	(No) Transfer to t^2 test
379	50 527F	
	75 575F	
380	66 491F	Calculate and store kt
	S5 F	
381	40 566F	
	L5 569F	Is t^2 included in the driving function?
382	L0 493F	
	32 383F	(yes)
383	22 387F	(no) Control to t^3 test
	50 527F	
384	75 576F	
	66 491F	
385	75 527F	Calculate and store kt^2
	66 491F	
386	S5 F	
	L4 566F	
387	40 566F	
	L5 583F	Is t^3 in the driving function?

LOCATION	ORDER	NOTES
388	LO 493F	
	32 389F	(yes)
389	26 395F	(no) Control to ln t test
	50 527F	
390	75 584F	
	66 491F	
391	75 527F	
	66 491F	Calculate and store kt^3
392	75 527F	
	66 491F	
393	S5 F	
	L4 566F	
394	40 566F	
	26 395F	
395	L5 570F	Is ln t included in the driving function?
	LO 493F	
396	36 397F	(yes)
	26 409F	(no) Transfer to log t test
397	L5 527F	
	LO 493F	
398	36 399F	
	26 425F	
399	22 399F	
	50 585F	
400	75 527F	
	66 491F	
401	S5 F	
	26 402F	
402	50 F	Calculate and store k ln k_1t
	50 402F	
403	26 923F	
	10 6F	
404	7J 497F	
	L4 499F	

LOCATION	ORDER	NOTES
405	40 587F	
	50 587F	
406	75 577F	
	66 491F	
407	S5 F	
	L4 566F	
408	40 566F	
	26 409F	
409	L5 571F	Is log t included in the driving function?
	L0 493F	
410	36 411F	(yes)
	26 425F	(no) Control to e^t test
411	L5 527F	
	L0 493F	
412	36 413F	
	26 425F	
413	22 413F	
	50 586F	
414	75 527F	
	66 491F	
415	S5 F	
	26 416F	
416	50 F	
	50 416F	
417	26 923F	Obtain and store k log k_1t
	10 6F	
418	7J 497F	
	L4 499F	
419	40 587F	
	50 587F	
420	7J 498F	
	40 588F	
421	50 588F	
	7J 578F	

LOCATION	ORDER	NOTES
422	66 491F	
	S5 F	
423	L4 566F	
	40 566F	
424	26 425F	
	26 425F	
425	L5 572F	Is e^t included in the driving function?
	L0 493F	
426	36 427F	(yes)
	26 444F	(no) Control to $t^{1/p}$ test
427	41 4F	
	L5 527F	
428	L0 493F	
	36 431F	
429	L5 579F	
	L4 566F	
430	40 566F	
	26 444F	
431	50 527F	
	71 589F	
432	66 489F	
	S5 F	
433	50 F	
	50 433F	
434	26 902F	
	40 591F	Obtain and store k e^{k_1t}
435	40 590F	
	50 591F	
436	7J 590F	
	40 591F	
437	L5 4F	
	L4 493F	
438	40 4F	
	L0 488F	

LOCATION	ORDER	NOTES
439	36 440F	
	22 435F	
440	L5 491F	
	66 591F	
441	7J 579F	
	66 491F	
442	S5 F	
	L4 566F	
443	40 566F	
	26 444F	
444	L5 492F	Is $t^{1/p}$ included in the driving function?
	L0 493F	
445	36 446F	(yes)
	26 474F	(no) Control to sin t test
446	L5 595F	
	00 20F	
447	46 449F	
	22 448F	
448	22 448F	Obtain and store $t^{1/p}$
	L5 527F	
449	50 (p)F	
	50 449F	
450	26 878F	
	40 594F	
451	L5 494F	
	L0 595F	
452	36 453F	
	26 456F	
453	50 594F	
	7J 487F	
454	40 594F	
	26 470F	
455	26 456F	
	26 456F	

LOCATION	ORDER	NOTES
456	L5 495F	
	L0 595F	
457	36 458F	
	26 462F	
458	50 594F	
	75 486F	
459	66 485F	
	S5 F	
460	40 594F	
	26 470F	
461	26 462F	Obtain and store k $t^{1/p}$
	26 462F	
462	L5 496F	
	L0 595F	
463	36 464F	
	26 467F	
464	50 594F	
	7J 486F	
465	40 594F	
	26 470F	
466	26 467F	
	26 467F	
467	50 594F	
	75 491F	
468	66 484F	
	S5 F	
469	40 594F	
	26 470F	
470	50 594F	
	75 593F	
471	66 491F	
	S5 F	
472	22 472F	
	40 566F	

LOCATION	ORDER	NOTES
473	26 474F	
	26 474F	
474	L5 574F	Is sin t included in the driving function?
	LO 493F	
475	36 476F	(yes)
	26 622F	(no) Control to cos t test
476	50 527F	
	75 596F	
477	66 491F	For k sin k_1t calculation, obtain and store k_1t x 10^{-4}
	S5 F	
478	40 597F	
	26 600F	
479	26 600F	
	26 600F	
480	26 600F	
	26 600F	
481	00 F	
	00 314159265358J	(x 10^{-1})
482	00 F	
	00 314159265J	(x 10^{-4})
483	00 F	
	00 628318530J	(2 x 10^{-4})
484	00 F	
	00 158489319250J	(antilog 16/5 x 10^{-4})
485	00 F	
	00 464158882664J	(antilog 8/3 x 10^{-3})
486	00 F	
	00 1000000000J	(10^{-3})
487	00 F	
	00 10000000000J	(10^{-2})
488	00 F	
	00 999F	
489	00 F	
	00 10000000J	(10^{-5})

```
LOCATION     ORDER                  NOTES

  490        00 F
             00 166666666666J
  491        00 F
             00 100000000J         (10^-4)
  492        00 F
             00 (order no.)F       (Set by the compiler)
  493        00 F
             00 1F
  494        00 F
             00 2F
  495        00 F
             00 3F
  496        00 F
             00 4F
  497        00 F
                                      64
             00 4436041956J        ( ----- log_e2)
                                    10000
  498        00 F
             00 434294481903J      ( 10^-4 log_e10^-4 )
  499        00 F
             00 921034037J         ( 10^-4 log_e10^-4 )
             00 600K
  600        L5 597F
             L0 483F
  601        40 598F
             32 600F
  602        L4 482F
             40 599F
  603        36 609F
             L4 482F
  604        22 604F
             66 481F
  605        S5 F
             66 486F
  606        S5 F
```

```
LOCATION     ORDER                NOTES

             40 599F
  607        26 616F
             26 616F
  608        26 616F
             26 616F
  609        L5 482F
             L0 599F
  610        40 599F         Obtain and store k sin k1t
             L5 599F
  611        66 481F
             S5 F
  612        L0 486F
             36 615F
  613        L4 486F
             66 486F
  614        S1 F
             26 616F
  615        51 491F
             22 617F
  616        50 F
             50 616F
  617        26 848F
             40 565F
  618        50 565F
             7J 580F
  619        26 620F
             26 620F
  620        L4 566F
             40 566F
  621        26 622F
             26 622F
  622        L5 573F         Is cos t included in the driving
             L0 493F         function?
  623        36 624F         (yes)
```

LOCATION	ORDER	NOTES
	22 649F	(no) Control to add the constant term
624	50 527F	
	75 564F	
625	66 491F	For k cos k_1t calculation, obtain and store k_1t x 10^{-4}
	S5 F	
626	40 597F	
	26 627F	
627	L5 597F	
	L0 483F	
628	40 598F	
	32 627F	
629	L4 482F	
	40 599F	
630	36 636F	
	L4 482F	
631	22 631F	
	66 481F	Obtain and store k cos k_1t
632	S5 F	
	66 486F	
633	S5 F	
	40 599F	
634	26 644F	
	26 644F	
635	26 644F	
	26 644F	
636	L5 482F	
	L0 599F	
637	40 599F	
	L5 599F	
638	66 481F	
	S5 F	
639	L0 486F	
	36 642F	
640	L4 486F	

```
LOCATION     ORDER              NOTES

             66 486F
  641        S1 F
             26 644F
  642        51 491F
             22 647F
  643        26 644F
             26 644F
  644        50 F
             50 644F
  645        26 848F
             S5 F
  646        40 563F
             26 647F
  647        50 563F
             7J 581F
  648        26 649F
             26 649F
  649        40 562F
             L5 582F       Put constant in A
  650        L4 562F       Add k cos k1t
             L4 566F       Add previous driving function value
  651        40 566F       Store complete driving function
                           value
             22 ( )F       Set by (374); return to main rou-
             00 668K       tine
  668        00 F
             00 6F
  669        00 F
             00 5F
  670        L5 669F       Compiler begins; set counter to in-
             40 6F         dicate order of the derivative
                           whose coefficient is being read in
  671        41 7F         Clear differential equation order-
                           setting counter
             41 9F         Clear "end of coefficient read in"
                           counter
  672        50 8F
                           Bring in coefficient
             50 672F
```

LOCATION	ORDER	NOTES
673	26 937F	
	41 F	Clear A
674	L2 8F	Subtract coefficient from zero
	36 698F	Coefficient ≥ 0; control to reduce read-in counter
675	41 F	Has differential equation order no. been set?
	L0 7F	
676	36 677F	(no)
	26 679F	(yes) Control to determine location of coefficient storage
677	L5 6F	Set differential equation order no.
	40 492F	
678	L5 493F	Set counter to indicate order no. has been set
	40 7F	
679	L5 6F	
	L0 699F	
680	36 681F	If coefficient of 5th order term, store in 500
	22 682F	
681	L5 8F	
	40 500F	
682	26 698F	
	L5 6F	
683	L0 496F	
	32 684F	If coefficient of 4th order term, store in 501
684	26 686F	
	L5 8F	
685	40 501F	
	26 698F	
686	L5 6F	
	L0 495F	
687	36 688F	If coefficient of 3rd order term, store in 502
	26 690F	
688	L5 8F	
	40 502F	
689	26 698F	
	26 690F	

LOCATION	ORDER	NOTES
690	L5 6F	
	L0 494F	
691	36 692F	If coefficient of 2nd order term, store in 503
	22 693F	
692	L5 8F	
	40 503F	
693	26 698F	
	L5 6F	
694	L0 493F	
	32 695F	If coefficient of 1st order term, store in 504
695	26 697F	
	L5 8F	
696	40 504F	
	26 698F	
697	L5 8F	If coefficient of dependent variable, store in 505
	40 505F	
698	L5 6F	Reduce "number of the order" counter by one
	L0 493F	
699	40 6F	
	80 4F	Skip order no.
700	L5 9F	
	L4 493F	
701	40 9F	Test for end of coefficient read-in
	L0 668F	
702	36 704F	End of coefficient read-in (if positive)
	26 672F	Not the end; control to contimue read-in
703	26 704F	
	26 704F	
704	50 8F	Bring in coefficient of driving function and store temporarily
	50 704F	
705	26 937F	
	26 707F	
706	26 707F	
	26 707F	

LOCATION	ORDER	NOTES
707	81 4F	Bring in the driving function indicator
	00 36F	
708	26 709F	
	26 709F	
709	36 717F	Begin sensing on a_0 to determine the driving functions
	00 1F	Continue sensing on a_0
710	32 738F	
	L5 8F	
711	40 577F	Driving function = ln t; store coefficient of ln t and set indicator
	L5 493F	
712	40 570F	
	80 4F	Skip N in LN
713	50 585F	
	50 713F	Store coefficient of t
714	26 937F	
	80 4F	Skip T
715	26 704F	Bring in next driving function
	26 717F	
716	26 717F	
	26 717F	
717	00 1F	
	36 734F	Continue sensing on a_o
718	00 2F	
	36 730F	
719	81 4F	Driving function = t^p; bring in p
	40 9F	
720	L0 495F	
	32 727F	
721	L5 9F	Determine p value
	L0 494F	
722	36 725F	
	L5 493F	Driving function = t; set indicator and store coefficient of t
723	40 568F	
	L5 8F	

LOCATION	ORDER	NOTES
724	40 575F	
	26 704F	Bring in next driving function
725	L5 493F	
	40 569F	Driving function = t^2; set indicator and store coefficient of t^2
726	L5 8F	
	40 576F	
727	26 704F	Bring in next driving function
	L5 493F	
728	40 583F	Driving function = t^3; set indicator and store coefficient of t^3
	L5 8F	
729	40 584F	
	26 704F	Bring in next driving function
730	L5 493F	
	40 592F	Driving function = $t^{1/p}$; set indicator and store coefficient of $t^{1/p}$
731	L5 8F	
	40 593F	
732	81 4F	Bring in p and store
	40 595F	
733	80 4F	Skip T
	26 704F	Bring in next driving function
734	L5 493F	
	40 572F	Driving function = e^{kt}; set indicator and store coefficient of e^{kt}
735	L5 8F	
	40 579F	
736	50 589F	
	50 736F	Read in and store coefficient of t in e^{kt}
737	26 937F	
	80 4F	Skip T
738	26 704F	Bring in next driving function
	00 1F	
739	32 745F	Contimue sensing on a_0
	00 1F	
740	36 751F	
	L5 493F	

LOCATION	ORDER	NOTES
741	40 574F	Driving function = sin t; set indicator and store coefficient of sin t
	L5 8F	
742	40 580F	
	80 8F	Skip IN (SIN sensed on S)
743	50 596F	Read in and store coefficient of t in sin kt
	50 743F	
744	26 937F	
	80 4F	Skip T
745	26 704F	Bring in next driving function
	80 4F	Skip S in COS (sensed on 0)
746	L5 493F	Driving function = cos t; set indicator and store coefficient of cos t
	40 573F	
747	L5 8F	
	40 581F	
748	50 564F	Read in and store coefficient of t in cos kt
	50 748F	
749	26 937F	
	80 4F	Skip T
750	26 704F	Bring in next driving function
	26 751F	
751	L5 8F	Store constant
	40 582F	
752	26 754F	
	26 754F	
753	26 754F	
	26 754F	
754	80 4F	Skip IC
	L5 492F	Test to determine where to store initial conditions
755	L0 669F	
	36 770F	Equation order = 5; store initial conditions accordingly
756	L5 492F	Differential equation order <5
	L0 496F	
757	36 768F	Equation order = 4;store initial conditions accordingly
	L5 492F	Differential equation order <4

LOCATION	ORDER	NOTES
758	L0 495F	
	36 766F	Equation order = 3; store initial conditions accordingly
759	L5 492F	Differential equation order <3
	L0 494F	
760	36 763F	Equation order = 2; store initial conditions accordingly
	26 761F	Equation order = 1; store initial conditions accordingly
761	50 515F	
	50 761F	
762	26 937F	
	26 772F	Control to read in Δt
763	50 514F	
	50 763F	
764	26 937F	
	26 772F	Control to read in Δt
765	26 776F	
	26 776F	
766	50 513F	
	50 766F	
767	26 937F	
	26 772F	Control to read in Δt
768	50 512F	
	50 768F	
769	26 937F	
	26 772F	Control to read in Δt
770	50 511F	
	50 770F	
771	26 937F	
	26 772F	
772	80 12F	Skip IN and R in INCR
	26 773F	
773	50 519F	
	50 773F	
774	26 937F	Transfer to beginning of differential equation solution routine
	22 20F	

LOCATION	ORDER	NOTES
	00 848K	
	Library Routine T6-S	Fast Sine-Cosine
	00 878K	
	Library Routine R2	Integral Root, $A^{1/p}$
	00 902K	
	Library Routine S4	Exponential, e^x
	00 923K	
	Library Routine S3	Logarithm
	00 937K	
	Library Routine N2	Input Fractions
	00 963K	
	Library Routine P1	Print Fractions
	24 670N	Transfer to beginning of compiler routine

VI. GENERAL INSTRUCTIONS FOR USE OF THE COMPILER ROUTINE

The Compiler routine assembles in the computer memory a program which, when executed by the computer, will provide the solution to an ordinary linear constant coefficient differential equation of any order up to and including the fifth. The equation "driving function" may consist of a constant plus any additive combination of the following functions (each used only once) multiplied by their respective coefficients. ln k_3t, e^{k_4t}, t, t^2, t^3, $t^{1/2}$ or $t^{1/3}$ or $t^{1/4}$ or $t^{1/5}$, sin k_1t, cos k_2t.

The solution consists of printed, punched, or plotted--as desired (and as available at the computer)--consecutive values of the independent variable, t, the dependent variable, e.g., y, and all the derivatives of y. These values begin with the initial conditions and continue with values at intervals of the independent variable corresponding to a previously-selected increment. After the computer begins the solution, it will contimue to yield values indefinitely or until computer "overflow" or "hang-up" occurs.

In order to obtain the solution to a differential equation coming within the category described above, carry out the following procedure:

1. Write, in descending order of the derivatives, the differential equation to be solved. If there is no constant in the driving function, add a zero at the end of the equation. Otherwise, add the constant at the end.

2. Divide all the terms by a constant such that the coefficients and constant term are all less than 10,000 and preferably lie

between 1 and 10. (The smaller these values are, the longer the computer can run before overflow or hang-up. See the discussion of limitations below.) Thereafter, divide all the coefficients and the constant term by 10,000. (The solution to the equation before the division by 10,000 is obtained by multiplying the computer output values by 10,000. Location of the decimal point in the print-out makes the corrected values available by inspection.) All the coefficients are now decimal fractions. The decimal point itself will not be carried into the computer but will be considered by the computer as lying immediately to the left of the numbers in the coefficients; therefore, retain all zeros to the left of other significant figures in the coefficients. Each coefficient must be preceded by its sign.

3. Re-write the equation:

3.1 Substitute the letter N followed by a number equal to the order of the derivative for each derivative symbol. Where the dependent variable or any order of the derivative from the first through the fifth is missing, substitute an N preceded by a +0 and followed by a number equal to the order of the derivative.

3.2 Substitute the symbols listed below for the driving functions:

sin k t................	FSIN k NT
cos k t................	FCOS k NT
ln k t.................	FLN k NT
e^{kt}..................	FE k NT

constant	FK
t	FT1
t^2	FT2
t^3	FT3
$t^{1/2}$	FR2T
$t^{1/3}$	FR3T
$t^{1/4}$	FR4T
$t^{1/5}$	FR5T

The signed coefficients of the driving functions precede these symbols. When the value k is included in a function such as in sin kt, cos kt, ln kt, or e^{kt}, this constant must be divided by 10,000 and preceded by its sign.

4. Place a slant sign (/) after the equation and then the letters "IC" to indicate that the initial conditions will follow. Write immediately following the letters "IC" the initial values (in descending order of the derivatives beginning with the derivative whose order is one less than the order of the equation) of the derivatives and the dependent variable, each divided by 10,000 and any other scaling constant used in step 2. Follow this by the initial value (divided by 10,000) of the independent variable. Precede each value by its sign. Put the letter "N" after the last value to indicate the end of the initial conditions.

5. Follow the initial conditions by another slant sign (/) and the letters "INCR." After these letters, put a plus sign and the value of the increment (unscaled). This value may be any fraction lying between 0 and +1. Any number so chosen is

considered by the computer to be a fraction having its decimal point immediately to the left of the digit(s) chosen. Again, put an "N" after the increment value to signify the end of the increment.

6. Put the Compiler routine on tape immediately followed by this equation, initial conditions, and the increment--in the format and in the order specified above.

7. The completed tape when fed to the computer will cause the computer to stop when it reaches the order 24 670. A "black-switch execute" then will start the computer emitting the series of values described in the second paragraph of these instructions. (An alternate method would be to feed only the Cpmpiler tape to the computer. When its "stop" order is reached, feed the equation tape to the computer reader and "execute.")

A typical example follows:

$$(1) \quad .05 \frac{d^4y}{dt^2} + \frac{d^2y}{dt} - 195 \frac{dy}{dt} + 10y = 100 \sin 2t + t^2 + 2$$

where $\frac{d^3y}{dt^3} = \frac{d^2y}{dt^2} = 0$ and $\frac{dy}{dt} = y = 1$ for $t = 0$; assume an increment of

0.1

$$(2a) \quad .0005 \frac{d^4y}{dt^2} + .01 \frac{d^2y}{dt^2} - 1.95 \frac{dy}{dt} + .1\ y = \sin 2t + .01\ t^2 + .02$$

$$(2b)\quad .00000005\,\frac{dy^4}{dt^2} + 0.000001\,\frac{dy^2}{dt^2} - .000195\,\frac{dy}{dt} + .00001\,y =$$

$$+.0001\sin 2t + .000001\,t^2 + .000002$$

```
(3)     +ON5 +00000005N4 +ON3 +000001N2 - 000195N1 +00001N0 =

        +0001FSIN+0002NT  +000001FT2  +000002FK/IC  +0  +0

        +000001  +000001  +ON/INCR  +1N
```

Equation (3) is the complete encoded equation.

The author has found it advantageous to have an equation-writing code check-off list. One frequently used follows:

1. Represent all five derivative orders and the independent variable.
2. Scale and put a sign on each coefficient.
3. Terminate signed initial conditions with "N."
4. Terminate a signed increment with "N."
5. Terminate scaled coefficients of t in the ln, sin, cos, and e functions with "N."
6. Add a zero if there is no constant in the driving function. Put the zero or the constant at the end of the encoded driving function together with its proper symbol.
7. Specify the proper number of initial conditions.
8. Correctly encode the driving functions.

When a print-out of the answers is obtained, it may be interpreted as follows:

1. The values of t, y, $\dot{y}$, $\ddot{y}$, $\dddot{y}$, $\ddddot{y}$, and $\overset{\cdot\cdot\cdot\cdot\cdot}{y}$ in this order are displayed beginning at the left of the page. All except the last two are located on one line. These two, $\ddddot{y}$ and $\overset{\cdot\cdot\cdot\cdot\cdot}{y}$, are printed in the middle of the line below. Eleven digits comprise each value.

2. The decimal point is located to the right of the fourth digit from the left—or at the break in the number presentation.

An example follows:

<u>**Print-out**</u>

```
0003  2000000    0209 4372338    -0000  0020318
```

<u>**Interpretation**</u>

t = 3.2   y = 209.4372338   y = -0.0020318

VII. MODIFICATION PROCEDURES AND FURTHER DEVELOPMENT OF THE COMPILER

Size of the storage facility of MISTIC has prevented making the foregoing compiler more general. It has been a purpose of this work, however, to point the general direction for writing a new, more comprehensive program or for expanding this one along the same lines when additional storage might become available. Both the Runge-Kutta method and its programming can be adapted (22) to the solution of more than one differential equation, and it is possible to write the same type of assembly program as that given herein by using similar techniques and allowing for more equations.

A program to allow for the solution of differential equations of higher order may be obtained readily by following the pattern set down in the foregoing Differential Equation routine. Actual programming involves merely the addition of higher order derivative-calculating paths (see Figures 1a and 1b) in parallel with those already existing—beginning at each order test and duplicating (except for higher order values) the last path already there. To provide the new path, modify the last-added path in the same way it represents a modification of the path just before it.

The Driving Function routine is an "open-ended" type of program in which additional function-calculating routines are brought into the program as it is made successively to sense certain locations for "1" or "0." Thus additional driving functions may be added readily by merely continuing the present program and then directing it to sense (at the appropriate time) the positive or negative value of certain storage. If the storage is positive at that time control may be transferred to some new function-calculating routine as desired. This my be done with the present program which still does not utilize some seventy-five storage locations. (This storage appears not to be enough to allow for sixth order equations, also, but may readily handle several driving functions, depending upon the length of the program required to calculate them. The Assembly routine might also be modified within this storage to handle the additional function if desired.)

Close inspection of the organization of the machine memory in the appendix is suggested as an initial step in any modification program. Simplification of writing the routines presented herein was facilitated by its detailed planning.

APPENDIX

Organization of the Memory

Location	Contents
0	Subroutines' temporary storage space
1	
2	
3	
4	
5	
6	Order of the derivative whose coefficient is being read in
7	Order indicator setting
8	Temporary coefficient storage
9	No. of coefficients which have been read in
10	
11	
12	
.	
.	
.	
20	Differential Equation Routine begins
.	
.	
499	Part 1 of Differential Equation Routine ends

Location	Contents
500	Coefficient of $\overset{\cdot\cdot\cdot\cdot\cdot}{y}$ (x 10^{-4})
501	" " $\ddddot{y}$ "
502	" " $\dddot{y}$ "
503	" " $\ddot{y}$ "
504	" " $\dot{y}$ "
505	" " y "
506	
507	
508	
509	
510	
511	Initial condition, $\overset{\cdot\cdot\cdot\cdot\cdot}{y}_0$ (x 10^{-4}) $= \overset{\cdot\cdot\cdot\cdot\cdot}{y}_{11}$
512	" " $\ddddot{y}_0$ " $= \ddddot{y}_{11}$
513	" " $\ddot{y}_0$ " $= \ddot{y}_{11}$
514	" " $\dot{y}_0$ " $= \dot{y}_{11}$
515	" " y_0 " $= y_{11}$
516	" " t_0 " $= t_{11}$
517	Δt (x 10^{-4})
518	$\frac{\Delta t}{2}$ (x 10^{-4})
519	Δt (not multiplied by 10^{-4})
520	$\frac{\Delta t}{2}$ " " " "
521	$\frac{\Delta t}{6}$ " " " "
522	Temporary storage for $\ddddot{y}$ times its coefficient (x 10^{-4}) in calculation of the highest derivative
523	Temporary storage for $\dddot{y}$ times its coefficient (x 10^{-4}) in calculation of the highest derivative
524	Temporary storage for $\ddot{y}$ times its coefficient (x 10^{-4}) in calculation of the highest derivative
525	Temporary storage for $\dot{y}$ times its coefficient (x 10^{-4}) in calculation of the highest derivative

Location	Contents
526	Temporary sterage for y times its coefficient ($x\ 10^{-4}$) in calculation of the highest derivative
527	Storage for t_{i1}, t_{i2}, t_{i3}, er t_{i4} as required in calculation of the driving function
528	
529	
530	
531	$\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{i2}$ ($x\ 10^{-4}$)
532	$\ddddot{y}_{i2}$ "
533	$\dddot{y}_{i2}$ "
534	$\ddot{y}_{i2}$ "
535	$\dot{y}_{i2}$ "
536	y_{i2} "
537	t_{i2} "
538	$\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{i3}$ "
539	$\ddddot{y}_{i3}$ "
540	$\dddot{y}_{i3}$ "
541	$\ddot{y}_{i3}$ "
542	$\dot{y}_{i3}$ "
543	y_{i3} "
544	t_{i3} "
545	$\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{i4}$ "
546	$\ddddot{y}_{i4}$ "
547	$\dddot{y}_{i4}$ "
548	$\ddot{y}_{i4}$ "
549	$\dot{y}_{i4}$ "
550	y_{i4} "
551	t_{i4} "

Location	Contents
552	$(\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{i1} + 2\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{i2} + 2\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{i3} + \overset{\cdot\cdot\cdot\cdot\cdot}{y}_{i4})$ $(x\ 10^{-4})$
553	$(\ddddot{y}_{i1} + 2\ddddot{y}_{i2} + 2\ddddot{y}_{i3} + \ddddot{y}_{i4})$ $(x\ 10^{-4})$
554	$(\dddot{y}_{i1} + 2\dddot{y}_{i2} + 2\dddot{y}_{i3} + \dddot{y}_{i4})$ $(x\ 10^{-4})$
555	$(\ddot{y}_{i1} + 2\ddot{y}_{i2} + 2\ddot{y}_{i3} + \ddot{y}_{i4})$ $(x\ 10^{-4})$
556	$(\dot{y}_{i1} + 2\dot{y}_{i2} + 2\dot{y}_{i3} + \dot{y}_{i4})$ $(x\ 10^{-4})$
557	$\Delta\ddddot{y}_i$ $(x\ 10^{-4})$
558	$\Delta\dddot{y}_i$ "
559	$\Delta\ddot{y}_i$ "
560	$\Delta\dot{y}_i$ "
561	Δy_i "
562	Value of k_1 cos kt $(x\ 10^{-4})$
563	Value of cos kt $(x\ 10^{-4})$
564	Coefficient of t in driving function = cos kt $(x\ 10^{-4})$
565	Value of sin kt $(x\ 10^{-4})$ (see 597-599)
566	Value of the driving function $(x\ 10^{-4})$
567	$\overset{\cdot\cdot\cdot\cdot\cdot}{y}_{i1}$
568	If a 1 is stored here, t is a driving function
569	" " 1 " " " , t^2 is a driving function
570	" " 1 " " " , ln t is a driving function
571	" " 1 " " " , log t is a driving function
572	" " 1 " " " , e^t is a driving function
573	" " 1 " " " ,cos t is a driving function
574	" " 1 " " " ,sin t is a driving function
575	Coefficient of t ($x\ 10^{-4}$)
576	" " t^2 "
577	" " ln t "

Location	Contents
578	Coefficient of log t (x 10^{-4})
579	" " e^{kt} "
580	" " sin t "
581	" " cos t "
582	Constant term
583	If a 1 is stored here, t^3 is a driving function
584	Coefficient of t^3 (x 10^{-4})
585	k_1, coefficient of t in ln k_1t driving function (x 10^{-4})
586	k_1, coefficient of t in log k_1t driving function "
587	ln k_1t (x 10^{-4})
588	log k_1t "
589	k_1, coefficient of t, in e^{k_1t} (x 10^{-4})
590	$1/e^{k_1t\text{x}10^{-3}}$
591	Temporary storage for $(1/e^{kt\text{x}10^{-3}})^n$ and final storage for $1/e^{kt}$
592	If a 1 is stored here, $t^{1/p}$ is a driving function
593	Coefficient of $t^{1/p}$ (x 10^{-4})
594	(t x $10^{-4})^{1/p}$ or (10^{-4} $t^{1/p}$) as needed
595	Value of p in $t^{1/p}$ driving function
596	Coefficient of t in sin k_1t driving function (x 10^{-4})
597	k_1t x 10^{-4} in sin k_1t (or cos k_1t) driving function calculation
598	(kt x 10^{-4}) - $2n\pi$ x 10^{-4}
599	$\left\|\left[(kt \text{ x } 10^{-4}) - 2n\pi \text{ x } 10^{-4}\right]\right\|$ $+\pi$x 10^{-4} after the bracketed term becomes negative; alse the fraction of π represented by kt (sin k_1t or cos k_1t)
600	Part 2 of Differential Equation program begins
.	
.	
651	Part 2 ends

Location	Contents
.	
.	
.	
670	Assembly Routine begins
.	
.	
.	
773	Assembly Routine ends
.	
.	
.	
848-877	Fast Sine-Cosine Routine, T6-S
878-901	Integral Root, R2
902-922	Exponential, S4
923-936	Logarithm, S3
937-962	Decimal Fraction Input, N2
963-990	Print Routine, P1
.	
999-1023	Decimal Order Input, X1

BIBLIOGRAPHY

1. M. Mandl,"Fundamentals of Digital Computers," Prentice-Hall, Inc., Englewood Cliffs, N. J.; 1958.

2. C. L. Johnson, "Analog Computer Techniques," McGraw-Hill Book Co., Inc., New York, N. Y. ; 1956.

3. R. K. Richards,"Arithmetic Operations in Digital Computers," D. Van Nostrand Co., Inc., Princeton, N. J.; 1955.

4. R. E. Sprague, "Fundamental Concepts of the Digital Differential Analyzer Method of Computation," Mathematical Tables and Other Aids to Computation, vol. 6, pp 41-49; 1952.

5. R. G. Selfridge, "Coding a General-Purpose Digital Computer to Operate as a Differential Analyzer," Proc. Western Joint Computer Conference, March, 1955.

6. F. H. Lesh, and F. G. Curl, "DEPI: An Interpretive Digital-Computer Routine Simulating Differential-Analyzer Operations," Memorandum No. 20-141, Jet Propulsion Laboratory, Cal, Inst. of Tech., Pasadena.

7. F. Lesh, "Methods of Simulating a Differential Analyzer on a Digital Computer," Jour, Ass. Computing Machinery, vol. 5; July, 1958.

8. M. L. Stein, Jack Rose, and D.B. Parker, "A Compiler with an Analog-Oriented Input Language," Proc. Western Joint Computer Conference; March, 1959.

9. M. L. Stein, and Jack Rose, "The Automatic Deduction of Differential Equations from Analog Set-Up Diagrams," Mathematical Pre-Print No.

13, Convair Astronautics, San Diego, California.

10. G. M. Hopper, "Automatic Coding Techniques-1955," Proc. High Speed Computer Conference, Louisiana State University; Feb.,1956.

11. ________. "Tomorrow - Automatic Programming," Petroleum Refiner - Special Computer Report, vol. 36 no. 2; Feb., 1957.

12. ________. "Automatic Programming for Business Applications," Computers and Automation,vol 7 no. 2; Feb.,1958.

13. J. L. Meriam,"Procedure for the Machine or Numerical Solution of Ordinary Linear Differential Equations for Two-Point Linear Boundary Values," Mathematical Tables and Other Aids to Computation, vol. 3, pp 532-539; 1948.

14. C. E. Froberg, " On the Solution of Ordinary Differential Equations with Digital Computing Machines," Kungl. Fysiografiska Sällskapets i Lund Förhandlingar (Proc. Roy. Physiog. Soc. Lund) vol. 20, pp 136-152; 1950.

15. S. Gill, "A Process for the Step-by-Step Integration of Differential Equations in an Automatic Digital Computing Machine," Proc. Cambridge Phil. Soc., vol. 47, pp 96-108; 1950.

16. M. K. Gavurin, "On a Method of Numerical Integration of Homogeneous Linear Differential Equations Convenient for Mechanization of the Computation," Trudy Mat. Inst. Steklov. 28, pp 152-156; 1949.

17. C. F. Gradwell, "Solution of Differential Equations on an Electronic Digital Computer," Engineer, vol. 196, pp 36-39; July, 1953.

18. S. Gorn, "Real Solution of Numerical Equations by High-Speed Machines," Ballistic Research Laboratories Report, Aberdeen Proving Ground, Aberdeen, Md.; 1955.

19. L. Lukaszewicz, and P. Szeptycki, "Electronic Integration of Differ-

ential Equations, ELI," Zastos, Mat., vol. 2, pp 399-415; 1956.

20. P. D. Williams, "A Study of Numerical Methods for Solving Differential Equations," Proc. Ass. Computing Machinery, University of Illinois; 1958.

21. M. V. Wilkes, D. J. Wheeler, and S. Gill,"Programs for an Electronic Digital Computer," Addison-Wesley Publishing Co., Inc., Reading, Mass.; 1957.

22. K. J. Nielsen,"Methods in Numerical Analysis," The Macmillan Co., New York; 1956.

23. D. D. McCracken, "Digital Computer Programming," John Wiley & Sons, Inc., New York; 1957.