C.3

JAMES A. PHILLIPS

LA-3292

LOS ALAMOS SCIENTIFIC LABORATORY
LOS ALAMOS of the NEW MEXICO
University of California

A FORTRAN Version of Nordsieck's Scheme for the Numerical Integration of Differential Equations

UNITED STATES
ATOMIC ENERGY COMMISSION
CONTRACT W-7405-ENG. 36

Printed in USA. Price $ 2.00. Available from the

Clearinghouse for Federal Scientific
and Technical Information,
National Bureau of Standards,
U. S. Department of Commerce,
Springfield, Virginia

LA-3292
UC-32, MATHEMATICS
AND COMPUTERS
TID-4500 (41st Ed.)

LOS ALAMOS SCIENTIFIC LABORATORY
LOS ALAMOS of the NEW MEXICO
University of California

Report written: March 1965
Report distributed: June 22, 1965

A FORTRAN Version of Nordsieck's Scheme for the Numerical Integration of Differential Equations

by

H. R. Lewis, Jr. and E. J. Stovall, Jr.

ABSTRACT

Modifications of Nordsieck's scheme for numerically integrating differential equations are described which permit satisfactory operation in floating-point arithmetic. The major modification, recommended for both fixed-point and floating-point operation, is a reformulation of the test for numerical stability. Also discussed, in relation to the present floating-point scheme, are Nordsieck's use of guard digits and novel rounding techniques. A computer subroutine for the modified scheme is presented in the FORTRAN-II and FORTRAN-IV languages.

ACKNOWLEDGMENTS

We express our appreciation to R. M. Brown and P. Ponzo of the Coordinated Science Laboratory of the University of Illinois, and to E. P. Gray and J. G. Monteabaro of the Applied Physics Laboratory of the Johns Hopkins University, for kindly providing us with their floating-point versions of Nordsieck's integration scheme. The present work began as a result of using their computer programs.

CONTENTS

ABSTRACT 3

ACKNOWLEDGMENTS 4

INTRODUCTION 7

I. THE STABILITY TEST 8

II. GUARD DIGITS AND SPECIAL ROUNDING PROCEDURES 12

III. THE FORTRAN SUBROUTINES 14

IV. RESULTS OF SOME TEST PROBLEMS 18

REFERENCES 23

APPENDICES

IA. FORTRAN-II LISTING OF DEQ FOR IBM 7094 24

IB. FAP LISTING OF RNDN FOR IBM 7094 28

IIA. FORTRAN-IV LISTING OF DEQ FOR IBM 7094 29

IIB. MAP LISTING OF RNDN FOR IBM 7094 33

III. FORTRAN-IV LISTING OF DEQ FOR IBM 7030 34

IV. FORTRAN-IV LISTING OF SAMPLE PROGRAM 38

INTRODUCTION

One of the important problems in numerical analysis which arises in scientific and engineering research is the numerical integration of a system of differential equations. Because of the frequency of this problem, it is valuable to have a general-purpose numerical scheme with which the integration of a large class of differential equations can be reliably performed. One numerical scheme, useful as such a general purpose method, and well adapted for efficient use with digital computers, has been proposed by A. Nordsieck.[1] This scheme is designed to solve a system of first-order equations,

$$\frac{dy_i}{dx} = f_i(x, y_1, y_2, \ldots, y_n), \; i = 1, 2, \ldots, n, \tag{1}$$

with given initial conditions, whenever the f_i are such that a unique solution exists. The basic formulas of the method are equivalent to finding the fifth-degree polynomial approximation to the desired solution of the differential equations which is determined from the values of y_i and f_i at the current value of the independent variable x, and from the values of f_i at the four preceding values of x. The effective approximating polynomial is identical with that of the Adams method of integration.[2,3] However, Nordsieck has reformulated and modified the Adams method in a way which is of interest for practical application.

An important practical feature of Nordsieck's scheme is the automatic increase and decrease of the elementary interval size during the course of the integration. This is accomplished by means of two tests that are performed at each elementary integration step. One test determines an approximate bound of the truncation error in the solution. The other test is intended to guarantee that the integration scheme be numerically stable throughout the integration; that is, those solutions of the equations of the numerical method which are not related to the differential equations are supposed to be damped out if this test is always satisfied.

Nordsieck formulated the integration scheme for fixed-point arithmetic. The purpose of this note is to explain modifications of the original formulation which allow satisfactory operation with floating-point arithmetic. The major modification, discussed in the first section of this report, is a reformulation of the stability test itself in order to correct a flaw in the original formulation; this modification is recommended for both fixed-point and floating-point operation. The special procedures proposed by Nordsieck (novel rounding techniques and the use of guard digits), which were helpful in avoiding malfunctions of the test, as it was originally formulated, are discussed in Sec. II. In Sec. III, the calling sequence and use of a FORTRAN computer subroutine for the modified Nordsieck scheme are described.* Results of some test problems are given in Sec. IV. Listings of three versions of this subroutine, in the FORTRAN-II and FORTRAN-IV languages for the IBM 7094 and IBM 7030 computers, are given in appendices. In Appendix IV is the listing of a sample FORTRAN-IV program for integrating the equation for Legendre polynomials.

Modification of the stability test came as a result of two difficulties which were sometimes encountered with earlier floating-point versions of the integration scheme; these were 1) reduction of the elementary integration interval to unnecessarily or absurdly small values, and 2) unstable "blowup" of the solution.

Problems which have been done with floating-point versions of the integration scheme include those reported by Nordsieck[1] and, also, integration of the equations of motion of a charged particle in a magnetic field. The occurrence of difficulties was very much more frequent with the latter problem. With the current floating-point version of the integration scheme, all of the problems are done satisfactorily.

I. THE STABILITY TEST

In his original paper[1] Nordsieck proposes a sufficient condition for insuring stability of the numerical method with a comfortable margin of safety. In terms of the elementary integration interval h and the eigenvalues of the matrix $\partial f/\partial y$, whose elements are $\partial f_i/\partial y_j$, the

*The programming details of this subroutine are, in large measure, based on the computer program of an earlier floating-point version of Nordsieck's integration scheme which was kindly made available by the Coordinated Science Laboratory of the University of Illinois. That program, dated October 1, 1963 and designated D2 (F) UOFI DEQ, was part of the 1604 computer library of the Coordinated Science Laboratory. The program of another earlier version, dated October 26, 1962 and designated BCC Library Routine No. 5.02.06, was made available to us by the Applied Physics Laboratory of the Johns Hopkins University.

condition is[4]

$$\frac{95}{288}\,|h\lambda| \leqq \frac{1}{8}\,, \qquad (2)$$

for each eigenvalue λ. Nordsieck does not directly require satisfaction of this inequality. Instead, he proposes a test which is intended to insure satisfaction of the inequality, and which is more easily applied. However, his test does not guarantee the validity of the stability condition (2), except in the special case that only a single differential equation is to be solved.

Let y and f denote the column matrices whose elements are y_i and f_i, respectively. In the course of the iterative solution of the implicit equations of the scheme, three approximations to y are computed — first $y^{(1)}$, then $y^{(2)}$, and finally $y^{(3)}$. These column matrices are related to the square matrix $\partial f/\partial y$ through the approximate equality[6]

$$(y^{(3)} - y^{(2)}) \cong \frac{95}{288}\, h\, \frac{\partial f}{\partial y}\,(y^{(2)} - y^{(1)}). \qquad (3)$$

If there is only one differential equation to be solved ($n = 1$), then $y^{(1)}$, $y^{(2)}$, $y^{(3)}$, and $\partial f/\partial y$ are all single numbers, as opposed to true matrix quantities, and we have

$$|y^{(3)} - y^{(2)}| \cong \frac{95}{288}\,\left|h\left(\frac{\partial f}{\partial y}\right)\right|\,|y^{(2)} - y^{(1)}|. \quad (n = 1) \qquad (4)$$

The test to insure satisfaction of the stability condition (2) proposed by Nordsieck for the case $n = 1$ is[6]

$$|y^{(3)} - y^{(2)}| \leqq \frac{1}{8}\,|y^{(2)} - y^{(1)}|. \qquad (5)$$

Indeed, for $n = 1$, the inequalities (2) and (5) are identical to within the approximation that Eq. (3) is an exact equality. (If $n = 1$, then $\lambda = \partial f/\partial y$.)

However, if there are two or more equations to be solved simultaneously ($n > 1$), then the situation is different. Let a norm of a matrix be denoted by enclosing the matrix symbol between double vertical bars; then, with suitably chosen norms, the relation corresponding to Eq. (4)

is*

$$\|y^{(3)} - y^{(2)}\| \leqq \frac{95}{288} \left\| h \frac{\partial f}{\partial y} \right\| \|y^{(2)} - y^{(1)}\|. \quad (n > 1) \tag{6}$$

Let λ_{max} be that eigenvalue of $\partial f/\partial y$ which has the largest magnitude. A standard inequality relating $\|\partial f/\partial y\|$ and $|\lambda_{max}|$ is[7]

$$|\lambda_{max}| \leqq \left\| \frac{\partial f}{\partial y} \right\|. \tag{7}$$

The stability test proposed by Nordsieck for the case $n > 1$ may be written as[6]

$$\|y^{(3)} - y^{(2)}\| \leqq \frac{1}{8} \|y^{(2)} - y^{(1)}\|, \tag{8}$$

where the column matrix norm used is either the largest magnitude of any element, or the euclidean norm. Combining relations (6), (7) and (8), we have

$$\frac{95}{288} |h\lambda_{max}| \leqq \frac{95}{288} \left\|h \frac{\partial f}{\partial y}\right\| \geqq \frac{\|y^{(3)} - y^{(2)}\|}{\|y^{(2)} - y^{(1)}\|} \leqq \frac{1}{8} . \tag{9}$$

It is to be noted that no bound whatsoever is obtained on $\|h\ \partial f/\partial y\|$ or on $|h\lambda_{max}|$. In fact, there are numerical examples for which the inequality (8), which represents the test, is satisfied while the inequality (2), which represents the stability condition, is violated. A two-dimensional case illustrating this point is the following. Let

$$\frac{95}{288} h \frac{\partial f}{\partial y} = \frac{1}{4} \begin{pmatrix} -1 & 2 \\ -3 & 4 \end{pmatrix} \text{ and}$$

$$(y^{(2)} - y^{(1)}) = \begin{pmatrix} 3 \\ 2 \end{pmatrix} \text{ ; then, using (3),}$$

*If the square matrix and column matrix norms are so chosen that (6) is valid, then these norms are said to be consistent with one another. Norms are usually chosen in this way; see, for example, Ref. 7.

$$(y^{(3)} - y^{(2)}) = \frac{1}{4}\begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

With these choices, we have

$$\frac{|y^{(3)} - y^{(2)}|}{|y^{(2)} - y^{(1)}|} = \begin{cases} \frac{1}{12}, & \text{if the "maximum element" norm is used.} \\ \frac{1}{\sqrt{104}}, & \text{if the euclidean norm is used.} \end{cases}$$

With either type of norm, we have

$$\frac{|y^{(3)} - y^{(2)}|}{|y^{(2)} - y^{(1)}|} < \frac{1}{8}, \text{ and the test (8) is satisfied.}$$

On the other hand, the eigenvalues of

$$\frac{95}{288} h \frac{\partial f}{\partial y}$$

are 1/4 and 1/2, so that

$$\frac{95}{288} h\lambda_{max} = \frac{1}{2} > \frac{1}{8},$$

which does not satisfy the stability condition (2). Thus, it is evident that the original formulation of a test to insure satisfaction of the stability condition (2), and therefore to insure stability, is incorrect except for the special case $n = 1$.*

The stability test used in the present FORTRAN version of the scheme is precisely the basic stability condition expressed by (2). That is, the elements of the matrix $\partial f/\partial y$ are evaluated — analytically if possible, numerically otherwise — and either an upper bound of the

*A. Nordsieck (private communication) agrees that the stability test as originally formulated does not insure satisfaction of the stability condition (2).

magnitudes of the eigenvalues, or the largest eigenvalue itself, is computed.

The original stability test was subject to malfunction because of round-off noise. This difficulty was alleviated, in the fixed-point version, by use of guard digits and novel rounding techniques. Such round-off noise problems do not interfere with the new stability test.

II. GUARD DIGITS AND SPECIAL ROUNDING PROCEDURES

In the fixed-point version of this integration scheme, with the stability test in the form given by expression (9), Nordsieck found it desirable to carry more digits in y than in f — so-called guard digits.[8] The number of extra digits was the integer nearest to $\log_\beta(|h|^{-1})$. (β is the base of the number system with which computations are performed. For example, $\beta = 2$ with binary arithmetic.) The reason given for this is to minimize the accumulation of round-off error in y when the number of elementary steps is large. A different reason for keeping the guard digits is that a certain form of round-off noise then tends not to interfere with the functioning of the original stability test (8). This can be seen in the following way. From the equations of the integration scheme,[1] it is easily derived that the differences, $(y^{(3)} - y^{(2)})$ and $(y^{(2)} - y^{(1)})$, can be expressed as

$$y^{(3)} - y^{(2)} = \frac{95}{288} h \left\{ f\left[x + h, y^{(2)}(x + h)\right] - f\left[x + h, y^{(1)}(x + h)\right]\right\} \tag{10a}$$

and

$$y^{(2)} - y^{(1)} = \frac{95}{288} h \left\{ f\left[x + h, y^{(1)}(x + h)\right] - f^P \right\} . \tag{10b}$$

The $y^{(i)}$ are approximations to y for the independent variable equal to x + h; f^P, the "predicted" value of f, is a first approximation to the value of f at x + h. Using Eqs. (10a) and (10b), the inequality (8) can be rewritten as

$$\left\| f\left[x + h, y^{(2)}(x + h)\right] - f\left[x + h, y^{(1)}(x + h)\right] \right\| \le \frac{1}{8} \left\| f\left[x + h, y^{(1)}(x + h)\right] - f^P \right\| . \tag{11}$$

When the test is written in this form, it is evident that round-off noise in the computed values of the derivatives can interfere with the functioning of the test. This is accentuated by the fact that both of the differences in the inequality can be quite small. The error in f(x, y) due to an error in y , for the case n = 1, can be estimated as follows. Letting Δy be the error in y and Δf the corresponding error in f , we have, in first approximation,

$$|\Delta f| \simeq \left|\frac{\partial f}{\partial y}\right| \; |\Delta y| . \tag{12}$$

However, $|\partial f/\partial y|$ is bounded by the stability condition

$$\left|\frac{\partial f}{\partial y}\right| \leqq \frac{1}{8\left(\frac{95}{288}\right)|h|} . \tag{13}$$

Combining (12) and (13), we have

$$|\Delta f| \lesssim \frac{36}{95} \left|\frac{1}{h}\right| \; |\Delta y| = \beta^{\log_\beta\left(\frac{36}{95}\frac{1}{|h|}\right)} \Delta y . \tag{14}$$

In binary arithmetic ($\beta = 2$), for the case n = 1, an error equal to the least count in y will usually give rise to an error less than the least count in f , if $\log_\beta(1/|h|)$ more digits are carried in y than in f . Thus, round-off noise of the magnitude of the least count in y would tend not to interfere with application of a stability test in the form of expression (11).

Because the number of figures in a floating-point number in a digital computer does not change, guard digits cannot be easily used for stopping the propagation of round-off noise from y into f , even in the case n = 1. (An alternative would be to use double-precision arithmetic for y .) Therefore, it is fortunate that the modified stability test, which is identical with the stability condition (2), is not influenced by such noise. Neither guard digits nor double-precision arithmetic has been used in the floating-point version of Nordsieck's scheme.

A novel way of rounding certain quantities which appear in this integration scheme was introduced in the original fixed-point version; this type of rounding was called "rounding away from zero."[9] The purpose of this rounding was to eliminate a type of noise which sometimes interfered with the proper operation of the two tests that control the size of the elementary interval. An analogous floating-point procedure can also be devised. Such a procedure has been tried in the floating-point version of the integration scheme which incorporates the new stability test. However, with that version of the integration scheme, we have not observed any overall improvement in the operation of the interval control logic when the rounding procedures are included; nor have we observed any malfunction of the interval control logic when these special rounding techniques are omitted. The floating-point "rounding away from zero," at least when done with the FORTRAN language, is rather time consuming, with the result that the computer time necessary for a particular problem can be substantially longer with the special rounding techniques than without. For these reasons, "rounding away from zero" has been omitted from the floating-point version of the integration scheme.

III. THE FORTRAN SUBROUTINES

There are three FORTRAN versions of the modified Nordsieck scheme: One for the IBM 7094 in FORTRAN-II, one for the IBM 7094 in FORTRAN-IV, and one for the IBM 7030 in FORTRAN-IV. The calling sequence and use are the same for each version. The two versions for the IBM 7094 include normal rounding of certain quantities which is accomplished via a subroutine called RNDN. (For the FORTRAN-II version, this subroutine is written in the FAP language; for the FORTRAN-IV version, it is written in the MAP language.) The version for the IBM 7030 does not include any rounding. The other differences between the three versions are minor ones having to do with differences between the FORTRAN-II and FORTRAN-IV languages and between the IBM 7094 and IBM 7030 computers.

Listings of the FORTRAN-II version for the IBM 7094, the FORTRAN-IV version for the IBM 7094, and the FORTRAN-IV version for the IBM 7030 are given in Appendices IA, IIA, and III, respectively. Listings of the FAP and MAP versions of RNDN are in Appendices IB and IIB, respectively.

Suppose that the differential equations to be solved are

$$\frac{dy_i}{dx} = f_i(x, y_1, y_2, \ldots, y_n), \quad i = 1, 2, \ldots, n,$$

and that the integration is to proceed from $x = x_1$ to $x = x_2$. The calling sequence for the integration subroutine is:

```
CALL DEQ(F, X, XLIMIT, Y, ERROR, NEQ, H, HMAX, JUMP, KSTEP, KCON, CLIF)
```

The allowable maximum number of equations that can be solved by DEQ is the dimension of the dimensioned quantities. (They are equally dimensioned.) As listed in the appendices, DEQ can solve a maximum of 20 equations.

The meaning of the arguments in the argument list is:

F	An array of dimension n such that F(i) = f_i. This array must be computed by the calling program for subsequent use by DEQ. The F(i) need not be computed until after the first return from DEQ. (See JUMP, below.)
X	The current value of the independent variable x. When DEQ is first called, it must be called with X = x_1. Thereafter, the value of X is adjusted by DEQ, and it must not be changed by the calling program.
XLIMIT	The final value of the independent variable x. When DEQ is first called, it must be called with XLIMIT = x_2, and XLIMIT is not to be changed before the integration has proceeded to $x = x_2$.
Y	An array of dimension n such that Y(i) = y_i. When DEQ is first called, it must be called with Y(i) equal to the value of y_i at $x = x_1$; that is, DEQ is first called with the elements of Y equal to the initial values of the y_i. Thereafter, the Y(i)'s are adjusted by DEQ, and they must not be changed by the calling program. When the integration is completed, and X = XLIMIT, then the value of Y(i) equals the final value of y_i, i.e., the value of y_i at $x = x_2$.
ERROR	A positive number, supplied by the calling program, which is related to the absolute value of the error (as opposed to the relative error) which is introduced in the values of the y_i when the integration has increased the value of x by one unit ($x \rightarrow x + 1$). The precise numerical relation of ERROR to the accuracy of the integration cannot be given. However, ERROR is an approximate bound on the sum of the truncation errors incurred at each integration step during the process of integrating over a unit interval of the independent variable. In general, the error introduced at any given step will propagate forward in a way primarily determined by the nature of the differential equations being solved. Because of the propagation of errors, the sum of the truncation errors at each step is not generally equal to the error in the final result of integration due to truncation errors. (ERROR is the quantity β^{-e} which is discussed in Ref. 1. Further details, including

the relation of β^{-e} to other errors in the scheme, can be found there. Note that the discussion in Ref. 1 assumes fixed-point arithmetic.) It can be said that the test in DEQ in which ERROR occurs tends to adjust the integration step size in such way that a certain degree of accuracy in the final result is economically achieved. With a given set of equations to be solved, it is useful first to choose ERROR about equal to the absolute value of the error allowable after a unit step, and then to observe the effect on the final result of using values of ERROR both larger and smaller than the first value. In that way, it is possible to determine experimentally the approximate relation between the value of ERROR and the final accuracy. As ERROR is decreased it may eventually happen that the accuracy also decreases because of increasing importance of round-off error. Also, if ERROR is made small enough, then the elementary step size will be made too small to be significant. (See JUMP, below.)

NEQ	NEQ = n, the number of equations in the system to be solved.
H	Current value of the integration step size. When DEQ is first called, it must be called with H equal to some starting value. It is convenient to let the starting value equal HMAX. Thereafter, the value of H is adjusted by DEQ, and it must not be changed by the calling program.
HMAX	The maximum allowable absolute value of H equals the absolute value of HMAX. (HMAX may be of either sign, but only $\|HMAX\|$ is used.) HMAX should be chosen smaller than the width of any fine structure in the solution, to insure that the correct solution is followed over such fine structure.
JUMP	The values of JUMP are -1, 0, and 1. When DEQ is first called, it must be called with JUMP = -1. Thereafter, JUMP must not be altered by the calling program. Whenever control is returned to the calling program, the value of JUMP must be ascertained. If JUMP = -1, then H has become small enough that, to within the accuracy of the computer, X + H = X. The integration cannot continue if this occurs. If JUMP = 0, then new values of the F(i) must be computed with the current values of X and Y(i). Then, DEQ is called again, leaving JUMP = 0. (The F(i) need be computed only if control is returned to the calling program with JUMP = 0.) If JUMP = 1, then the integration to XLIMIT has ended. The integration can be carried further in the same direction by changing XLIMIT to a new final value of x, and again calling DEQ, leaving JUMP = 1.

KSTEP A running index which counts the number of elementary steps taken during the integration. If KSTEP is less than or equal to 24, then the starting sequence is being executed. If KSTEP is larger than 24, then the main integration procedure is being executed. If KSTEP reaches the largest integer that can be stored in the computer, then KSTEP is automatically set equal to 28 at the next step. This insures that the procedure will not revert to the starting sequence during the integration. KSTEP need not and should not be set by the calling program.

KCON An integer, equal to 0 or 1, which indicates whether or not a new value of CLIF is to be computed by the calling program, using the current values of X and Y(i). If KCON = 0, then CLIF is not to be computed. If KCON = 1, then CLIF is to be computed.

CLIF A number, computed by the calling program, which is used by DEQ for adjusting H to insure numerical stability of the method. CLIF is to be greater than or equal to the magnitude of the largest eigenvalue of the matrix whose elements are $a_{ij} = \partial f_i/\partial y_j$. That is, if $\lambda_i (i = 1, \ldots, n)$ is the $i^{\underline{th}}$ eigenvalue (possibly complex) of this matrix, then CLIF $\geqq |\lambda_i|_{max}$, so that CLIF is an upper bound of the magnitudes of the eigenvalues. The stability condition which is applied is expression (2) of Sec. I. Upper bounds of $|\lambda_i|_{max}$ are provided by the following two expressions:

$$|\lambda_i|_{max} \leqq \text{Max} \sum_k |a_{jk}|$$

$$|\lambda_i|_{max} \leqq \text{Max} \sum_j |a_{jk}|$$

These two bounds are generally different and may be significantly larger than $|\lambda_i|_{max}$. If it is very inconvenient to determine an upper bound, then an appropriate value for CLIF (one which makes the method stable) must be chosen intuitively or on the basis of experience.

In Appendix IV is the listing of a sample FORTRAN-IV program for integrating the equation for Legendre polynomials, in steps of 0.1, from an initial value of x to a final value not greater than 0.95.

IV. RESULTS OF SOME TEST PROBLEMS

In addition to results obtained with the four test problems described by Nordsieck[1] (Problems 3 through 6, below), we also report results with two other problems (Problems 1 and 2, below).

Problem 1

$$\frac{d^2y}{dx^2} = -y$$

This equation was integrated from $x = 0$ to $x = 10\pi$, with the initial conditions $y = 0$ and $dy/dx = 1$. The results, for three values of ERROR, are shown in Table I.

TABLE I

x	y	dy/dx	ERROR	Version
31.415926	$-.21389202 \times 10^{-5}$	.99999816	10^{-4}	FORTRAN-II, IBM 7094
	$-.89575723 \times 10^{-6}$	.99999980	10^{-6}	
	$-.10448896 \times 10^{-5}$	1.0000004	10^{-8}	
31.415926	$-.22733957 \times 10^{-5}$	.99999823	10^{-4}	FORTRAN-IV, IBM 7094
	$-.95797372 \times 10^{-6}$	.99999984	10^{-6}	
	$-.10048494 \times 10^{-5}$	1.0000004	10^{-8}	
31.415926	$-.18014973 \times 10^{-5}$	.99999809	10^{-4}	FORTRAN-IV, IBM 7030
	$-.56000062 \times 10^{-6}$	.99999998	10^{-6}	
	$-.53630424 \times 10^{-6}$	1.0000000	10^{-8}	

Problem 2

$$(1 - x^2)\frac{d^2y}{dx^2} - 2x\frac{dy}{dx} + \ell(\ell + 1)y = 0$$

The Legendre polynomial of order ℓ is a solution of this equation. The equation was integrated from $x = -.9$ to $x = .9$, with the initial conditions $y = P_\ell(-.9)$ and

$$\frac{dy}{dx} = \left.\frac{dP_\ell}{dx}\right|_{x = -.9} .$$

The results, for $\ell = 4$ and two values of ERROR, are shown in Table II. The initial values quoted in the table were computed from the exact formulas for $P_4(x)$ and dP_4/dx.

TABLE II

x	y	$(1 - x^2)$ dy/dx	ERROR	Version
-.9	.20793745	-1.1414249		FORTRAN-II, IBM 7094
.9	.20793735	1.1414254	10^{-3}	
	.20793742	1.1414250	10^{-6}	
-.9	.20793748	-1.1414249		FORTRAN-IV, IBM 7094
.9	.20793732	1.1414254	10^{-3}	
	.20793740	1.1414250	10^{-6}	
-.9	.20793750	-1.1414250		FORTRAN-IV, IBM 7030
.9	.20793743	1.1414255	10^{-3}	
	.20793750	1.1414250	10^{-6}	

Problem 3

$$\frac{dy}{dx} = \begin{cases} 0, & \text{for } |x - \tfrac{1}{2}| \geq 2^{-31} \\ 2^{5}, & \text{for } |x - \tfrac{1}{2}| < 2^{-31} \end{cases}$$

This equation was integrated on the IBM 7030 from $x = 0$ to $x = 1$, with $y(0) = 0$ and HMAX = 2^{-8}. The exact value of $y(1)$ is $2^{-5} = .03125$. The results, for three values of ERROR, are shown in Table III. The minimum value of the integration step size, H_{min}, which occurred during the integration and the value of KSTEP at the end of the integration are also given. (KSTEP-24) is the number of steps after the starting sequence. The width of the rectangular pulse is too small for the problem to be done on the IBM 7094. However, the equation

$$\frac{dy}{dx} = \begin{cases} 0, & \text{for } |x - \tfrac{1}{2}| \geq 2^{-22} \\ 2^{5}, & \text{for } |x - \tfrac{1}{2}| < 2^{-22} \end{cases}$$

was integrated on the IBM 7094 with results similar to those in Table III.

TABLE III

ERROR	KSTEP	H_{min}	y(1)
2^{-30}	440	2^{-37}	.0310059
2^{-34}	476	2^{-41}	.0312347
2^{-41}	538	2^{-48}	.0312497

Problem 4

$$\frac{dy}{dx} = 2^{7} \frac{(2^{-30})^2}{x^2 + (2^{-30})^2}$$

This analytic derivative is similar to the derivative in Problem 3. The equation was integrated on the IBM 7030 from $x = -1/2$ to $x = 1/2$, with $y(-1/2) = 0$ and HMAX = 2^{-8}. The exact solution is

$$2^{20}y = \frac{1}{8}\tan^{-1}(2^{29}) + \frac{1}{8}\tan^{-1}\frac{x}{2^{-30}} .$$

Thus, $2^{20}y(1/2) = 0.39269908$. The results, for three values of ERROR, are shown in Table IV. The value of KSTEP at the end of the integration is also given.

TABLE IV

ERROR	KSTEP	$2^{20}y(1/2)$
2^{-32}	453	.39274749
2^{-36}	518	.39270319
2^{-40}	643	.39269939

Problem 5

$$\frac{dy}{dx} = 20\,\frac{y}{x}$$

This unstable equation was integrated on the IBM 7030 from $x = 1/2$ to $x = 1$, with $y(1/2) = 2^{-21}$ and HMAX = 2^{-4}. The exact solution is

$$y = \tfrac{1}{2}x^{20}.$$

Thus, $y(1) = 1/2$. The result, for ERROR = 2^{-25}, is shown in Table V. At the end of the integration, KSTEP was 126.

TABLE V

x	y
.50	4.76837×10^{-7}
.5078125	6.50520×10^{-7}
1.0	.500195

Problem 6

$$x^2 \frac{d^2y}{dx^2} + x \frac{dy}{dx} + [x^2 - (16)^2]y = 0$$

A solution of this equation is the Bessel function of order 16. The equation was integrated from x = 6 to x = 6138 with HMAX = 1 and initial conditions y(6) = 1.201950 X 10^{-6} and dy/dx = 2.986480 X 10^{-6}. The results, for ERROR = 2^{-23} and ERROR = 2^{-28}, are shown in Table VI and Table VII, respectively.

TABLE VI (ERROR = 2^{-23})

KSTEP	x	y(DEQ)	y(correct)	dy/dx(DEQ)	dy/dx(correct)
	6.0		1.201950 x 10^{-6}		2.986480 x 10^{-6}
29	6.125	1.633716 x 10^{-6}	1.633712 x 10^{-6}	3.963782 x 10^{-6}	3.963764 x 10^{-6}
59403	6138.0	1.360368 x 10^{-3}	1.362485 x 10^{-3}	1.008985 x 10^{-2}	1.009251 x 10^{-2}

TABLE VII (ERROR = 2^{-28})

KSTEP	x	y(DEQ)	y(correct)	dy/dx(DEQ)	dy/dx(correct)
	6.0		1.201950 x 10^{-6}		2.986480 x 10^{-6}
102721	6138.0	1.362435 x 10^{-3}	1.362485 x 10^{-3}	1.009249 x 10^{-2}	1.009251 x 10^{-2}

REFERENCES

1. A. Nordsieck, Math. Comp. 16, 22 (1962). [See also A. Nordsieck, Proc. Symp. Appl. Math., Amer. Math. Soc., Vol. XV, pp. 241-250 (1963)].

2. W. E. Milne, Numerical Solution of Differential Equations (John Wiley and Sons, Inc., New York, 1953), pp. 53-55.

3. L. Collatz, The Numerical Treatment of Differential Equations, 3rd Ed. (Springer-Verlag, Berlin, 1960), pp. 83-86.

4. Reference 1, Eq. (8) and pp. 29, 30, and 36.

5. Reference 1, Eqs. (8) and (20).

6. Reference 1, Eq. (22a) and p. 36.

7. J. H. Wilkinson, Rounding Errors in Algebraic Processes (Her Majesty's Stationery Office, London, 1963), pp. 79-82.

8. Reference 1, pp. 25, 35, and 46.

9. Reference 1, pp. 38-39 and Appendix C.

APPENDIX IA. FORTRAN-II LISTING OF DEQ FOR IBM 7094

```
      SUBROUTINE DEQ(FF,XX,XLIMIT,YY,ERROR,NEQ,HH,HMAX,JUMP,KKSTEP,
     1KCON,CCLIF)
     0DIMENSION STARTY(20),Y(20),SY(20),SAVEY(20),F(20),FP(20),
     1DELTA  (20),DALTA  (20),A(20),B(20),C(20),D(20),
     2AA(20),BB(20),CC(20),DD(20),SF(20),FF(20),YY(20)
      NE = NEQ
      H=HH
      X=XX
      KSTEP=KKSTEP
      DO 1100 I=1,NE
      F(I)=FF(I)
 1100 Y(I)=YY(I)
      CLIF=CCLIF
      IF(KSTEP-32767)993,991,991
  991 KSTEP=28
C     TEST FOR TYPE OF ENTRY
  993 IF(JUMP)1,998,999
  998 GO TO IA,(1000,11,21,802,803)
C     JUMP POS.   RESTORE VALUES
  999 X=SAVEX
  992 JUMP=0
      DO 901 I=1,NE
      F(I)=SF(I)
  901 Y(I)=SAVEY(I)
      GO TO 102
C     JUMP NEG.   INITIALIZE
    1 DO 5 I=1,NE
      STARTY(I)=Y(I)
      A(I)=0.001
      B(I)=0.001
      C(I)=0.001
    5 D(I)=0.001
      KSTEP=0
      KDELAY=0
      KCON=0
      T=95./(288.*64.)
      U=863./(12.*5040.)
      V=95./288.
      P=25./24.
      Q=35./72.
      R=5./48.
      S=1./120.
      ASSIGN 1000 TO IA
      JUMP=0
      GO TO 1101
C     BEGIN INTEGRATION STEP
 1000 DO 1111 I=1,NE
      SF(I)=F(I)
 1111 SAVEY(I)=Y(I)
C      H TOO SMALL, RETURN WITH JUMP NEG.
  600 IF(ABSF(X+H)-ABSF(X))605,601,605
  601 JUMP=-1
      GO TO 1101
  605 X=X+H
      DO 10 I=1,NE
      Y(I)=Y(I)+H*(F(I)+A(I)+B(I)+C(I)+D(I))
      Y(I)=RNDNF(Y(I))
   10 FP(I)=F(I)+2.0*A(I)+3.0*B(I)+4.0*C(I)+5.0*D(I)
```

```
      ASSIGN 11 TO IA
      GO TO 1101
   11 DO 12 I=1,NE
   12 SY(I)=Y(I)
      DO 20 I=1,NE
      DELTA  (I)=F(I)-FP(I)
   20 Y(I)=Y(I)+V*DELTA  (I)*H
      ASSIGN 21 TO IA
      KCON=1
      GO TO 1101
   21 KCON=0
      DO 30 I=1,NE
      DALTA  (I)=F(I)-FP(I)
      DALTA(I)=RNDNF(DALTA(I))
   29 Y(I)=SY(I)+V*DALTA  (I)*H
      Y(I)=RNDNF(Y(I))
   30 CONTINUE
C     TEST FOR STARTING SEQUENCE
   31 IF(KSTEP-28)35,40,40
C     APPLY TEST 2 ON ZEROTH STEP
   35 IF(KSTEP)50,50,60
C     HALVING TESTS
   40 DO 45 I=1,NE
      IF(ABSF(DALTA  (I))-ERROR/ABSF(H))45,45,55
   45 CONTINUE
   50 IF(V*H*CLIF-0.125)60,60,55
   55 X=X-H
C     FAIL TESTS, HALVE H
  223 H=H/2.0
      KDELAY=0
      DO 56 I=1,NE
      A(I)=A(I)/2.0
      B(I)=B(I)/4.0
      C(I)=C(I)/8.0
      F(I)=SF(I)
      Y(I)=SAVEY(I)
   56 D(I)=D(I)/16.0
      GO TO 1000
C     PASS TESTS, CORRECT A,B,C,D
   60 KSTEP=KSTEP+1
      DO 65 I=1,NE
      A(I)=A(I)+3.0*B(I)+6.0*C(I)+10.0*D(I)+P*DALTA  (I)
   62 B(I)=B(I)+4.0*C(I)+10.0*D(I)+Q*DALTA(I)
   64 C(I)=C(I)+5.0*D(I)+R*DALTA(I)
   67 D(I)=D(I)+S*DALTA(I)
   65 CONTINUE
C     IF IN STARTING SEQUENCE, BRANCH
      IF(KSTEP-24) 70,90,100
   70GO TO (1000,1000,1000,74,1000,1000,1000,78,1000,1000,1000,74,1000,
     11000,1000,86,1000,1000,1000,74,1000,1000,1000),KSTEP
C     4TH ,12TH,20TH STEP,GO BACK
   74 H=-H
      DO 75 I=1,NE
      A(I)=-A(I)
   75 C(I)=-C(I)
      GO TO 1000
C     8TH STEP, GO FORWARD
   78 H=-H
      DO 79 I=1,NE
      Y(I)=STARTY(I)
      A(I)=-A(I)
   79 C(I)=-C(I)
```

```
      GO TO 1COC
C     16TH STEP, HALVE H, APPLY TEST 1
   86 H=H/2.0
      DO 87 I=1,NE
      A(I)=A(I)/2.0
      B(I)=B(I)/4.0
      C(I)=C(I)/8.0
   87 D(I)=D(I)/16.C
      DC 88 I=1,NE
      IF(ABSF(DALTA  (I))-ERROR/ABSF(H))88,88,89
   88 CONTINUE
C     PASS TEST  GO FCRWARD WITH HALVED H
      GO TO 78
C     FAIL TEST  BEGIN AGAIN WITH HALVED H
   89 H=-H
      DO 92 I=1,NE
   92 Y(I)=STARTY(I)
      GO TO 1
C     24TH STEP, DOUBLE H, STARTING SEQUENCE ENDS
   90 H=H*2.0
      DO 91 I=1,NE
      A(I)=A(I)*2.0
      B(I)=B(I)*4.0
      C(I)=C(I)*8.0
   91 D(I)=D(I)*16.C
      GO TO 78
  100 KDELAY =KDELAY+1
C     WILL NEXT STEP MOVE PAST XLIMIT
  102 IF(ABSF(XLIMIT-X)-ABSF(H))103,1C3,110
C     YES.....SAVE X AND Y, INTEGRATE TO XLIMIT, RETURN.
  103 ENDH=XLIMIT-X
      DC 105 I=1,NE
      AA(I)=ENDH*A(I)/H
      BB(I)=ENDH**2*B(I)/H**2
      CC(I)=ENDH**3*C(I)/H**3
  1C5 DD(I)=ENDH**4*D(I)/H**4
      SAVEX=X
      DO 80C I=1,NE
      SF(I)=F(I)
  80C SAVEY(I)=Y(I)
  801 X=XLIMIT
      DO 106 I=1,NE
      Y(I)=Y(I)+ENDH*(F(I)+AA(I)+BB(I)+CC(I)+DD(I))
      Y(I)=RNDNF(Y(I))
  106 FP(I)=F(I)+2.C*AA(I)+3.C*BB(I)+4.0*CC(I)+5.0*DD(I)
      ASSIGN 802 TO IA
      GC TO 1101
  802 DC 805 I=1,NE
  805 SY(I)=Y(I)
      DO 107 I=1,NE
      DELTA  (I)=F(I)-FP(I)
  1C7 Y(I)=Y(I)+V*CELTA  (I)*ENDH
      ASSIGN 803 TO IA
      GO TO 11C1
  803 DO 108 I=1,NE
      DALTA  (I)=F(I)-FP(I)
  108 Y(I)=SY(I)+V*CALTA  (I)*ENDH
      Y(I)=RNDNF(Y(I))
      JUMP=1
      GO TO 1101
C     NO.....TEST FOR DOUBLING.  IF OK, BEGIN NEXT STEP AFTER DCUBLING
  110 IF(ABSF(XLIMIT-X)-ABSF(2.*H))10C0,1C00,111
```

```
 111 IF(KDELAY-4)1COC,120,12C
 120 IF(ABSF(2.*H)-ABSF(HMAX))121,121,1CCO
 121 DO 125I=1,NE
     IF(ABSF(DALTA  (I))-ERROR/(128.*ABSF(H)))125,125,1000
 125 CONTINUE
     IF(V*H*CLIF-0.0625)130,1000,100C
 130 CONTINUE
 335 H=2.0*H
     DO 135 I=1,NE
     A(I)=2.C*A(I)
     B(I)=4.0*B(I)
     C(I)=8.0*C(I)
 135 D(I)=16.0*C(I)
     KDELAY=C
     GO TO 1C00
11C1 NEQ=NE
     HH=H
     XX=X
     KKSTEP=KSTEP
     DO 1102 I=1,NE
     FF(I)=F(I)
1102 YY(I)=Y(I)
     CCLIF=CLIF
     RETURN
     END
```

APPENDIX IB. FAP LISTING OF RNDN FOR IBM 7094

```
*       FAP
*RNDN            NORMAL ROUND
        ENTRY    RNDN
 RNDN   FRN
        TRA      1,4
        END
```

APPENDIX IIA. FORTRAN-IV LISTING OF DEQ FOR IBM 7094

```
      SUBROUTINE DEQ(FF,XX,XLIMIT,YY,ERROR,NEQ,HH,HMAX,JUMP,KKSTEP,
     1KCON,CCLIF)
      DIMENSION STARTY(20),Y(20),SY(20),SAVEY(20),F(20),FP(20),
     1DELTA  (20),DALTA  (20),A(20),B(20),C(20),D(20),
     2AA(20),BB(20),CC(20),DD(20),SF(20),FF(20),YY(20)
      NE = NEQ
      H=HH
      X=XX
      KSTEP=KKSTEP
      DO 1100 I=1,NE
      F(I)=FF(I)
 1100 Y(I)=YY(I)
      CLIF=CCLIF
      IF(KSTEP-32767)993,991,991
  991 KSTEP=28
C     TEST FOR TYPE OF ENTRY
  993 IF(JUMP)1,998,999
  998 GO TO IA,(1000,11,21,802,803)
C     JUMP POS.   RESTORE VALUES
  999 X=SAVEX
  992 JUMP=0
      DO 901 I=1,NE
      F(I)=SF(I)
  901 Y(I)=SAVEY(I)
      GO TO 102
C     JUMP NEG.   INITIALIZE
    1 DO 5 I=1,NE
      STARTY(I)=Y(I)
      A(I)=0.001
      B(I)=0.001
      C(I)=0.001
    5 D(I)=0.001
      KSTEP=0
      KDELAY=0
      KCON=0
      T=95.0/(288.0*64.0)
      U=863.0/(12.0*5040.0)
      V=95.0/288.0
      P=25.0/24.0
      Q=35.0/72.0
      R=5.0/48.0
      S=1.0/120.0
      ASSIGN 1000 TO IA
      JUMP=0
      GO TO 1101
C     BEGIN INTEGRATION STEP
 1000 DO 1111 I=1,NE
      SF(I)=F(I)
 1111 SAVEY(I)=Y(I)
C      H TOO SMALL, RETURN WITH JUMP NEG.
  600 IF(ABS(X+H)-ABS(X))605,601,605
  601 JUMP=-1
      GO TO 1101
  605 X=X+H
      DO 10 I=1,NE
      Y(I)=Y(I)+H*(F(I)+A(I)+B(I)+C(I)+D(I))
      Y(I)=RNDN(Y(I))
   10 FP(I)=F(I)+2.0*A(I)+3.0*B(I)+4.0*C(I)+5.0*D(I)
```

```
      ASSIGN 11 TO IA
      GO TO 1101
   11 DO 12 I=1,NE
   12 SY(I)=Y(I)
      DO 20 I=1,NE
      DELTA  (I)=F(I)-FP(I)
   20 Y(I)=Y(I)+V*DELTA  (I)*H
      ASSIGN 21 TO IA
      KCON=1
      GO TO 1101
   21 KCON=0
      DO 30 I=1,NE
      DALTA  (I)=F(I)-FP(I)
      DALTA(I)=RNDN(DALTA(I))
   29 Y(I)=SY(I)+V*DALTA  (I)*H
      Y(I)=RNDN(Y(I))
   30 CONTINUE
C     TEST FOR STARTING SEQUENCE
   31 IF(KSTEP-28)35,40,40
C     APPLY TEST 2 ON ZEROTH STEP
   35 IF(KSTEP)50,50,60
C     HALVING TESTS
   40 DO 45 I=1,NE
      IF(ABS(DALTA (I))-ERROR/ABS(H))45,45,55
   45 CONTINUE
   50 IF(V*H*CLIF-0.125)60,60,55
   55 X=X-H
C     FAIL TESTS, HALVE H
  223 H=H/2.0
      KDELAY=0
      DO 56 I=1,NE
      A(I)=A(I)/2.0
      B(I)=B(I)/4.0
      C(I)=C(I)/8.0
      F(I)=SF(I)
      Y(I)=SAVEY(I)
   56 D(I)=D(I)/16.0
      GO TO 1000
C     PASS TESTS, CORRECT A,B,C,D
   60 KSTEP=KSTEP+1
      DO 65 I=1,NE
      A(I)=A(I)+3.0*B(I)+6.0*C(I)+10.C*D(I)+P*DALTA  (I)
   62 B(I)=B(I)+4.0*C(I)+10.0*D(I)+Q*DALTA(I)
   64 C(I)=C(I)+5.0*D(I)+R*DALTA(I)
   67 D(I)=D(I)+S*DALTA(I)
   65 CONTINUE
C     IF IN STARTING SEQUENCE, BRANCH
      IF(KSTEP-24) 70,90,100
   700GO TO (1000,1000,1000,74,1000,1000,1000,78,1000,1000,1000,74,1000,
     11000,1000,86,1000,1000,1000,74,1000,1000,1000),KSTEP
C     4TH ,12TH,20TH STEP,GO BACK
   74 H=-H
      DO 75 I=1,NE
      A(I)=-A(I)
   75 C(I)=-C(I)
      GO TO 1000
C     8TH STEP, GO FORWARD
   78 H=-H
      DO 79 I=1,NE
      Y(I)=STARTY(I)
      A(I)=-A(I)
   79 C(I)=-C(I)
```

```
      GC TO 1000
C     16TH STEP, HALVE H, APPLY TEST 1
   86 H=H/2.0
      DO 87 I=1,NE
      A(I)=A(I)/2.0
      B(I)=B(I)/4.0
      C(I)=C(I)/8.0
   87 D(I)=D(I)/16.C
      DO 88 I=1,NE
      IF(ABS(DALTA (I))-ERRCR/ABS(H))88,88,89
   88 CONTINUE
C     PASS TEST  GO FORWARD WITH HALVED H
      GO TO 78
C     FAIL TEST  BEGIN AGAIN WITH HALVED H
   89 H=-H
      DC 92 I=1,NE
   92 Y(I)=STARTY(I)
      GO TO 1
C     24TH STEP, DOUBLE H, STARTING SEQUENCE ENDS
   90 H=H*2.0
      DO 91 I=1,NE
      A(I)=A(I)*2.0
      B(I)=B(I)*4.C
      C(I)=C(I)*8.0
   91 D(I)=D(I)*16.0
      GO TO 78
  100 KDELAY =KDELAY+1
C     WILL NEXT STEP MOVE PAST XLIMIT
  102 IF(ABS(XLIMIT-X)-ABS(H))103,103,110
C     YES.....SAVE X AND Y, INTEGRATE TO XLIMIT, RETURN.
  103 ENDH=XLIMIT-X
      DO 105 I=1,NE
      AA(I)=ENDH*A(I)/H
      BB(I)=ENDH**2*B(I)/H**2
      CC(I)=ENDH**3*C(I)/H**3
  105 DD(I)=ENDH**4*D(I)/H**4
      SAVEX=X
      DO 800 I=1,NE
      SF(I)=F(I)
  800 SAVEY(I)=Y(I)
  801 X=XLIMIT
      DO 106 I=1,NE
      Y(I)=Y(I)+ENDH*(F(I)+AA(I)+BB(I)+CC(I)+DD(I))
      Y(I)=RNDN(Y(I))
  106 FP(I)=F(I)+2.0*AA(I)+3.C*BB(I)+4.0*CC(I)+5.0*DD(I)
      ASSIGN 802 TO IA
      GO TO 1101
  802 DO 805 I=1,NE
  805 SY(I)=Y(I)
      DO 107 I=1,NE
      DELTA  (I)=F(I)-FP(I)
  107 Y(I)=Y(I)+V*DELTA  (I)*ENDH
      ASSIGN 803 TO IA
      GO TO 1101
  803 DO 108 I=1,NE
      DALTA  (I)=F(I)-FP(I)
  108 Y(I)=SY(I)+V*DALTA  (I)*ENDH
      Y(I)=RNDN(Y(I))
      JUMP=1
      GO TO 1101
C     NO.....TEST FOR DOUBLING.  IF OK, BEGIN NEXT STEP AFTER DOUBLING
  110 IF(ABS(XLIMIT-X)-ABS(2.0*H))100C,1CC0,111
```

```
 111 IF(KDELAY-4)1COC,120,120
 120 IF(ABS(2.0*H)-ABS(HMAX))121,121,10CC
 121 DO 125I=1,NE
     IF(ABS(CALTA (I))-ERRCR/(128.0*ABS(H)))125,125,1C00
 125 CONTINUE
     IF(V*H*CLIF-0.0625)13C,1000,100C
 130 CONTINUE
 335 H=2.0*H
     DO 135 I=1,NE
     A(I)=2.0*A(I)
     B(I)=4.0*B(I)
     C(I)=8.0*C(I)
 135 D(I)=16.0*D(I)
     KDELAY=0
     GO TO 1000
1101 NEC=NE
     HH=H
     XX=X
     KKSTEP=KSTEP
     DO 1102 I=1,NE
     FF(I)=F(I)
1102 YY(I)=Y(I)
     CCLIF=CLIF
     RETURN
     END
```

APPENDIX IIB. MAP LISTING OF RNDN FOR IBM 7094

```
$IBMAP RNDN
       ENTRY   RNDNF
 RNDNF CLA*    3,4
       FRN
       TRA     1,4
       END
```

APPENDIX III. FORTRAN-IV LISTING OF DEQ FOR IBM 7030

```
      SUBROUTINE DEQ(FF,XX,XLIMIT,YY,ERROR,NEQ,HH,HMAX,JUMP,KKSTEP,
     1KCON,CCLIF)
      DIMENSION STARTY(20),Y(20),SY(20),SAVEY(20),F(20),FP(20),
     1DELTA  (20),CALTA  (20),A(20),B(20),C(20),D(20),
     2AA(20),BB(20),CC(20),CD(20),SF(20),FF(20),YY(20)
      NE = NEQ
      H=HH
      X=XX
      KSTEP=KKSTEP
      DO 1100 I=1,NE
      F(I)=FF(I)
 1100 Y(I)=YY(I)
      CLIF=CCLIF
      IF(KSTEP-99999999999 )993,991,991
  991 KSTEP=28
C     TEST FOR TYPE OF ENTRY
  993 IF(JUMP)1,998,999
  998 GO TO IA,(1000,11,21,802,803)
C     JUMP POS.   RESTORE VALUES
  999 X=SAVEX
  992 JUMP=0
      DO 901 I=1,NE
      F(I)=SF(I)
  901 Y(I)=SAVEY(I)
      GO TO 102
C     JUMP NEG.   INITIALIZE
    1 DO 5 I=1,NE
      STARTY(I)=Y(I)
      A(I)=0.001
      B(I)=0.001
      C(I)=0.001
    5 D(I)=0.001
      KSTEP=0
      KDELAY=0
      KCON=0
      T=95./(288.*64.)
      U=863./(12.*5040.)
      V=95.0/288.0
      P=25.0/24.0
      Q=35.0/72.0
      R=5.0/48.0
      S=1.0/120.0
      ASSIGN 1000 TO IA
      JUMP=0
      GO TO 1101
C     BEGIN INTEGRATION STEP
 1000 DO 1111 I=1,NE
      SF(I)=F(I)
 1111 SAVEY(I)=Y(I)
C      H TOO SMALL, RETURN WITH JUMP NEG.
  600 IF(ABS(X+H)-ABS(X))605,601,605
  601 JUMP=-1
      GO TO 1101
  605 X=X+H
      DO 10 I=1,NE
      Y(I)=Y(I)+H*(F(I)+A(I)+B(I)+C(I)+D(I))
   10 FP(I)=F(I)+2.0*A(I)+3.0*B(I)+4.0*C(I)+5.0*D(I)
      ASSIGN 11 TO IA
      GO TO 1101
   11 DO 12 I=1,NE
   12 SY(I)=Y(I)
      DO 20 I=1,NE
      DELTA  (I)=F(I)-FP(I)
   20 Y(I)=Y(I)+V*DELTA  (I)*H
      ASSIGN 21 TO IA
      KCON=1
      GO TO 1101
   21 KCON=0
      DO 30 I=1,NE
      DALTA  (I)=F(I)-FP(I)
   29 Y(I)=SY(I)+V*DALTA  (I)*H
   30 CONTINUE
C     TEST FOR STARTING SEQUENCE
   31 IF(KSTEP-28)35,40,40
C     APPLY TEST 2 ON ZEROTH STEP
   35 IF(KSTEP)50,50,60
C     HALVING TESTS
   40 DO 45 I=1,NE
      IF(ABS(DALTA (I))-ERROR/ABS(H))45,45,55
   45 CONTINUE
   50 IF(V*H*CLIF-0.125)60,60,55
   55 X=X-H
C     FAIL TESTS, HALVE H
  223 H=H/2.0
      KDELAY=0
      DO 56 I=1,NE
      A(I)=A(I)/2.0
      B(I)=B(I)/4.0
      C(I)=C(I)/8.0
      F(I)=SF(I)
      Y(I)=SAVEY(I)
   56 D(I)=D(I)/16.0
      GO TO 1000
C     PASS TESTS, CORRECT A,B,C,D
   60 KSTEP=KSTEP+1
      DO 65 I=1,NE
      A(I)=A(I)+3.0*B(I)+6.0*C(I)+10.0*D(I)+P*DALTA  (I)
   62 B(I)=B(I)+4.0*C(I)+10.0*D(I)+Q*DALTA(I)
   64 C(I)=C(I)+5.0*D(I)+R*DALTA(I)
   67 D(I)=D(I)+S*DALTA(I)
   65 CONTINUE
C     IF IN STARTING SEQUENCE, BRANCH
      IF(KSTEP-24) 70,90,100
   70GO TO (1000,1000,1000,74,1000,1000,1000,78,1000,1000,1000,74,1000,
     11000,1000,86,1000,1000,1000,74,1000,1000,1000,1000),KSTEP
C     4TH ,12TH,20TH STEP,GO BACK
   74 H=-H
      DO 75 I=1,NE
      A(I)=-A(I)
   75 C(I)=-C(I)
      GO TO 1000
C     8TH STEP, GO FORWARD
   78 H=-H
      DO 79 I=1,NE
      Y(I)=STARTY(I)
      A(I)=-A(I)
   79 C(I)=-C(I)
      GO TO 1000
C     16TH STEP, HALVE H, APPLY TEST 1
   86 H=H/2.0
```

```
      DO 87 I=1,NE
      A(I)=A(I)/2.0
      B(I)=B(I)/4.0
      C(I)=C(I)/8.0
   87 D(I)=D(I)/16.C
      DO 88 I=1,NE
      IF(ABS(DALTA (I))-ERROR/ABS(H))88,88,89
   88 CONTINUE
C     PASS TEST  GO FORWARD WITH HALVED H
      GO TO 78
C     FAIL TEST  BEGIN AGAIN WITH HALVED H
   89 H=-H
      DO 92 I=1,NE
   92 Y(I)=STARTY(I)
      GO TO 1
C     24TH STEP, DOUBLE H, STARTING SEQUENCE ENDS
   90 H=H*2.0
      DO 91 I=1,NE
      A(I)=A(I)*2.0
      B(I)=B(I)*4.0
      C(I)=C(I)*8.0
   91 D(I)=D(I)*16.C
      GO TO 78
  100 KDELAY =KDELAY+1
C     WILL NEXT STEP MOVE PAST XLIMIT
  102 IF(ABS(XLIMIT-X)-ABS(H))103,103,110
C     YES.....SAVE X AND Y, INTEGRATE TO XLIMIT, RETURN.
  103 ENDH=XLIMIT-X
      DO 105 I=1,NE
      AA(I)=ENDH*A(I)/H
      BB(I)=ENDH**2*B(I)/H**2
      CC(I)=ENDH**3*C(I)/H**3
  105 DD(I)=ENDH**4*D(I)/H**4
      SAVEX=X
      DO 800 I=1,NE
      SF(I)=F(I)
  800 SAVEY(I)=Y(I)
  801 X=XLIMIT
      DO 106 I=1,NE
      Y(I)=Y(I)+ENDH*(F(I)+AA(I)+BB(I)+CC(I)+DD(I))
  106 FP(I)=F(I)+2.0*AA(I)+3.0*BB(I)+4.0*CC(I)+5.0*DD(I)
      ASSIGN 802 TO IA
      GO TO 1101
  802 DO 805 I=1,NE
  805 SY(I)=Y(I)
      DO 107 I=1,NE
      DELTA  (I)=F(I)-FP(I)
  107 Y(I)=Y(I)+V*DELTA  (I)*ENDH
      ASSIGN 803 TO IA
      GO TO 1101
  803 DO 108 I=1,NE
      DALTA  (I)=F(I)-FP(I)
  108 Y(I)=SY(I)+V*DALTA  (I)*ENDH
      JUMP=1
      GO TO 1101
C     NO.....TEST FOR DOUBLING.  IF OK, BEGIN NEXT STEP AFTER DOUBLING
  110 IF(ABS(XLIMIT-X)-ABS(2.0*H))1000,1000,111
  111 IF(KDELAY-4)1C00,120,12C
  120 IF(ABS(2.0*H)-ABS(HMAX))121,121,10C0
  121 DO 125I=1,NE
      IF(ABS(DALTA (I))-ERROR/(128.0*ABS(H)))125,125,1C00
  125 CONTINUE
```

```
      IF(V*H*CLIF-0.0625)130,1000,1000
  130 CONTINUE
  335 H=2.0*H
      DO 135 I=1,NE
      A(I)=2.0*A(I)
      B(I)=4.0*B(I)
      C(I)=8.0*C(I)
  135 D(I)=16.0*C(I)
      KDELAY=0
      GO TO 1000
 1101 NEQ=NE
      HH=H
      XX=X
      KKSTEP=KSTEP
      DO 1102 I=1,NE
      FF(I)=F(I)
 1102 YY(I)=Y(I)
      CCLIF=CLIF
      RETURN
      END
```

APPENDIX IV. FORTRAN-IV LISTING OF SAMPLE PROGRAM

```
C     SAMPLE FORTRAN-IV PROGRAM FOR COMPUTING
C     LEGENDRE POLYNOMIALS.  Y(2) = LEGENDRE
C     POLYNOMIAL OF ORDER V.  Y(1) = (1.-X**2) TIMES
C     DERIVATIVE OF LEGENDRE POLYNOMIAL.
      DIMENSION Y(2),F(2)
C     INPUT TAPE = 10.  OUTPUT TAPE = 9.
   51 READ (10,50)X0,V,ERROR,Y1,Y2
   50 FORMAT(1P5E12.7)
      X=X0
      Y(1)=Y1
      Y(2)=Y2
      WRITE (9,11)ERROR,V
   11 FORMAT(7H-ERROR=,E10.4,26H                         V=F6.0)
      WRITE (9,1)X,Y(1),Y(2)
    1 FORMAT(3E16.8)
      HMAX=0.125
      H=HMAX
      JUMP=-1
    6 XLIMIT=X+0.1
    5 CALL DEQ(F,X,XLIMIT,Y,ERROR,2,H,HMAX,JUMP,KSTEP,KCON,CLIF)
      IF(JUMP)2,3,4
    2 WRITE (9,30)
   30 FORMAT(8H JUMP=-1)
      CALL EXIT
    3 F(1)=-V*(V+1.)*Y(2)
      F(2)=Y(1)/(1.-X**2)
      IF(KCON)5,5,21
   21 CLIF=SQRT(ABS(V*(V+1.)/(1.-X**2)))
      GO TO 5
    4 WRITE (9,1)X,Y(1),Y(2)
      IF(.95-X)51,51,6
      END
```