FINAL REPORT

PROJECT A-1167

RESEARCH IN PRECISION INTEGRATION METHODS

L. J. GALLAHER AND I. E. PERLIN

GRANT NUMBER NGR 11-002-101

1 APRIL 1969 to 31 MARCH 1970

Supported by the
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
WASHINGTON, D. C.

1970

Engineering Experiment Station
GEORGIA INSTITUTE OF TECHNOLOGY
Atlanta, Georgia

GEORGIA INSTITUTE OF TECHNOLOGY
Rich Electronic Computer Center
Atlanta, Georgia

FINAL REPORT

PROJECT A-1167

RESEARCH IN PRECISION INTEGRATION METHODS

By

L. J. GALLAHER and I. E. PERLIN

GRANT NUMBER NGR 11-002-101

1 APRIL 1969 to 31 MARCH 1970

Supported by the
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
WASHINGTON, D. C.

ABSTRACT

This report contains the results of comparisons and computer tests of several methods for numerically integrating systems of coupled non-linear differential equations (initial value problems). The methods tested are:

1) Shanks' formulas for the method of Runge and Kutta.

2) The Adams, Bashforth, and Moulton method.

3) Butcher's formulas for the method of Stetter and Gragg.

4) Cowell's method.

Each of these methods was programmed as a general purpose subroutine in double precision FORTRAN V for the UNIVAC 1108. The test problem on which results are given here is that of a particle in an inverse square attractive force field with an elliptic orbit of eccentricity 0.8. Single orbits were run for orders from 7 through 13 and accuracies in the range 10^{-7} to 10^{-12}. Plots are given of error versus time and versus number of function evaluations for the various methods and orders.

The results show that all of the methods perform well but the higher orders of each method are significantly faster at the higher accuracies. At corresponding orders the Butcher formulas appear to be superior.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

I. INTRODUCTION

This report gives the results of tests and comparisons for a collection of methods for numerically integrating systems of coupled non-linear differential equations.

The methods tested are :

1) Shanks' formulas for the method of Runge and Kutta.

2) The Adams, Bashforth and Moulton method.

3) Butcher's formulas for the method of Stetter and Gragg.

4) Cowell's method.

For each of these methods, general purpose subroutines were written that would handle up to 99 equations. Each method has automatic error and step size control. The multi-step methods use the Shanks subroutine for starting and for restarting to reduce step size.

Each of the methods is programmed in double precision FORTRAN V for the UNIVAC 1108.

Several test problems were used to exercise the methods. The results reported here are for an orbit of a particle in an inverse square attractive force field, with an elliptic orbit of eccentricity 0.8. Tests were run for various orders from 7 through 13 on the different methods, and plots are given of error versus the time and versus the number of function evaluations.

The results show that, while all of the methods perform well, the higher order methods are significantly faster at the higher accuracies. At corresponding orders and accuracies the Butcher formulas appear to be faster. There is as suggestion that if higher order Shanks formulas were available these would be effective at the higher accuracies.

Chapter II contains separate descriptions of each of the four integration methods together with details on the error and step size control, subroutine parameters and flow diagrams. The programs themselves are listed in Appendix A.

The coefficients for a wide range of orders for each method are listed in Appendix B.

II. INTEGRATION METHODS

A. Introduction

The four integration methods described in this chapter are as follows:

1) Shanks' formulas for the Runge-Kutta method.
2) The Adams, Bashforth and Moulton method.
3) Butcher's formulas for the Stetter-Gragg method.
4) Cowell's method.

Each of these methods is programmed in double precision (~18 decimal significant figures) in the FORTRAN V language for the UNIVAC 1108. They are written as general purpose subroutines that can handle up to 99 simultaneous ordinary differential equations. The methods are of variable order; that is, the order is specified by the user. The coefficients of the method and order to be used are supplied by the user and passed to each subroutine at the time called. Sets of these coefficients are listed in Appendix B. In practice these coefficients are read in from a machine language tape (in double precision) by the calling program.

Each of the subroutines references the external subroutine FUNCTI to obtain the derivatives for the equations being integrated. FUNCTI is of course written by the user to describe the particular problem.

A point to note is that the information about the dependent variable Y and its derivatives that is passed to and from FUNCTI, and to and from each of the subroutines, is contained in array positions 2 through N + 1 where N is the number of equations being integrated. The first positions in the Y vector and its derivative vector are not used by these programs.

The step size control in each method uses a parameter to indicate the expected manner that the errors accumulate and propagate. Set at 1.0 this parameter indicates that the errors are expected to be additive and all of the same sign. Setting this parameter to 0.5 indicates that the errors are expected to be random. With this parameter the subroutine tries at each step to project ahead and estimate the final error at the end of the integration and adjust the step size and error at each step accordingly.

In order to conserve storage, all the multi-step methods use some common storage allocated by the main program with the statement,

```
COMMON/COMMON/FA(35, NPI)
```

where NPI is a parameter. NPI is set to one more than the number of equations to be integrated but not greater than 100. The starting subroutine does not reference this common area.

The multistep methods all use a special Runge-Kutta-Shanks starting subroutine for getting started and for reducing step size. They also use the regular Runge-Kutta-Shanks subroutine for ending if the final step is smaller than the current step at ending. Thus,while the Runge-Kutta-Shanks subroutine could be used by itself,the others would need both the starting subroutine and the regular Runge-Kutta-Shanks in order to run.

The subroutines themselves are listed in Appendix A, together with a skeleton main program and FUNCTI subroutine.

B. The Runge-Kutta-Shanks Method

1. Introduction

The procedure described is a generalization of the Runge-Kutta method for solving a system of differential equations. It may be applied to an arbitrary system of first-order differential equations of the form

$$\vec{y}\,' = f\ (x,\vec{y})$$

with the initial conditions

$$\vec{y}\ (x_o) = \vec{y}_o$$

where $\vec{y}\ (x) = \begin{pmatrix} y_1\ (x) \\ \vdots \\ y_n\ (x) \end{pmatrix}$, $\quad \vec{y}\,'(x) = \begin{pmatrix} y_1'(x) \\ \vdots \\ y_n'(x) \end{pmatrix}$,

$$\vec{f}\ (x,\vec{y}) = \begin{pmatrix} f_1\ (x,y_1, \ldots, y_n) \\ \vdots \\ f_n\ (x,y_1, \ldots, y_n) \end{pmatrix}, \qquad \vec{y}_o = \begin{pmatrix} y_{10} \\ \vdots \\ y_{n0} \end{pmatrix} .$$

2. Description of the Method

The Shanks Method is a single-step procedure for finding a numerical solution of a first-order ordinary differential equation or system of differential equations in which the derivatives of the dependent variables may be expressed explicitly as functions of the independent and dependent variables.

Consider the system of differential equation

$$\vec{y}\,' = \vec{f}\ (x,\vec{y}) \quad .$$

Suppose the value of $\vec{y}$ (x) is known. The value $\vec{y}$ (x + h) is approximated by

$$\vec{y}(x+h) = \vec{y}(x) + h \sum_{i=1}^{m} \gamma_i \vec{f}_i (x,h,\vec{y}),$$

where

$$\vec{f}_1 (x,h,\vec{y}) = \vec{f}(x,\vec{y}),$$

$$\vec{f}_i (x,h,\vec{y}) = \vec{f}\,(x + \alpha_i h, \vec{y} + h \sum_{j=1}^{i-1} \beta_{ij}\vec{f}_j), \; i = 2, \ldots, m.$$

The coefficients α_i (i = 2, . . ., m),

β_{ij} (i = 2, . . ., m; j = 1, . . ., i-1), and γ_i(i = 1 , . . ., m)

are chosen so as to make the approximation correct to some order. A special case of the Shanks formula is the fourth-order Runge-Kutta formula:

$$\alpha_2 = 1/2,\ \alpha_3 = 1/2,\ \alpha_4 = 1,$$

$$\beta_{21} = 1/2,\ \beta_{31} = 0,\ \beta_{32} = 1/2,\ \beta_{41} = \beta_{42} = 0,\ \beta_{43} = 1,$$

$$\gamma_1 = 1/6,\ \gamma_2 = 1/3,\ \gamma_3 = 1/3,\ \gamma_4 = 1/6$$

For useful values of the various combinations of α, β, and γ, see Shanks [18].

3. The Computer Program

The subroutine was programmed for the UNIVAC 1108 computer in the FORTRAN V language. Double precision arithmetic (18 decimal digits) was used throughout except where integers are indicated.

3.1 Error Estimates and Step Size Control

In this subroutine a single set of Shanks formulas is used. Suppose a vector $\vec{y}$ (x) is known. Then the Shanks method is applied to one step of size h (where $h = \frac{\Delta x}{c}$, Δx is the length of the interval, and c is a power of two), and to two steps of size $\frac{h}{2}$, as follows:

$$\vec{y}_p = \vec{y}\,(x) + h \sum_{i=1}^{m} \gamma_i \vec{f}_i\,(x,h,\vec{y}),$$

$$\vec{y}_m = \vec{y}\,(x) + \frac{h}{2} \sum_{i=1}^{m} \gamma_i \vec{f}_i\,(x,\frac{h}{2},\vec{y})$$

$$\vec{y}_c = \vec{y}_m + \frac{h}{2} \sum_{i=1}^{m} \gamma_i \vec{f}_i\,(x + \frac{h}{2},\frac{h}{2},\vec{y}_m).$$

Both $\vec{y}_p$ and $\vec{y}_c$ are estimates of $\vec{y}(x + h)$. An error estimate $E_k = \frac{|y_{ck} - y_{pk}|}{f}$ (where f is an empirical factor) is calculated for each independent variable y_k. If both $E_k > \frac{E_{ak}}{c^p}$ and $E_k > \frac{E_{rk}|y_{ck}|}{c^p}$ for any dependent variable where E_{ak} is an absolute error estimate, E_{rk} is a relative error estimate, and p is an input parameter, usually 1 or 1/2, then the step is rejected and the step size is halved; otherwise the step is accepted and y_c is taken as the vector $\vec{y}(x + h)$. If for every dependent variable, either $E_k > \frac{E_{ak}}{2^{(j+3)}c^p}$ or $E_k > \frac{E_{rk}|y_{ck}|}{2^{(j+3)}c^p}$, where j is the order, then the step size is doubled. If the step size h is larger than the distance to the end of the interval, then that distance is taken as the step size.

3.2 Input and Output of the Subroutine

The subroutine is called as follows:

```
CALL SHANKS (N,XIV,XF,YV,IM,ORDER,CF,P,EA,ER,DXV)
```

where the parameters have the following meaning:

N - number of dependent variables (integer);

XIV - initial value of the independent variable;

XF - final value of the independent variable;

YV - array of initial values of the dependent variables;

IM - the number of function evaluations in each application of the Shanks method (integer);

ORDER - the order of the Shanks formulas used (integer);

CF - the array of Shanks coefficients, starting in the first element arranged as follows: for each i, the corresponding $\alpha_i \beta_{ij}$'s, followed by α_i, with the γ_i's at the end;

P - an exponent (usually 1/2 or 1) used in step size control (1 assuming the errors are additive; 1/2 assuming that they are random);

EA - an array of absolute error asked;

ER - an array of relative error asked;

DXV - a recommended starting step size (the actual starting step size will be $\frac{XF - XIV}{c}$, where c is the smallest power of 2 for which $\left|\frac{XF - XIV}{c}\right| \leq |DX|$).

The final values of the dependent variable are stored in YV before exiting the procedure.

All variables and arrays not designated integer are double precision.

3.3 Flow Diagram and Program Listing.

Figure 1 is the flow diagram for the Runge-Kutta-Shanks procedure. A listing of the program is given in Appendix A.

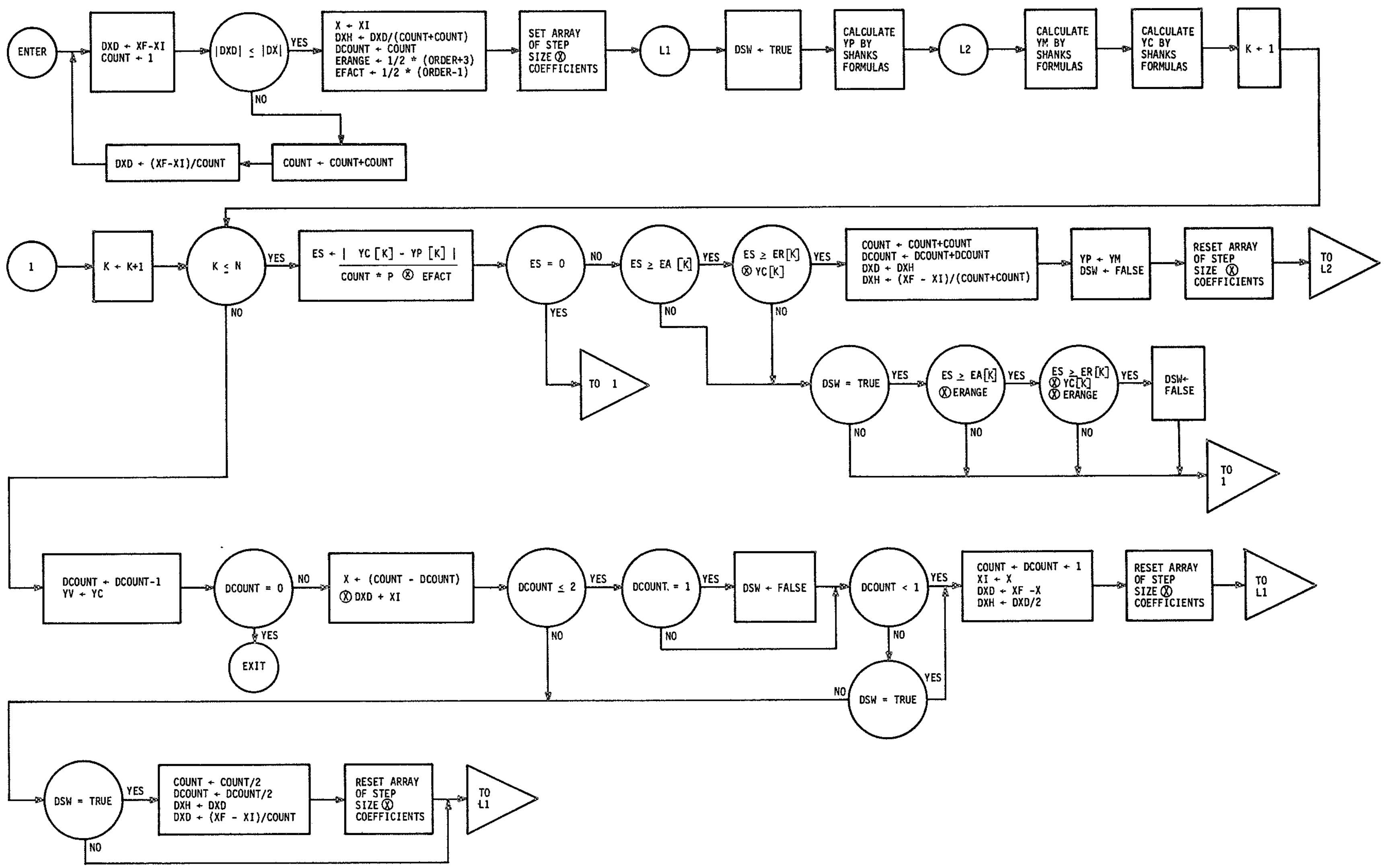

Figure 1. Flow Diagram for the Runge-Kutta-Shanks Procedure.

C. The Methods of Adams, Bashforth and Moulton

1. Description of the Method

The method investigated consists of the combination of two different versions of the method of Adams into a predictor-corrector system [5]. The use of this system to obtain numerical solutions to a set of simultaneous differential equations with given initial conditions is independent both of the number of equations in the set to be solved and of the orders of the individual equations in the set; provided, however, that each equation of order m is expressed as a set of m coupled first order equations.

In general then, one deals with the system of equations

$$\vec{y}'(x) \equiv \frac{d}{dx}\vec{y}(x) = \vec{f}\left(x,\vec{y}(x)\right), \qquad (1\text{-}1)$$

where $\vec{y}'$, $\vec{y}$, and $\vec{f}$ are vectors, each having a number of components, N, equal to $\sum_{i=1}^{k} m_i$, where k is the number of equations in the set to be solved, and the m_i are their individual orders.

This vector differential equation is equivalent to the integral equation

$$\vec{y}(x+h) = \vec{y}(x) + \int_{x}^{x+h} \vec{f}\left(t,\vec{y}(t)\right) dt. \qquad (1\text{-}2)$$

At the point $x = x_q \equiv x_{q-1} + h$, this integral is approximated first by

$$\vec{y}_q^{(0)} = \vec{y}_{q-1} + h \sum_{\mu=0}^{q-1} \beta_{q-1,q-1-\mu}\vec{f}_\mu \qquad (1\text{-}3a)$$

and then repeatedly by

$$\vec{y}_q^{(\nu+1)} = h\beta^*_{q,o}\vec{f}\left(x_q, \vec{y}^{(\nu)}(x_q)\right) + h\sum_{\mu=0}^{q-1} \beta^*_{q,q-\mu}\vec{f}_\mu + \vec{y}_{q-1}$$

$$\equiv h\beta^*_{q,o}\vec{f}^{(\nu)} + \vec{C}, \quad \nu = 0,1,2, \ldots \tag{1-3b}$$

which converges toward $\vec{y}_q \equiv \vec{y}(x_q)$ as ν increases. Formula (1-3a) is called the Adams-Bashforth predictor equation, and formula (1-3b) is the Adams-Moulton corrector.

The coefficients $\beta_{q\rho}$ and $\beta^*_{q\rho}$ are derived by the equivalent of integrating Lagrangian polynomials fitted to $\vec{f}$, but are independent of both $\vec{f}$ and h. The polynomial for the predictor is of degree q-1 passing through the q points $\vec{f}_0, \vec{f}_., \ldots, \vec{f}_{q-1}$, while that for the corrector is of degree q passing through the q + 1 points $\vec{f}_0, \vec{f}_1, \ldots, \vec{f}_q$.

An explicit formula for the $\beta_{q\rho}$ is

$$\beta_{q\rho} = (-1)^\rho\left\{\binom{\rho}{\rho}\gamma_\rho + \binom{\rho+1}{\rho}\gamma_{\rho+1} + \ldots + \binom{q}{\rho}\gamma_q\right\}, \quad q = 0,1,2, \ldots, \ \rho = 0,1, \ldots, q$$

where the $\binom{\rho+i}{\rho}$ represent binomial coefficients and the γ_ρ are found by the recursion relation

$$\gamma_\rho + \tfrac{1}{2}\gamma_{\rho-1} + \ldots + \frac{1}{\rho+1}\gamma_o = 1, \quad \rho = 0,1,2, \ldots,$$

and an explicit formula for the $\beta^*_{q\rho}$ is

$$\beta^*_{q\rho} = (-1)^\rho\left\{\binom{\rho}{\rho}\gamma^*_\rho + \binom{\rho+1}{\rho} + \ldots + \binom{q}{\rho}\gamma^*_q\right\}, \quad q = 0,1,2, \ldots, \ \rho = 0,1, \ldots, q$$

where $\gamma^*_o = 1$ and $\nu^*_\rho = \gamma_\rho - \gamma_{\rho-1}$, $\rho = 1,2,3, \ldots$.

Bounds on the errors for the two approximations are the maximums within the interval $[x_o, x_q]$ of

$$\left| \gamma_q h^{q+1} \frac{d^{q+1}}{dx^{q+1}} \vec{y} \right| \qquad \text{(for Adams-Bashforth)} \qquad (1\text{-}4a)$$

and of

$$\left| \gamma^*_{q+1} h^{q+2} \frac{d^{q+2}}{dx^{q+2}} \vec{y} \right| \qquad \text{(for Adams-Moulton)} \qquad (1\text{-}4b)$$

and M, the order of the predictor-corrector system, is assumed to approximate that of the corrector, which is q + 1.

2. The Computer Program

The subroutine ADAMS itself is written to be included in programs written in double precision for the UNIVAC 1108 computer. The language is FORTRAN V. There are no unusual hardware requirements, because all input and output to the procedure is under control of the including program through the formal parameter list.

2.1 Parameters

The following lists of formal parameters will be useful in describing the operation of procedure ADAMS. In the remainder of this discussion the interchange of upper and lower case letters, necessitated by approximating the notation [5] within the limited character set available to a computer, is straight-forward and will be done freely without further comment. Except for those variables designated integer, all variables are double precision. All arrays are double precision.

Formal Parameters

Identifier	Type	Usage or Meaning
N	Integer	The number of equations to be integrated.
XI	Double	Initial value of the independent variable.
XF	Double	Final value of the independent variable.
Y	Array	Current dependent variable vector. Contains initial values at entry and final values at exit.
P	Double	Power of C1 used in error control.
Q	Integer	Number of back $\vec{f}$ points used in the approximating polynomials. One less than M, the order of the method.
DXV	Double	Upper bound on the initial step size.
EA	Array	Absolute error bound vector.
ER	Array	Relative error bound vector.
ADMSCF	Array	Contains the $\beta_{q\rho}$, the $\beta^*_{q\rho}$, and $1 - \gamma_{q+1}/\gamma^*_{q+1}$.
RKSFNS	Integer	Function evaluations per step for procedures START and SHANKS.
RXSRDR	Integer	Order of R.K.S. method to be used by START and SHANKS.
RKSCFF	Array	Coefficients for START and SHANKS. See the descriptions of START and SHANKS elsewhere in this report for details.

2.2 The FUNCTI Subroutine

A subroutine for calculating the vector $\vec{y}' = \vec{f}(x,\vec{y}(x))$ must be an external subroutine characterizing the set of differential equations to be solved by a program using ADAMS. This subroutine is called by ADAMS as FUNCTI and must itself have the following formal parameter list:

Identifier	Type	Usage or Meaning
N	Integer	Number of equations.
X	Double	Current value of the independent variable.
YV	Array	Current dependent variable vector (input).
FV	Array	f value vector (output).

2.3 Orders Available

The subroutine ADAMS is written to be completely general with regard to order, and any order may be used if the necessary coefficients are placed in the ADMSCF array. For a given order $M = q + 1$, there are $2q + 2 = 2M$ coefficients which should appear in the array beginning at position one in the following order:

$$\beta_{q-1,q-1}, \beta_{q-1,q-2}, \ldots, \beta_{q-1,o}, \beta^*_{q,q}, \beta^*_{q,q-1}, \ldots, \beta^*_{q,o}, \left| 1-\gamma_{q+1}/\gamma^*_{q+1} \right| .$$

2.4 Starting, Error Estimates and Step Size Control

Since the Adams methods is a multistep method it cannot start itself but must rely on a starting procedure that will supply at least q-1 $\vec{f}$ points which, together with a given initial $\vec{f}$ point and a current $\vec{y}$ point, comprise a history upon which it can build. The starting procedure used here is the Runge-Kutta-Shanks procedure START, described elsewhere in this report. The number of function evaluations per step and the order of Runge-Kutta-Shanks method used by START may be varied at will by the user through the formal parameters of ADAMS. This will achieve optimum compatibility with the order of Adams method being used for each given set of differential equations being solved.

Initial step size is determined by the formal parameter DX. The initial trial start will be made with a step H = INTERVAL / C1, where C1 is set to the smallest integer power of two, such that $|H| \leq |DX|$ and $|H| \leq |INTERVAL|/Q$. This causes the procedure ADAMS to take at least one step after starting regardless of the magnitude of DX. If the procedure START cannot meet the error requirements at the initial H, it doubles C1 repeatedly until these requirements can be met.

To minimize running time without introducing errors intolerably large, the error in each component of the final $\vec{Y}$ vector is controlled through the use of the formal parameters $\vec{EA}$ and $\vec{ER}$. $\vec{EA}$ specifies the maximum allowable absolute magnitude of the error in each component of $\vec{Y}$, and $\vec{ER}$ specifies the maximum allowable relative magnitude. These two error control vectors are used in conjunction with the quantity GR = $|1-\gamma_q / \gamma^*_{q+1}|$, which is derived from the bounds (1-4), and a parameter P, chosen from the interval $[\frac{1}{2},1]$ by empirical determination of the randomness of the round-off error in a particular set of differential equations. (P = ½ corresponds to totally random error and P = 1 corresponds to totally additive error.) In practice γ_{q+1} has been used in GR instead of γ_q to be conservative, because the quantity being controlled is only an estimate of the true error.

The extimated error vector $\overrightarrow{ERROR}$ is defined to be $|\vec{y}^{(c)} - \vec{y}^{(p)}|$, where $\vec{y}^{(p)}$ is the $\vec{y}_q(o)$ of (1-3a) and $\vec{y}^{(c)}$ is $\vec{y}_q^{(\nu_f+1)}$ in (1-3b), with ν_f being the first ν for which every component of

$$\overrightarrow{CHANGE} \equiv |\vec{f}_q^{(\nu+1)} - \vec{f}_q^{(\nu)}| = \left| \left(\vec{y}_q^{(\nu+1)} - \vec{y}_q^{(\nu)} \right) / h\beta^*_{q,o} \right|$$

is less than the corresponding component of either

$$\left|\frac{\vec{EA} \cdot GR}{C1^P \cdot 2^{Q+5} \cdot h\beta_{q,o}}\right| \quad \text{or} \quad \left|\frac{\vec{ER} \cdot GR}{C1^P \cdot 2^{Q+5} \cdot h\beta^*_{q,o}} \cdot \vec{f}^{(\nu+1)}_q\right|$$

If any component of $\overrightarrow{ERROR}$ is larger than the corresponding components of both

$$\left|\frac{\vec{EA} \cdot GR}{C1^P}\right|$$

and

$$\left|\frac{\vec{ER} \cdot GR}{C1^P} \cdot \vec{y}^{(c)}\right| ,$$

then $\vec{y}$ is replaced by $\vec{y}^{(b)}$, the step size is halved, and q-1 new $\vec{f}$ points and a new current $\vec{y}$ are obtained from the procedure START. If it is not necessary to halve the step size, then $\vec{y}^{(c)}$ becomes the new $\vec{y}$. If every component of $\overrightarrow{ERROR}$ is smaller for three consecutive steps than the corresponding components of both

$$\left|\frac{\vec{EA} \cdot GR}{C1^P \cdot 2^{Q+5}}\right| \quad \text{and} \quad \left|\frac{\vec{ER} \cdot GR}{C1^P \cdot 2^{Q+5}} \cdot \vec{y}^{(c)}\right|$$

then if there are at least 2q-1 back points in the FH array and there are at least two more steps of the current size necessary to reach XF, the step size is doubled before the next trial step. If it is not necessary either to halve or to double the step size, X is increased by H and a new trial step is made.

The procedure ADAMS continues as described until XF is reached unless repeated halvings and doublings of the step size bring the independent variable to within a fraction of a single step of XF. When this occurs, the fractional step is completed by the Runge-Kutta-Shanks procedure SHANKS, described elsewhere in this report. The order of Runge-Kutta-Shanks method and the number of function evaluations per step used here will be the same for a given integration as those used by the procedure START.

2.5 Flow Diagram and Program Listing

Figure 2 is the flow diagram for the method of Adams, Bashforth and Moulton. The program listing is in Appendix A.

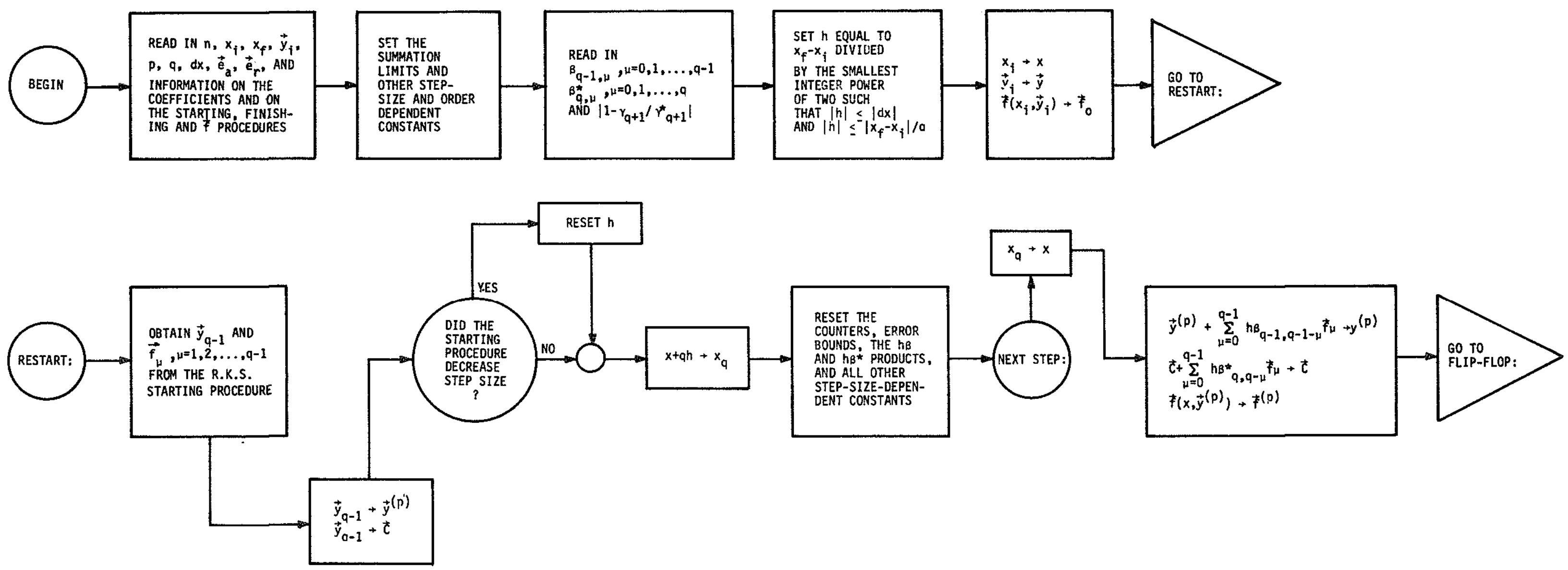

Figure 2. Flow Diagram for the Adams Method.

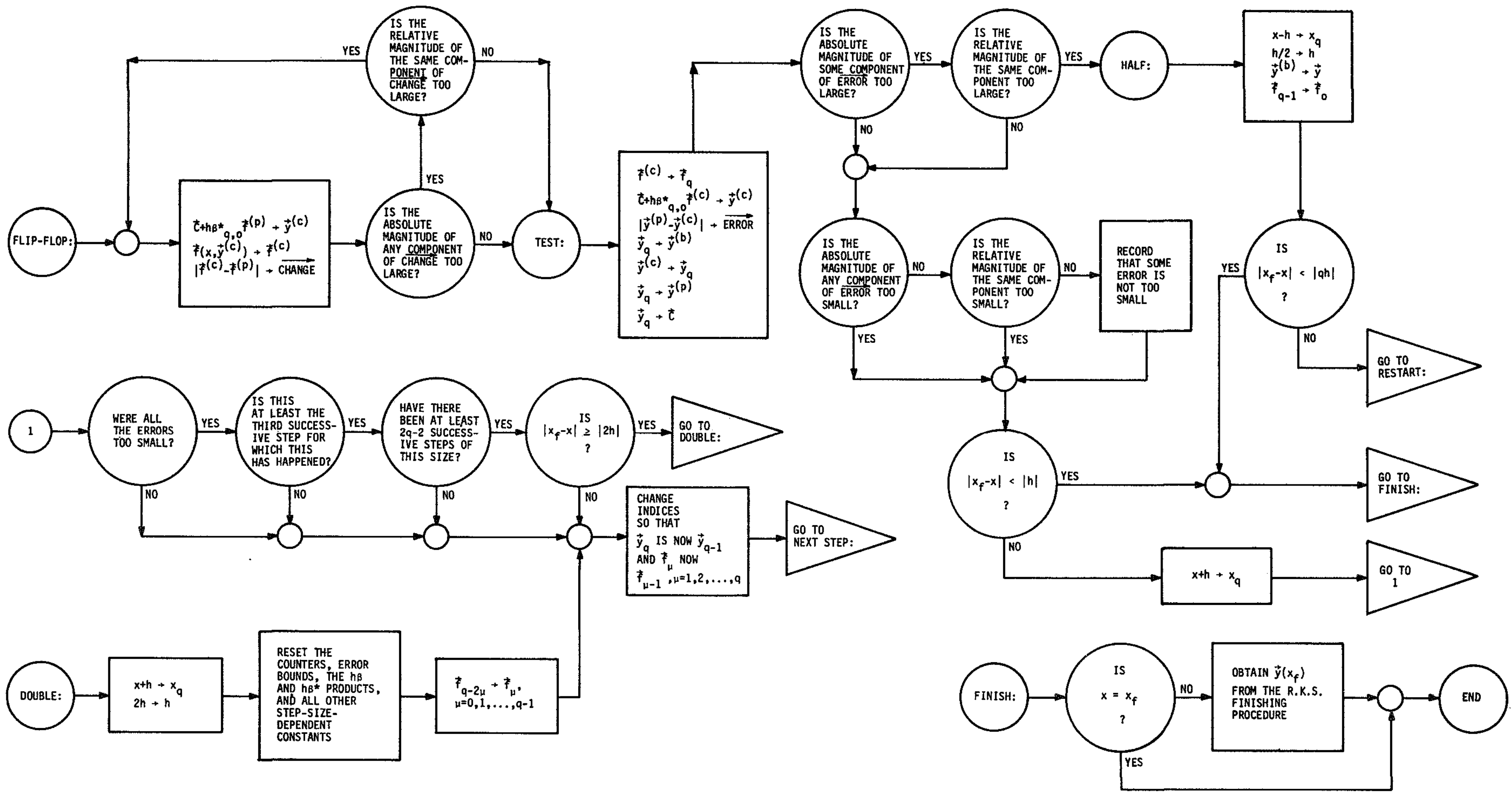

Figure 2. (Continued) Flow Diagram for the Adams Method.

D. The Method of Stetter, Gragg, and Butcher

1. Description of the Method

Following is a discussion of a method for the numerical integration of ordinary differential equations described by J. C. Butcher [13]. In his paper, Butcher presents a modification to the multistep process such that for $k \leqq 7$ (where k = the number of steps) processes of order $2k + 1$ are available.

- A large number of possible multistep methods exist for the numerical integration of the differential equation

$$\frac{dy}{dx} = f(x,y), \; y(x_o) = y_o \tag{1-1}$$

Such methods are usually characterized by an integer k and a set of constants α_1, α_2 ---, $\alpha_k, \beta_o, \beta_1$, ---, β_k. A solution is first found for the variable y at a set of points x_1, x_2, ---, x_{k-1}, (where $x_i = x_o + ih$) and thereafter by the formula:

$$\begin{aligned} y_n &= \alpha_1 y_{n-1} + \alpha_2 y_{n-2} + \text{---} + \alpha_k y_{n-k} \\ &+ h(\beta_o f_n + \beta_1 f_{n-1} + \text{---} + \beta_k f_{n-k}) \end{aligned} \tag{1-2}$$

for $n = k, k+1$, --- where $y_i = y(x_i)$ and $f_i = f(x_i, y_i)$. Dahlquist [3] has shown that if the parameters α and β are chosen under a condition of stability, the order of a method cannot exceed $k + 1$ (if k is odd) or $k + 2$ (if k is even).

A modification to this process is presented by Butcher which consists of the addition to the right-hand side of equation (1-2) of an extra term $h \beta f_{n-\theta}$ where β and θ are additional parameters to be chosen. The modified formula has the form:

$$y_n = \alpha_1 y_{n-1} + \alpha_2 y_{n-2} + --- + \alpha_k y_{n-k}$$

$$+ h\ (\beta\ f_{n-\theta} + \beta_o f_n + \beta_1 f_{n-1} + --- + \beta_k f_{n-k}) \quad (1\text{-}3)$$

A procedure for choosing the coefficients is presented by Butcher. The simplest stable processes are for $k = 1,2,3$ with $\theta = 1/2$ and for $k = 4,5,6$ with $\theta = 1/3$. A stable process also exists for $k = 7$ with $\theta = 13/40$.

The method for implementing the formulas is to estimate $y_{n-\theta}$ and y_n using appropriate predictor formulas, then use these predicted values to evaluate the right-hand side of equation (1-3). The forms of the predictor formulas used are:

$$y_{n-\theta} = A_1 y_{n-1} + A_2 y_{n-2} + --- + A_k y_{n-k}$$

$$+ h\ (B_1 f_{n-1} + B_2 f_{n-2} + --- + B_k f_{n-k}) \quad (1\text{-}4)$$

$$y_n = a_1 y_{n-1} + a_2 y_{n-2} + --- + a_k y_{n-k}$$

$$+ h\ (b\ f_{n-\theta} + b_1 f_{n-1} + b_2 f_{n-2} + --- + b_k f_{n-k}) \quad (1\text{-}5)$$

To use this process, $y_{n-\theta}$ is first estimated using equation (1-4). The value of the function is then determined for $y_{n-\theta}$, and these two results are used in equation (1-5) to determine a value for y_n. The value of the function is then determined for y_n and a final value is then estimated using equation (1-3).

2. The Computer Program

A double precision FORTRAN subroutine was written to implement the integration procedure described above on the UNIVAC 1108 Computer. The procedure was written to integrate a system of differential equations which have the form:

$$\frac{dy}{dx} = \vec{f}(x,\vec{y}), \ \vec{y}(x_o) = \vec{y}_o.$$

Since the integration procedure described by Butcher is a multistep process, it must at all times have a history of back points. The process is, therefore, not self starting; it must rely on some other process to develop the first k steps. The starting procedure used in this implementation is a basic Runge-Kutta procedure as modified by E. B. Shanks and is discussed in paragraph F of this chapter. The starting procedure is called at the beginning of an integration and whenever it is necessary to reduce the step-size.

The step-size control is based on the difference between a predictor and a corrector; the control allows for halving and doubling of the step-size only. Equation (1-5) is used as the predictor (y_{np}) and equation (1-3) is considered to be the corrector (y_{nc}). An estimate of the magnitude of the error in a step is given by the absolute value of the difference in these two quantities. This is used in conjunction with a relative error term ER and an absolute error term EA in the following manner: if

$$\left|\vec{y}_{np} - \vec{y}_{nc}\right| > (\vec{E}A)\left(\frac{DX}{XF - XI}\right)^{EX}$$

and

$$\left|\vec{y}_{np} - \vec{y}_{nc}\right| > \left| (\vec{ER})(\vec{y}_{nc}) \right| \left(\frac{DX}{XF - XI} \right)^{EX}$$

then the step is rejected and the starting procedure is entered with the previous point and a step-size equal to half the old step-size. If

$$\left|\vec{y}_{np} - \vec{y}_{nc}\right| < (\vec{EA}) \left(\frac{DX}{XF - XI} \right)^{EX} \left(\frac{1}{2^{2k + 4}} \right)$$

or

$$\left|\vec{y}_{np} - \vec{y}_{nc}\right| < \left| (\vec{ER}) \left(\frac{DX}{XF - XI} \right)^{EX} (\vec{y}_{nc}) \right| \left(\frac{1}{2^{2k + 4}} \right)$$

for three steps (without an intervening halving of the step-size) and if there is sufficient history of back points, then the step is accepted and the step-size is doubled. If neither the conditions for halving nor the conditions for doubling are met, then the step is accepted and the step-size remains constant. It is important to note that the above criteria must be satisfied for all corresponding components of the vector quantities before the conditions are considered to be met.

The method of ending the integration procedure is to run until the value of the independent variable plus the next step is either equal to or greater than the given final value i.e.

$$X + DX \geq XF.$$

If it is exactly equal, then the procedure takes one more step and quits. If X + DX > XF, then a special ending procedure is called to take the final step. This ending procedure is also basic Runge-Kutta procedure as modified by E. B. Shanks. It is discussed in paragraph B of this chapter. The procedure call for the Butcher procedure must be as follows:

CALL BUTCHER (N, XI, XF, K, EA, ER, DXV, CON, EX, RKC, YIV, RKSNF, RKSODR)

N - the number of dependent variables (integer)

XI - the initial value of the independent variable

XF - the final value of the independent variable

K - the number of steps to be used in the Butcher method (integer)

EA - the acceptable absolute error vector contained in an array of dimension N

ER - the acceptable relative error vector contained in an array of dimension N

DXV - the suggested initial step-size

CON - the array row containing the Butcher constants required for the order of the method specified

EX - the error exponent

RKC - the array containing the Runge-Kutta constants

YIV - the initial values of the dependent variable; upon exiting the Butcher procedure, this array will contain the final values of the dependent variables

RKSNF- the number of function evaluations in the Runge-Kutta-Shanks procedure (integer)

RKSODR-the order of the Runge-Kutta-Shanks procedure (integer)

All variables and arrays not designated integer are double precision.

2.1 Flow Diagram and Program Listing

Figure 3 is the flow diagram for the method of Stetter, Gragg, and Butcher. A listing of the program is given in Appendix A.

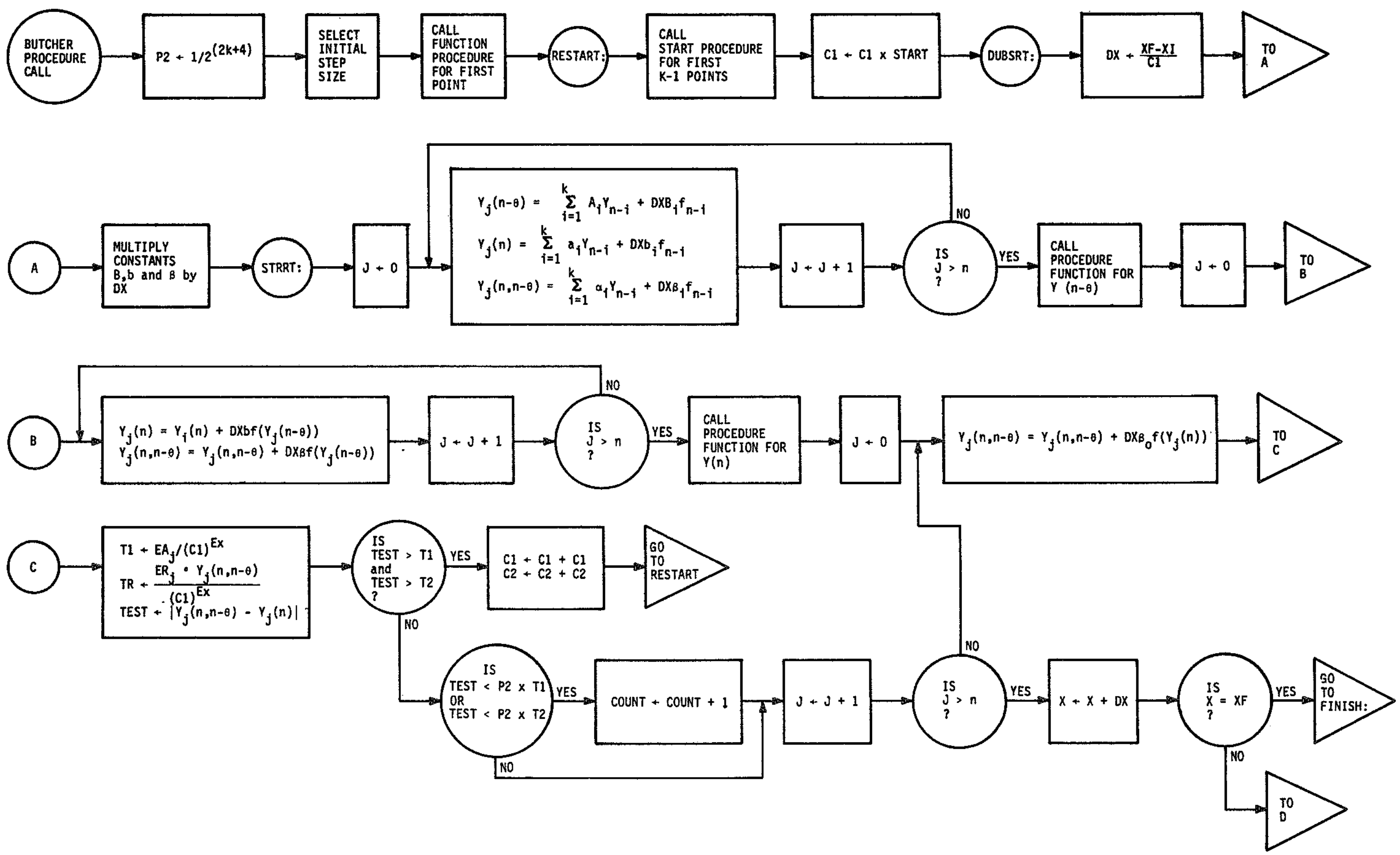

Figure 3. Flow Diagram for the Stetter-Gragg-Butcher Method.

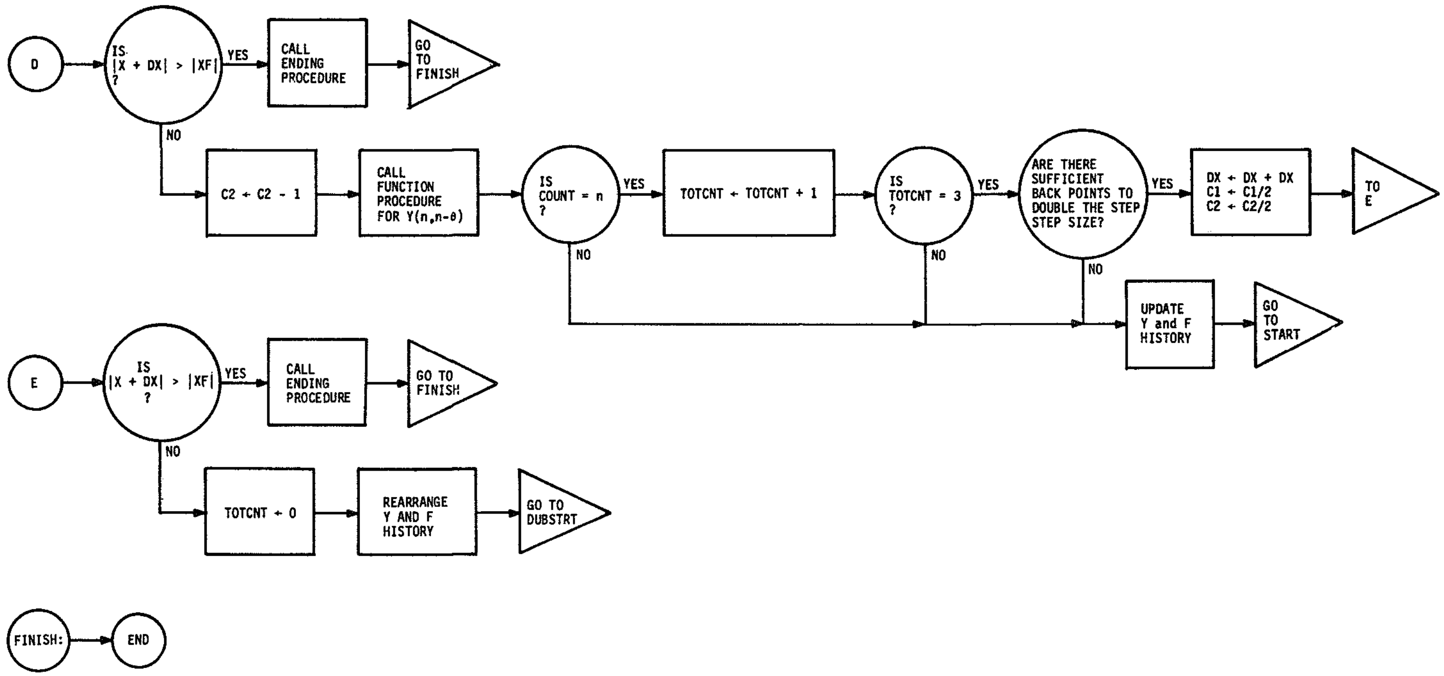

Figure 3. (Continued) Flow Diagram for the Stetter-Gragg-Butcher Method.

E. The Cowell Method

1. Description of the Method

Cowell's method as described herein is a multistep predictor-corrector method for the numerical solution of the first-order vector differential equation

$$\vec{y}\,'(x) = \frac{d}{dx}\,\vec{y}(x) = \vec{f}\,(x,\vec{y}(x)),\ \vec{y}\,(x_o) = \vec{y}_o. \qquad (1\text{-}1)$$

A complete derivation and description of Cowell's method can be found in [9] and [11]; only the essential formulas are included here.

The following notation is adopted. Let q be an even positive integer, m = q/2, h be the step size (assumed to be constant over some set of calculations),

$$x_n = x_o + nh,\ \vec{y}_n = \vec{y}\,(x_n),\ \text{and}\ \vec{f}_n = \vec{f}\,(x_n,\ \vec{y}_n).$$

The predictor formula is

$$\vec{y}_n = h\,[\overrightarrow{\delta^{-1}}\,f_{n-\frac{1}{2}} + \sum_{j=0}^{q} P_j\,\vec{f}_{n-1-j}], \qquad (1\text{-}2)$$

The corrector formula is

$$\vec{y}_n = h\,[\overrightarrow{\delta^{-1}}\,f_{n-\frac{1}{2}} + \sum_{j=0}^{q} C_j\,\vec{f}_{n-j}]\ , \qquad (1\text{-}3)$$

and the mid-range formula is

$$\vec{y}_n = h\,[\overrightarrow{\delta^{-1}}\,f_{n-\frac{1}{2}} + \sum_{j=0}^{q} M_j\,\vec{f}_{n+m-j}]. \qquad (1\text{-}4)$$

The predictor formula gives $\vec{y}_n$ in terms of $\overrightarrow{\delta^{-1}f}_{n-\frac{1}{2}}$ and the function values at the previous q+1 points; the corrector formula gives a new value of $\vec{y}_n$ in terms of $\overrightarrow{\delta^{-1}f}_{n-\frac{1}{2}}$, the old value of $\vec{y}_n$, and the function values at the previous q points; the mid-range formula gives a value of $\vec{y}_n$ in terms of $\overrightarrow{\delta^{-1}f}_{n-\frac{1}{2}}$ and the function values at the q+1 consecutive points centered around x_n.

The equation

$$\overrightarrow{\delta^{-1}f}_{n-\frac{1}{2}} = \overrightarrow{\delta^{-1}f}_{n-1-\frac{1}{2}} + \vec{f}_{n-1} \qquad (1\text{-}5)$$

completes the set of formulas necessary for the numerical solution of (1-1).

If it is assumed that

$$\left[\vec{f}_i\right]_{i=0}^{q} \quad \text{and } \vec{y}_m$$

have been obtained by some starting procedure, the mid-range formula (1-4) can be applied with n=m to obtain

$$\overrightarrow{\delta^{-1}f}_{m-\frac{1}{2}}.$$

Equation (1-5) can then be applied m times to obtain

$$\overrightarrow{\delta^{-1}f}_{q-\frac{1}{2}}$$

For each positive integer i

$$\overrightarrow{\delta^{-1}f}_{q+i-\frac{1}{2}}$$

can be computed from

$$\overrightarrow{\delta^{-1}f}_{q+i-1-\frac{1}{2}}$$

and $\vec{f}_{q+i-1}$ using (1-5); $\vec{y}_{q+i}$ can be computed using the predictor (1-2); $\vec{f}_{q+i}$ can be computed from the predicted value; $\vec{y}_{q+i}$ can be computed using the corrector (1-3); $\vec{f}_{q+i}$ can be computed from the corrected value; if necessary, iteration can be resorted to, using (1-3), until the last two computed values of $\vec{y}_{q+i}$ agree to sufficient accuracy. For any $j \geq m$ a value of $\vec{y}_{q+j-m}$ can be obtained from the mid-range formula (1-4) and compared with the value obtained from the predictor-corrector step. If the two values of $\vec{y}_{q+j-m}$ are in sufficient agreement, the values up through $\vec{y}_{q+j}$ are considered acceptable; if not, $\vec{y}_{q+j-m}$ is considered the last acceptable value and all values beyond are rejected.

Hence, the knowledge of (1-2), (1-3), (1-4), and (1-5) is sufficient to apply Cowell's method in the numerical solution of (1-1). The coefficients $\left\{P_j\right\}_{j=0}^{q}$, $\left\{C_j\right\}_{j=0}^{q}$, and $\left\{M_j\right\}_{j=0}^{q}$ are given in [11] for q=4, 6, 8, 10, 12, 14, and 16.

2. <u>The Computer Program</u>

The Cowell computer program is a UNIVAC 1108 FORTRAN V double precision subroutine called as follows:

call Cowell (n, xi, xf, y, ea, er, p, dxv, rksfn, rksrdr, rkscff, q, cwllcf)

The parameters of the procedure are defined as follows:

<u>n</u> - the number of dependent variables in the vectors $\vec{y}$ and $\vec{f}$ (integers);

<u>xi</u> - x_o, the starting value of the independent variable x;

<u>xf</u> - the final value of the independent variable x;

<u>y</u> - the array in which $\vec{y}_o = \vec{y}$ (xi) is located upon entry and in which $\vec{y}$ (xf) is located upon exit;

ea - the array containing the absolute error vector;

er - the array containing the relative error vector;

p - the exponent used in step size control;

rksfn - the number of function evaluations used in the Runge-Kutta-Shanks starting and closing procedures (integer);

rksrdr - the order of the Runge-Kutta-Shanks closing procedure (integer);

rkscff - the array containing the Runge-Kutta-Shanks coefficients for the starting and closing procedures;

q - the even integer used in describing Cowell's method (integer);

cwllcf - the array containing the Cowell coefficients.

All variables not designated integer are double precision. All arrays are double precision.

The procedure performs the numerical integration of (1-1) from x = xi to x = xf. The step size h used is always the length of the interval xf - xi divided by a power of 2 in order to avoid error building in the independent variable, and two counters c1 and c2 are kept. c1 is always a positive, integral power of 2, and h = (xf - xi)/c1. c2 is the number of steps necessary to step from the present x to xf using the current step size h. Initially c2 = c1; as each step is taken c2 is decremented by one and the present value of x is computed by x = xf - h c2. If h is halved, c1 and c2 are doubled; if h is doubled, c1 and c2 are halved. Hence, c2 need not be integral.

The error vectors $\vec{ea}$ and $\vec{er}$, like $\vec{y}$, have n components. (Although the base of the arrays y, ea, and er is one, the n components are placed in positions 2, 3,.. n +1 of the arrays.) The procedure's error control attempts to guarantee that, in integrating from xi to xf, each component of $\vec{y}$ will not be in absolute error more than the corresponding component of $\vec{ea}$ and will not be in relative error more than the corresponding component of $\vec{er}$. At each step, the procedure requires that for each i, $1 \leq i \leq n$, either the absolute error in y [i] does not exceed ea [i]/(cl^p) or the relative error in y [i] does not exceed er [i]/(cl^p).

If p = 1 and $\vec{er} = 0$ then the accumlated error in any component of $\vec{y}$ should not exceed the corresponding component of $\vec{ea}$. If the error is assumed to accumulate randomly as the square root of the number of steps, $p = \frac{1}{2}$ and $\vec{er} = 0$ will cause the accumulated error in any component of $\vec{y}$ to be approximately the corresponding component of $\vec{ea}$.

If p = 1 and $\vec{ea} = 0$ then the accumulated error in any component of $\vec{y}$ should not exceed the corresponding component of $\vec{er}$ times the largest value assumed by that component of $\vec{y}$ during the integration. If the error is assumed to accumulate randomly as the square root of the number of steps, $p = \frac{1}{2}$ and $\vec{ea} = 0$ will cause the accumulated error in any component of $\vec{y}$ to be approximately the corresponding component of $\vec{er}$ times some average value assumed by that component of $\vec{y}$ during the integration.

The subroutine FUNCTI which computes $\vec{f} = \vec{f}$ (x,y) has the following call:

call FUNCTI (n, x, yv, fv);

The parameters of the procedure FUNCTI are defined as follows:

<u>n</u> - the number of dependent variables in the vectors $\vec{y}$ and $\vec{f}$ (integer)

<u>x</u> - the value of the independent variable

<u>yv</u> - the array in which $\vec{y}$ is stored

<u>fv</u> - the array in which $\vec{f}$ is stored after computation

The procedure start is the general multistep method starting procedure described in paragraph F of this chapter. The procedure shanks is the Runge-Kutta-Shanks integration procedure described in paragraph B of this chapter. The coefficient array rkscff contains the Runge-Kutta-Shanks coefficients in the order required by the procedures start and shanks. The number of function evaluations rksfn is required by both start and shanks; the order rksrdr is required by shanks.

The array cw11cf contains the coefficients of (1-2), (1-3), and (1-4) in the order $P_0, P_1, \ldots, P_q, C_0, C_i, \ldots, C_q, M_0, M_1, \ldots, M_q$; P_0 is in the first position of the array.

The suggested initial step size dxv is optional. The procedure first sets c1 = 2 and doubles c1 until c1 $\geq$ q. If dxv = 0 or dxv $\neq$ 0 and $h \leq |dx|$ then c1 is left alone. Otherwise, c1 is doubled until $h \leq |dx|$.

The integration now begins.

$$\vec{f}_0 = \vec{f}(x_0, \vec{y}_0)$$

is computed. The start procedure is called to obtain

$$\left[\vec{f}_i\right]_{i=1}^{q}, \ \vec{y}_m, \ \vec{y}_q \text{ and } x_q.$$

c1 and c2 are adjusted if h was changed by the start procedure. c2 is decremented by q since q steps took place in the start procedure. If c2< m, closing takes place. Otherwise,

$$\overrightarrow{\delta^{-1} f_{m-\frac{1}{2}}}$$

is calculated from

$$\left[\vec{f}_i\right]_{i=0}^{q}$$

and $\vec{y}_m$ using the mid-range formula (1-4). m applications of (1-5) yield

$$\overrightarrow{\delta^{-1} f_{q-\frac{1}{2}}}$$

and n is set equal to q.

For $1 \le i \le m$ the following set of steps takes place. c2 is decremented by 1, and x_{n+i} is calculated.

$$\overrightarrow{\delta^{-1} f_{n+i-\frac{1}{2}}}$$

is calculated from

$$\overrightarrow{\delta^{-1} f_{n+i-1-\frac{1}{2}}}$$

and $\vec{f}_{n+i-1}$ using (1-5). $\vec{y}_{n+i}$ is calculated using the predictor (1-2), and $\vec{f}_{n+i}$ is calculated. $\vec{y}_{n+i}$ is next calculated using the corrector (1-3), and $\vec{f}_{n+i}$ is again calculated. Let $\vec{v}$ be the vector which is the absolute value of the difference between the last two calculated values of $\vec{y}_{n+i}$. Each component of $\vec{v}$ is compared with the corresponding component of $\vec{ea}/(10 \cdot c1^{p})$ for absolute error and with the product of the corresponding components of $\vec{er}/(10 \cdot c1^{p})$ and the last calculated value of $\vec{y}_{n+i}$ for relative error. If any component of $\vec{v}$ exceeds in both the absolute and the relative error tests, the steps which calculate $\vec{y}_{n+i}$ using the corrector (1-3), calculate $\vec{f}_{n+i}$ from the value of $\vec{y}_{n+i}$, and which test the last two calculated values of $\vec{y}_{n+i}$ are repeated. When each component of $\vec{v}$ does not exceed in either the absolute or the relative error test, the last values of $\vec{y}_{n+i}$ and $\vec{f}_{n+i}$ are retained.

The mid-range formula (1-4) is now used to calculate a new value of $\vec{y}_n$ from

$$\left[\vec{f}_{n+i}\right]_{i=m}^{m}$$

and

$$\overrightarrow{\delta^{-1} f_{n-\frac{1}{2}}}.$$

Let $\vec{v}$ be the vector which is the absolute value of the differences between the new value of $\vec{y}_n$ and the previously calculated value of $\vec{y}_n$. If sufficient history is available for doubling the step size, i.e., $n > q + m$, each component of $\vec{v}$ is compared with the corresponding component of $\vec{ea}/(10 \cdot c1^p \cdot 2^{q+3})$ for absolute error and with the product of the corresponding components of $\vec{er}/(10 \cdot c1^p \cdot 2^{q+3})$ and the new value of $\vec{y}_n$ for relative error.

If each component of $\vec{v}$ does not exceed in either the absolute or the relative error tests, the last m steps are accepted, c1 and c2 are halved, and the step size is doubled. If $c2 < m$, closing takes place. Otherwise

$$\left[\vec{f}_i\right]_{i=0}^{q}$$

becomes

$$\left[\vec{f}_{n-m+2i}\right]_{i=0}^{q}$$

$\vec{y}_m$ becomes $\vec{y}_{n-m}$, $\vec{y}_q$ becomes $\vec{y}_{n+m}$, x_q becomes x_{n+m}, and, as if the starting procedure had calculated these values, control returns to the step where

$$\overrightarrow{\delta^{-1} f_{m-\frac{1}{2}}}$$

is calculated using the mid-range formula (1-4).

If any component of $\vec{v}$ exceeds in both the absolute and the relative error tests, this component and each untested component is compared with the corresponding component of $\vec{ea}/(10 \cdot c1^p)$ for absolute error and with the product of the corresponding components of $\vec{er}/(10 \cdot c1^p)$ and the new value of $\vec{y}_n$ for relative error. If each component of $\vec{v}$ does not exceed in either the absolute or the relative error test, the last m steps are accepted and the step size remains unchanged. If c2 < m, closing takes place. Otherwise, n becomes n + m and control returns to the steps which calculate

$$\left[\vec{y}_{n+i}\right]_{i=1}^{m} .$$

If any component of $\vec{v}$ exceeds in both the absolute and the relative error tests, the last m steps are rejected, c2 is incremented by m, c1 and c2 are doubled, and the step size is halved. $\vec{f}_0$ becomes $\vec{f}_n$, $\vec{y}_0$ becomes $\vec{y}_n$, x_0 becomes x_n, and control is returned to the step which calls the start procedure.

If sufficient history is not available for doubling, control transfers as if the first component of $\vec{v}$ exceeded both the first component of $\vec{ea}/(10 \cdot c1^p \cdot 2^{q+3})$ and the product of the first components of $\vec{er}/(10 \cdot c1^p \cdot 2^{q+3})$ with the first component of the new value of $\vec{y}_n$.

Closing takes place whenever m steps at the present step size would carry the integration beyond xf, i.e., whenever c2 < m. If c2 > 0, the Runge-Kutta-Shanks procedure is used to integrate from the present value of x to xf; if c2 = 0, the present value of x is xf. In either case, the integration is now complete.

Several efficiency measures are employed in the program. First, the coefficients

$$\left[P_j\right]_{j=0}^{q},$$

$$\left[C_j\right]_{j=0}^{q},$$

and

$$\left[M_j\right]_{j=0}^{q}$$

are multiplied by the step size h and stored as multiplied until the step size changes. Second, the vectors $\vec{ea}/(10 \cdot cl^P)$, $\vec{er}/(10 \cdot cl^P)$, $\vec{ea}/(10 \cdot cl^P \cdot 2^{q+3})$, and $\vec{er}/(10 \cdot cl^P \cdot 2^{q+3})$ are calculated from $\vec{ea}$ and $\vec{er}$ and stored as calculated until the step size changes. Third, the corrector partial sum

$$\overrightarrow{h\delta^{-1}f_{n-\frac{1}{2}}} + h\sum_{j=1}^{q} C_j\vec{f}_j$$

is computed and stored at each step; successive applications of the corrector only require adding $h \cdot C_0 \cdot f_n$ to obtain $\vec{y}_n$. Fourth, during applications of the corrector, two arrays are used to store the last two calculated values of $\vec{y}_n$; a flag is used to mark the last calculated value so that the next value is placed in the unflagged array and the flag is switched. This avoids transfer from array to array as successive corrector iterates are computed. Fifth, cyclic indexing is used to avoid moving the function value history after each step or set of steps unless doubling takes place.

One unusual condition can result. If, during any step taken in computing

$$\left[\vec{y}_{n+i}\right]_{i=1}^{m},$$

the number of times through the corrector exceeds eight, control transfers as if the set of m steps has been completed and rejected; i.e., a step size halving was called for with a restart beginning at $\vec{y}_n$.

2.1 Flow Diagram and Program Listing

Figure 4 is the flow diagram for the Cowell method. The program listing is in Appendix A.

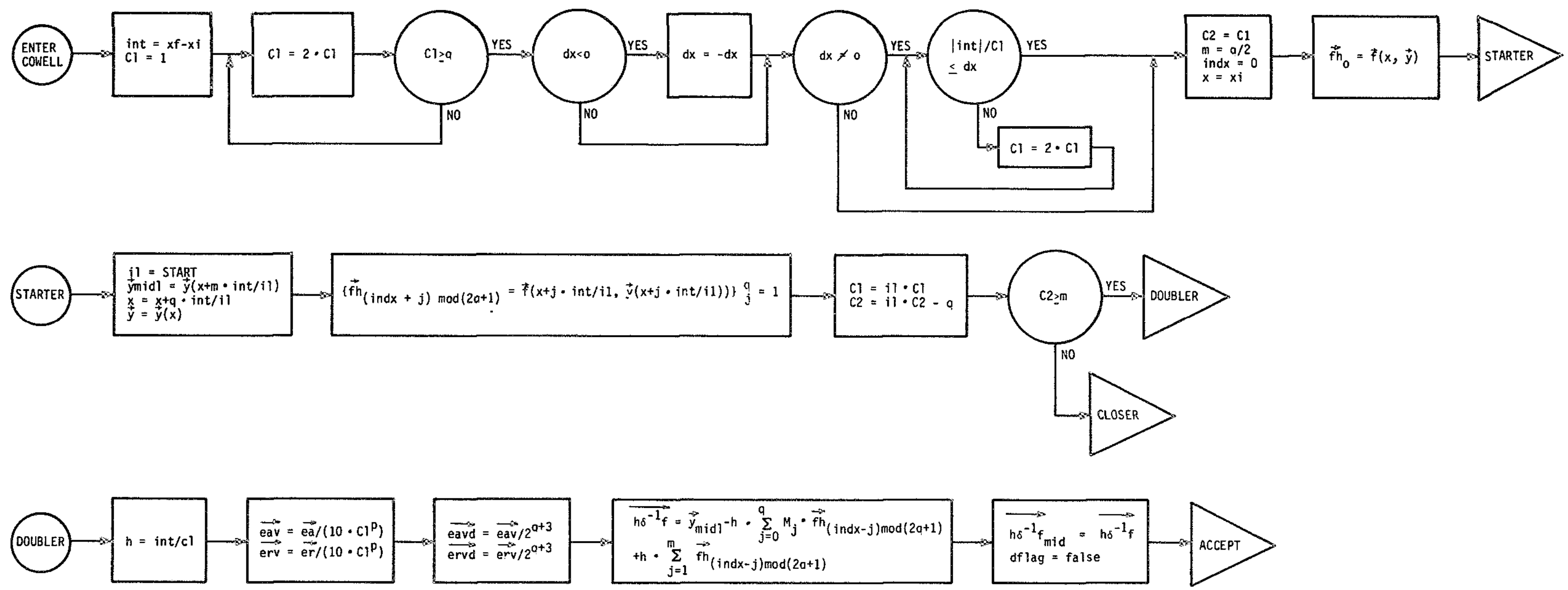

Figure 4. Flow Diagram for the Cowell Method.

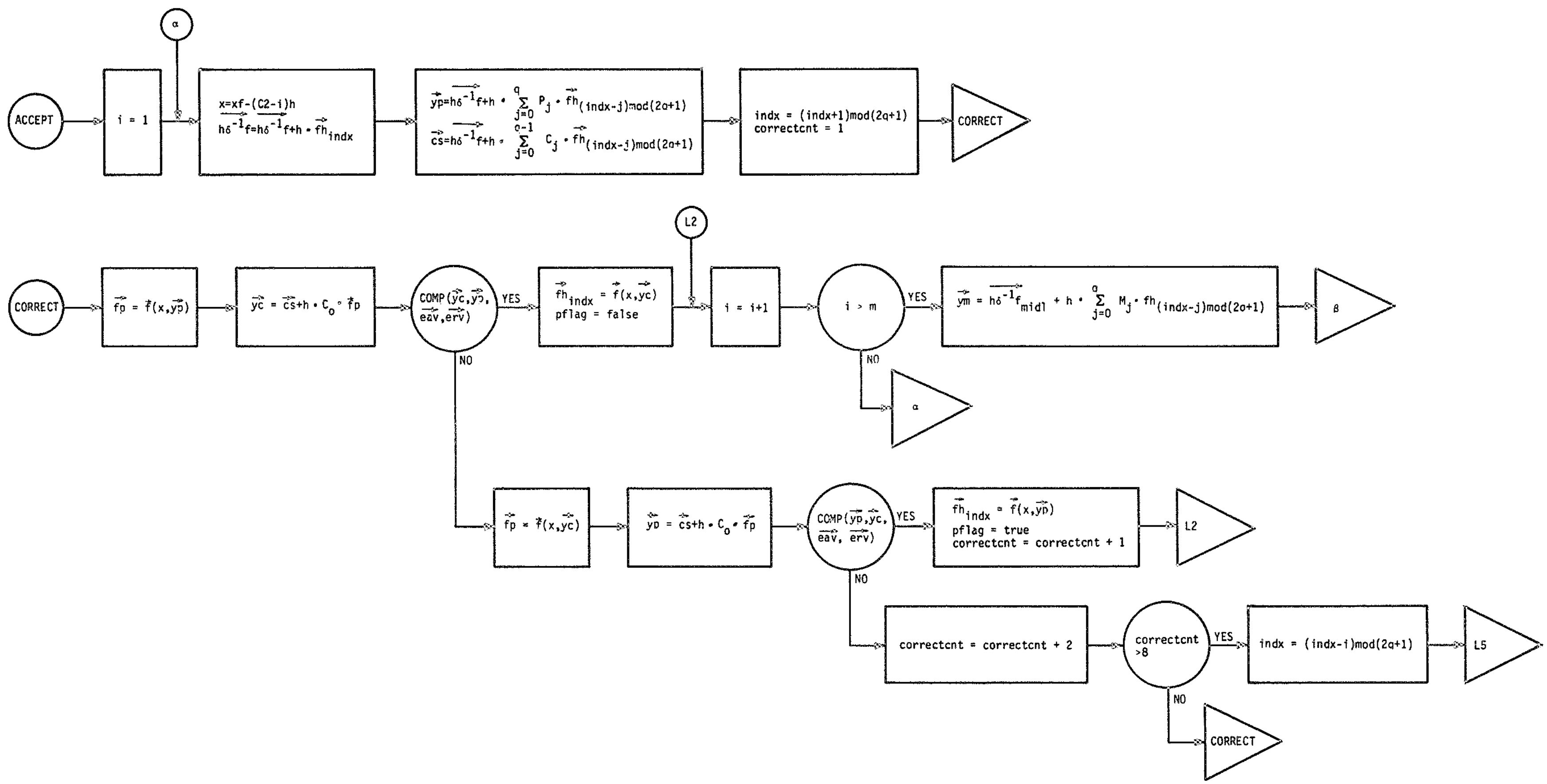

Figure 4. (Continued) Flow Diagram for the Cowell Method.

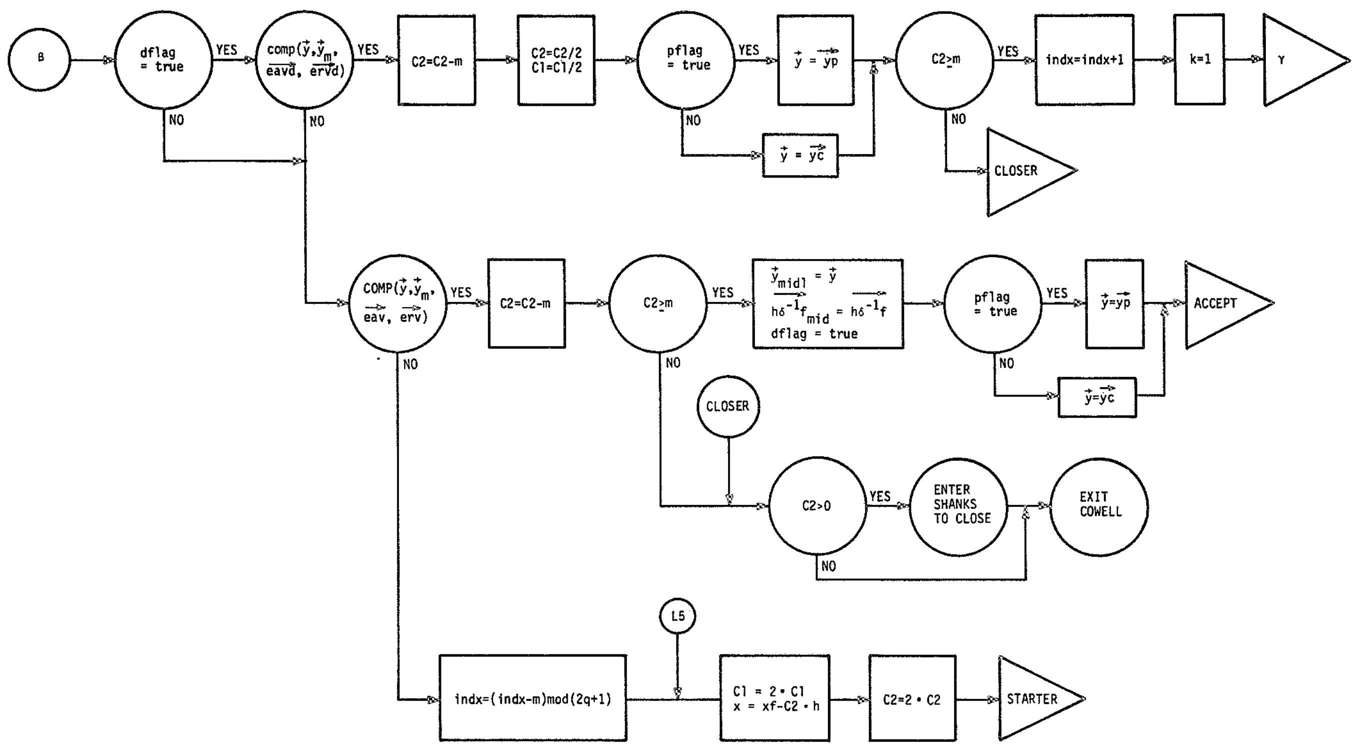

Figure 4. (Continued) Flow Diagram for the Cowell Method.

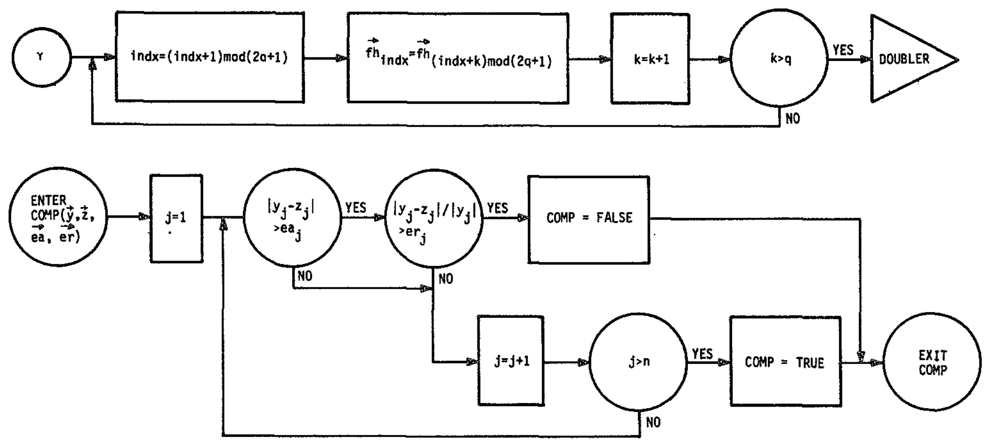

Figure 4. (Continued) Flow Diagram for the Cowell Method.

F. The General Multistep Method Starting Subroutine

1. Introduction

The general multistep method starting procedure is a UNIVAC 1108 FORTRAN V double-precision Runge-Kutta-Shanks subroutine used for obtaining starting values for the Adams, Butcher, and Cowell multistep methods. Start is a real valued function called as follows:

start (n, xi, xf, icl, ea, er, m, x, yiv,
yh, fh, yfv, cyi, cym, pa, p,
fneval, rkscns)

2. Description of the Program

The parameters of the function are defined as follows:

n - the number of dependent variables (integer);

xi - the starting value of the independent variable x passed to the multistep method;

xf - the final value of the independent variable x passed to the multistep method;

icl - the integer counter (xf - xi)/h from the multistep method (integer);

ea - the absolute error vector passed to the multistep method;

er - the relative error vector passed to the multistep method;

m - the number of history points to be calculated by start (integer);

x - the value of the independent variable at which start begins its integration;

yiv - the array which contains on entry for Adams and Cowell the values of the dependent variables at x and which contains on exit for Cowell the values of the dependent variable at the mth point calculated by start;

yh - the array which contains on entry for Butcher in row cyi the values of the dependent variables at x and which contains on exit for Butcher the values of the dependent variables at each of the m points calculated by start;

fh - the array which contains on entry in row cyi the function values at x and which contains on exit the function values at each of the m points calculated by start;

yfv - the array which contains on exit the values of the dependent variables at the mth point calculated by start for Adams or the m/2th point calculated by start for Cowell;

cyi - the cyclic index identifying on entry the row of yh in which the values of the dependent variables at x are stored for Butcher and the row of fh in which the function values at x are stored for any method (integer);

cym - the number of rows in the arrays yh and fh (integer);

pa - the parameter which is zero for Adams, one for Cowell, two for Butcher (integer);

p - the exponent such that the absolute error at each step is not to exceed ea/cl^p and the relative error at each step is not to exceed er/cl^p;

fneval - the number of function evaluations required by the Runge-Kutta-Shanks procedure (integer);

rkscns,- the array which contains the Runge-Kutta-Shanks coefficients in the same order as required by the procedure shanks described in section B.

The value of start on exit is two to the power of the number of halvings which took place within start (a real).

All variables and arrays not designated otherwise are double precision.

Although the base of the arrays _ea_, _er_, _yiv_, and _yfv_ and of the rows of _yh_ and _fh_ is one, the n components are placed in position 2, . . ., n+1 and the first position is unused.

The procedure attempts to calculate m (if m is even and positive) or m + 1 (if m is odd) Runge-Kutta-Shanks steps of size h = (xf - xi)/c1. After each even step of size h is taken, one step of size 2h is taken over the interval spanned by the two steps of size h. The absolute value of the differences in each dependent variable between the 2h-step and the second h-step is compared with the corresponding component of $\vec{ea}/(c1/2)^p$ for absolute error and with the product of the corresponding component of $\vec{er}/(c1/2)^p$ and the corresponding dependent variable value from the second h-step for relative error. If each component of the difference does not exceed in either the absolute or the relative error test and m steps have not yet been taken, the process of two h-steps, one 2h-step, and test is continued. If any component of the difference exceeds in both the absolute and the relative error tests, c1 is doubled, h is halved, and integration begins again at x. The first step of previous size h was saved and becomes the first step of present size 2h.

The m calculated function values from h-steps are placed in rows (cy1 + 2) mod cym+1, (cyi + 3) mod cym+1, ..., (cyi+m+1) mod cym+1 of the array fh. For Butcher, the corresponding dependent variable values from h-steps are placed in the corresponding rows of the array yh; if m is odd, the values of the dependent variable after h-step m + 1 are placed in row (cyi + m + 2) mod cym+1 of yh. For Adams, the dependent variable values from h-step m are placed in the array yfv. For Cowell, the dependent variable values from h-step m are placed in the array yiv and from h-step m/2 (m is always even for Cowell) are placed in yfv. If m is zero, no calculation takes place.

Start calls an external subroutine _runkut_.

2.1 Flow Diagram and Program Listing

Figure 5 is the flow diagram for the starting procedure. The program listing is in Appendix A.

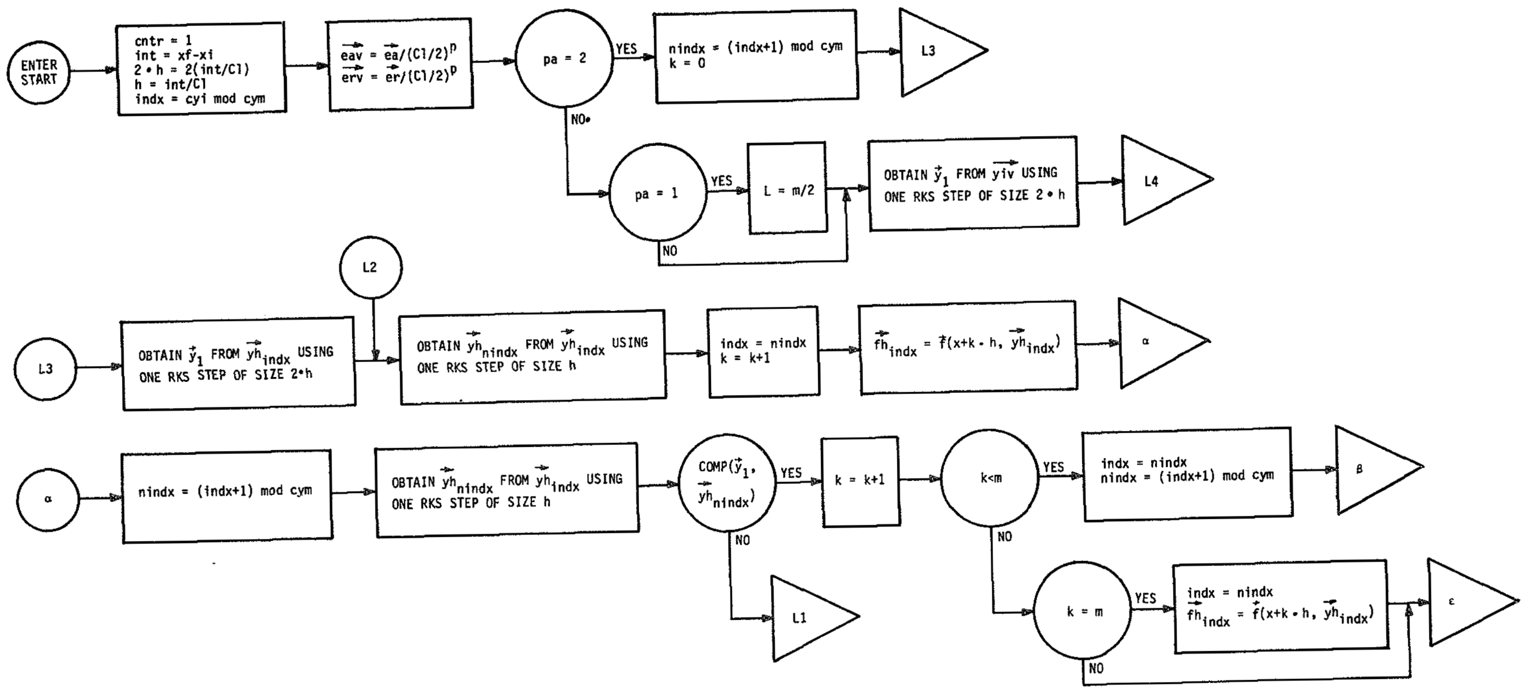

Figure 5. Flow Diagram for the General Multistep Method Starting Procedure.

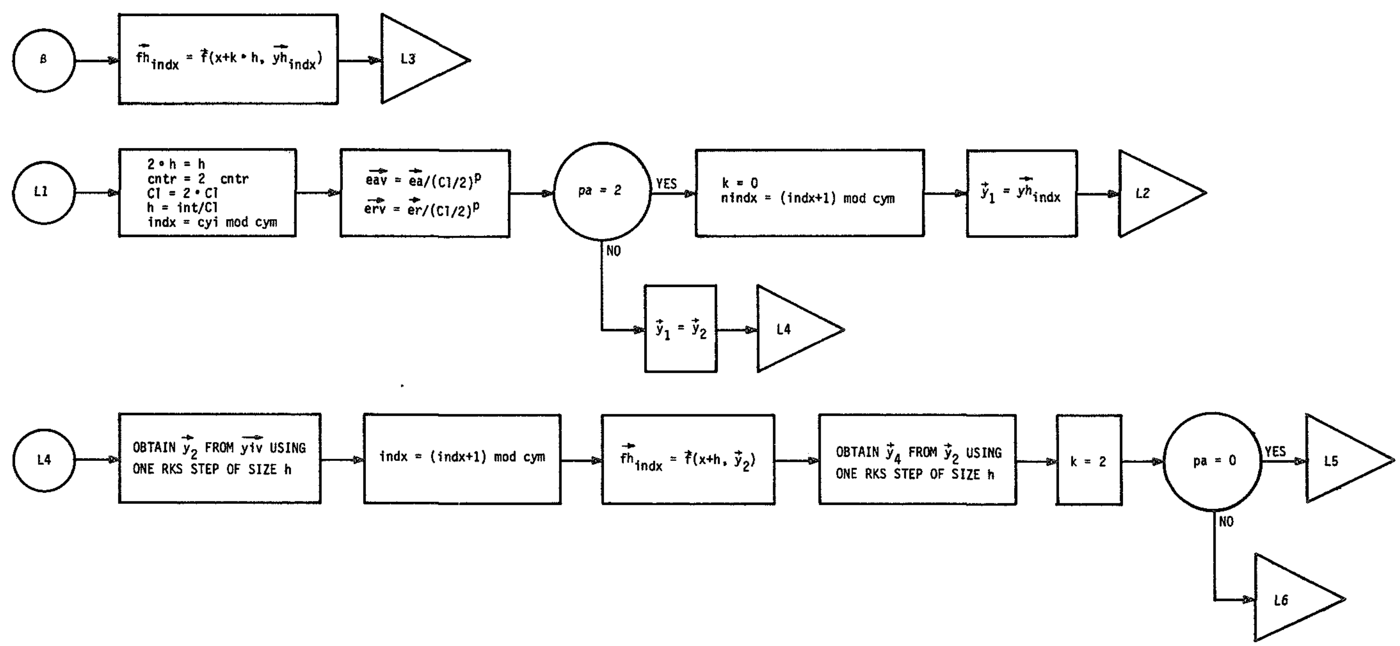

Figure 5. (Continued) Flow Diagram for the General Multistep Method Starting Procedure.

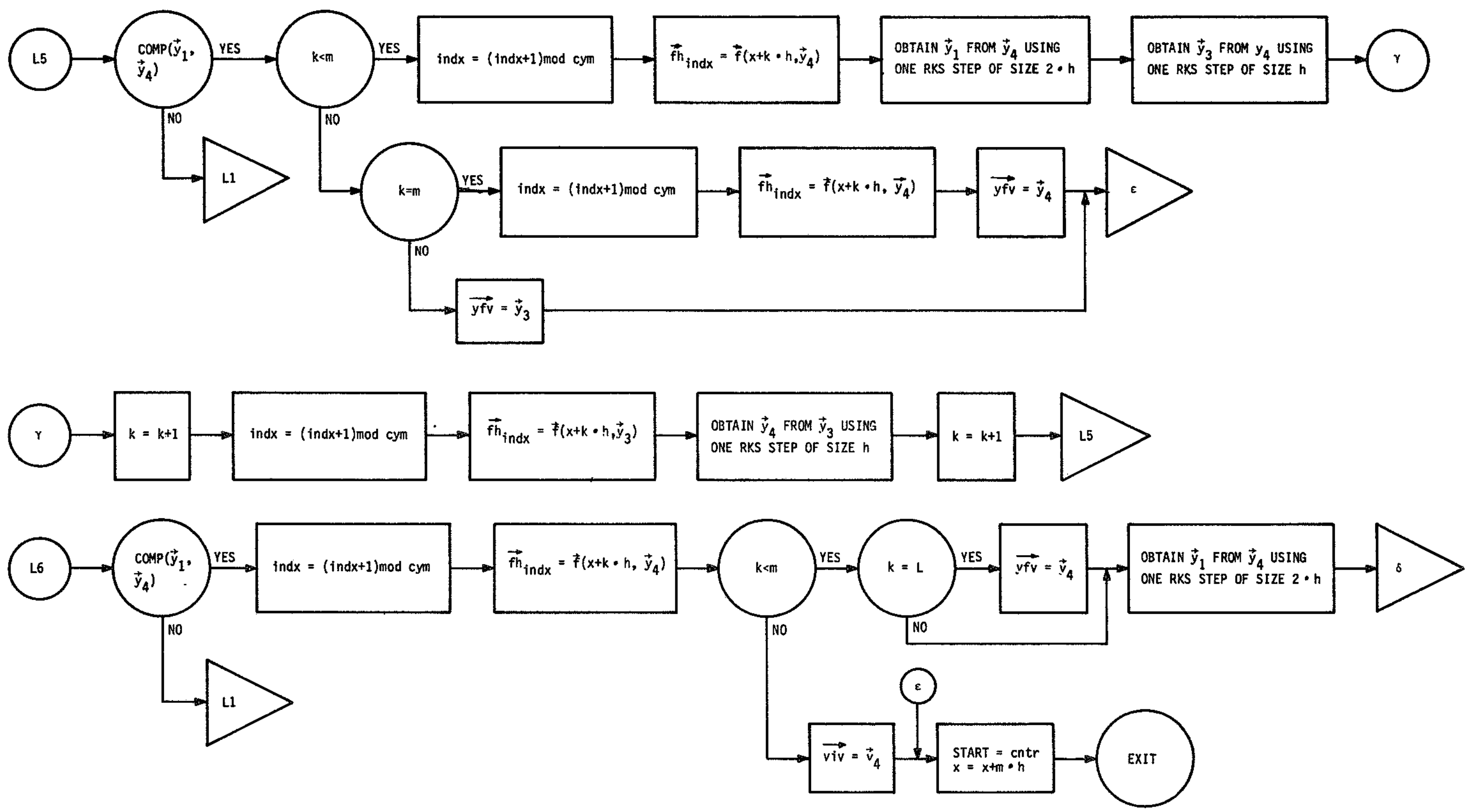

Figure 5. (Continued) Flow Diagram for the General Multistep Method Starting Procedure.

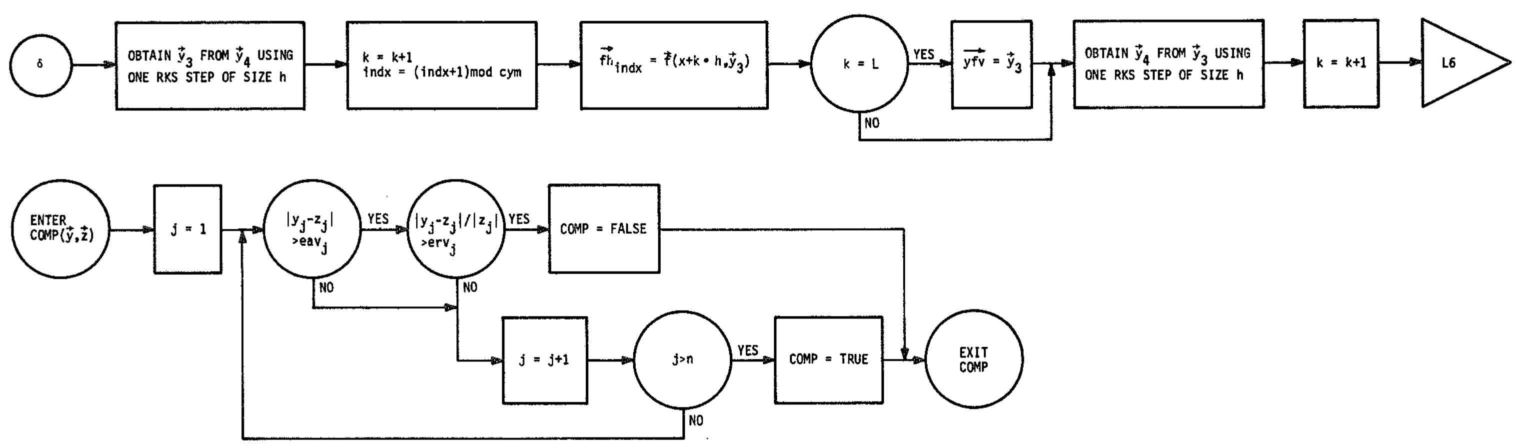

Figure 5. (Continued) Flow Diagram for the General Multistep Method Starting Procedure.

G. The Derivative Subroutine FUNCTI

A procedure for calculating the vector $\vec{y}' = \vec{f}\left(x, \vec{y}(x)\right)$ must be an external subroutine characterizing the set of differential equations to be solved. This subroutine is called by each of the integration subroutines and must itself have the following formal parameter list:

Identifier	Type	Usage or Meaning
N	Integer	Number of equations being integrated.
X	Double	Current value of the independent variable.
YV	Double Array	Current dependent variable vector (input).
FV	Double Array	$\vec{f}$ value vector (output).

The components of the $\vec{y}$ vector are contained in positions 2 through N + 1 of YV, and the components of $\vec{f}$ are returned in corresponding positions 2 through N + 1 of FV. The call on FUNCTI is as follows:

```
CALL FUNCTI (N,X,YV,FV).
```

A skeleton FUNCTI subroutine is given in the listings in Appendix A.

III. RESULTS AND CONCLUSIONS

A. The Test Problem

The results presented here are for the test problem of 2-dimensional motion in an inverse square attractive force field using x-y coordinates. The equations are

$$\frac{dv_x}{dt} = - \frac{\partial}{\partial x} V(x,y)$$

$$\frac{dv_y}{dt} = - \frac{\partial}{\partial y} V(x,y)$$

$$\frac{dx}{dt} = v_x$$

$$\frac{dy}{dt} = v_y$$

where $V(x,y) = - (x^2 + y^2)^{-\frac{1}{2}}$

The initial conditions were chosen to give an elliptical orbit of eccentricity 0.8. In this orbit the error and step size controls will produce several step size changes as the orbit is traversed. The initial conditions are:

at $t = 0$,

$x = 0.2$,

$y = 0$,

$v_x = 0$,

$v_y = 3.0$.

One complete orbit was run (t final = 2π) and the error taken as the square root of the sum of the squares of the differences between initial and final conditions.

The accuracy range investigated was $\sim 10^{-7}$ to $\sim 10^{-12}$. The UNIVAC 1108 maintains ~18 significant figures (decimal) in double precision.

B. Results

Results are presented here for test runs made with the following methods and orders:

Shanks formulas 7-7, 7-9, 8-10 and 8-12 (here the first number gives the order, the second gives the number of function elevations per step);

Adams method orders 7, 9, 11, 13;

Cowell's method orders 7, 9, 11, 13;

Butcher's formulas orders 5, 7, 9, 11, 13.

All of the multistep methods used Shanks 8-10 for start and restart for all orders. These orders were chosen to give as close as comparison as possible across methods at corresponding orders.

Figure 6 shows what might be expected of the error behavior as a function of number of steps taken to perform a given integration. In the large step-size region (small number of steps), the truncation error dominates and decreases as N^{-n}, where N is the number of steps, and n is the order of the error. In the small step-size (large number of steps), the rounding error should dominate. If rounding errors behave like randon variables, one would expect the rounding error to increase as $N^{\frac{1}{2}}$.

Figure 7 shows results of runs made with Shanks' formulas. Plotted here is error vs. number of function evaluations and error vs. computer time to complete the integration. The labelings indicate the order and number of function evaluations per step, respectively. These points lie on the truncation error part of the curve and show the advantage of the higher order methods at high accuracy requirements.

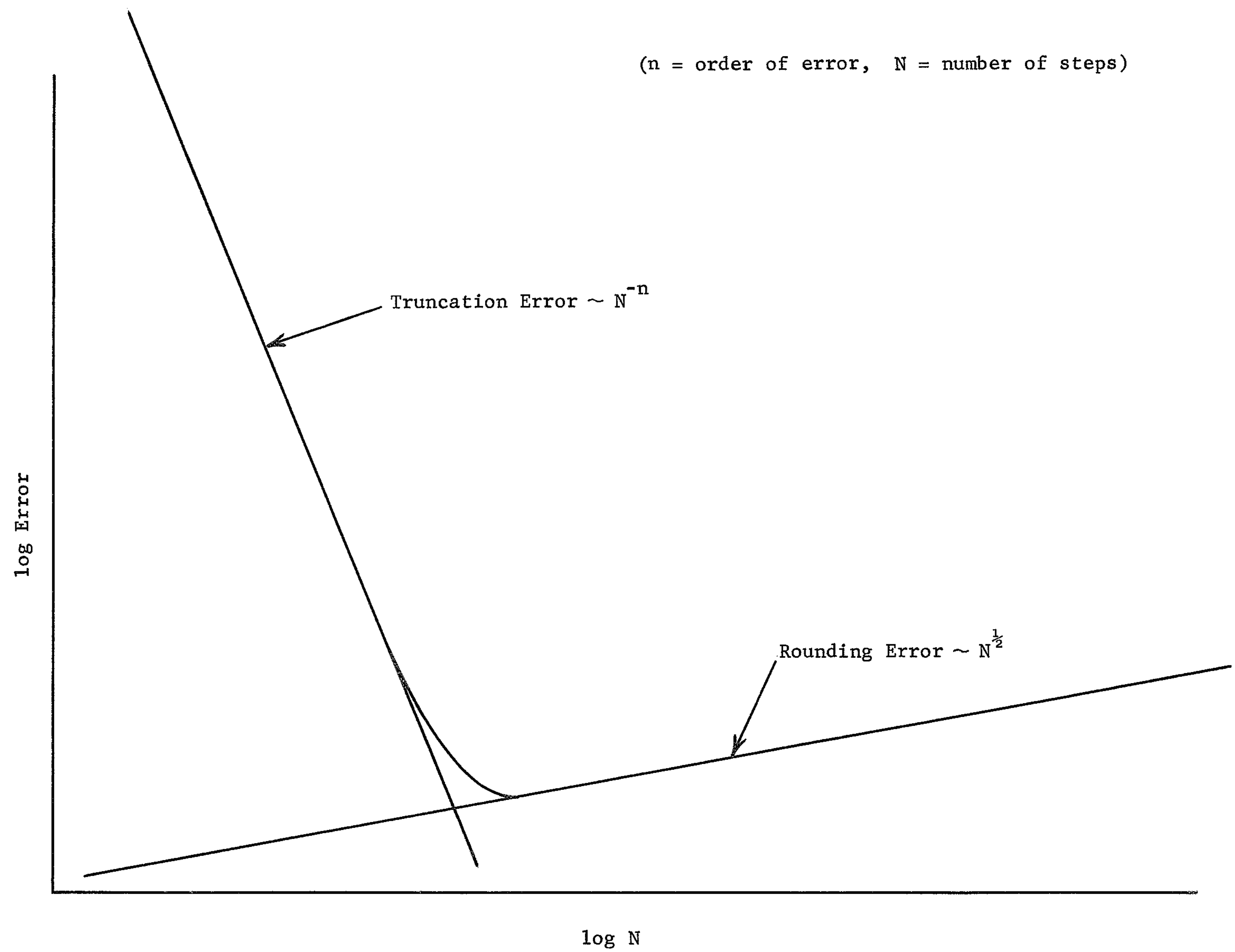

Figure 6. Expected Error Behavior in Numerical Integration of Differential Equations.

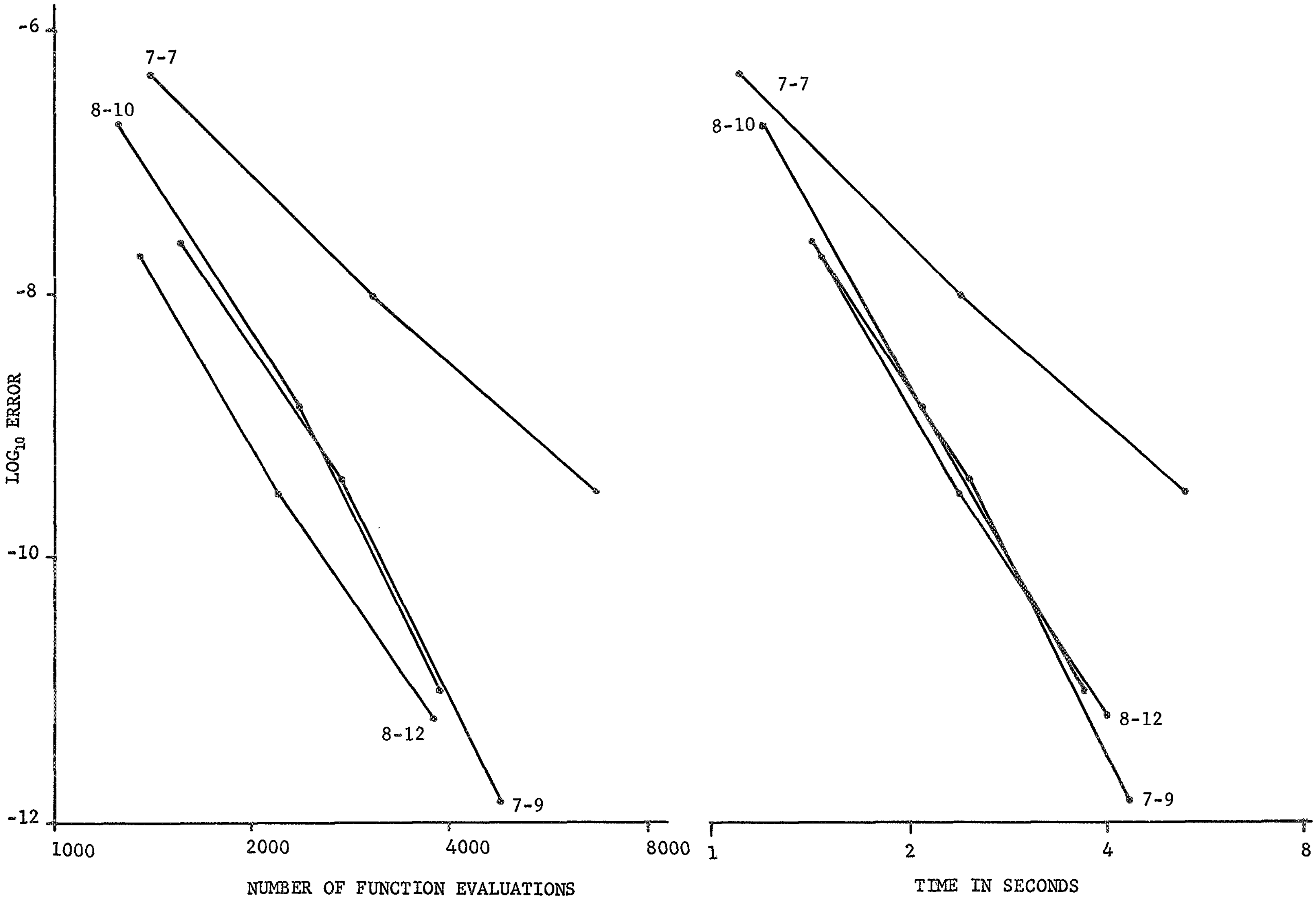

Figure 7. Error Behavior for the Runge-Kutta Formulas of Shanks.

Figure 8 shows results of runs made with Adams' method. Again the plots are of error vs. number of function evaluations and vs. time. Labels indicate order. Again this is the truncation error part of the curve, and one sees that the slopes are steeper for the higher orders.

Figure 9 shows results of runs made with Cowell's method. Both time and number of function evaluations are plotted vs. error, and the labels indicate orders. Here one sees that the region of rounding error domination is being approached or entered by the 13 order formula below errors of 10^{-12}.

Figure 10 shows runs made with Butcher's formulas. Only number of function evaluations (not time) vs. error is plotted; the labels indicate order. Here again we see that the region of rounding error is being approached at errors in the vicinity 10^{-12}.

Figure 11 is a composite plot of the four methods each of order 7. The slopes are seen to be all more or less the same. Butcher's method appears to be better than the others above 10^{-10} but enters the rounding region first.

Figure 12 is also a composite graph of different methods, all of order 13. Again Butcher's formula appears superior in the truncation region but enters the rounding region first.

Figure 13 gives a more detailed examination of Butcher's method (compare with Figure 10). Many more points were taken here, and we see considerable scatter when points are taken close together. This gives a better picture of how the rounding error region is entered by the 7th and 13th orders. The 9th and 11th orders do not exhibit rounding errors as large as those of the 7th and 13th orders.

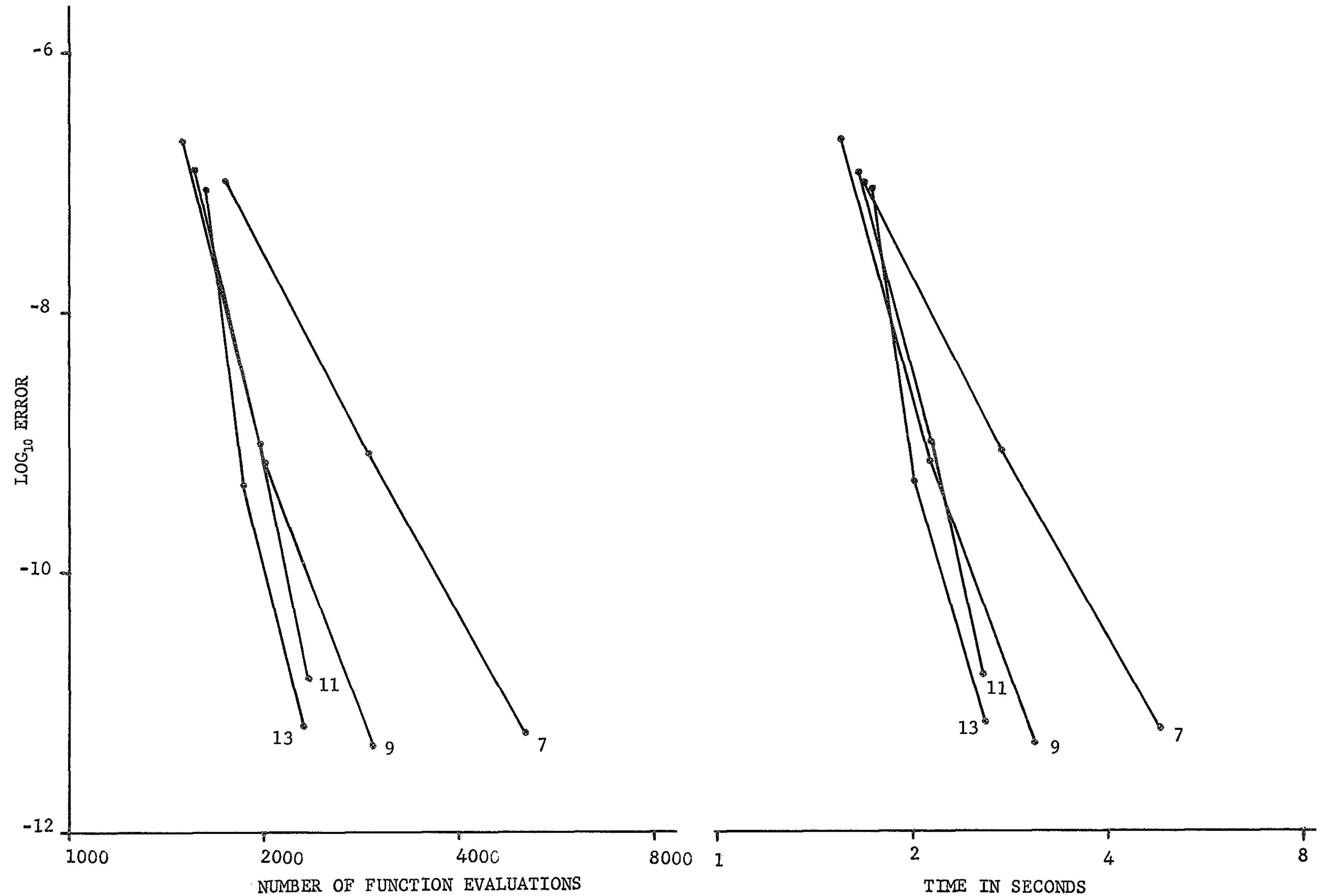

Figure 8. Error Behavior for Adams' Method.

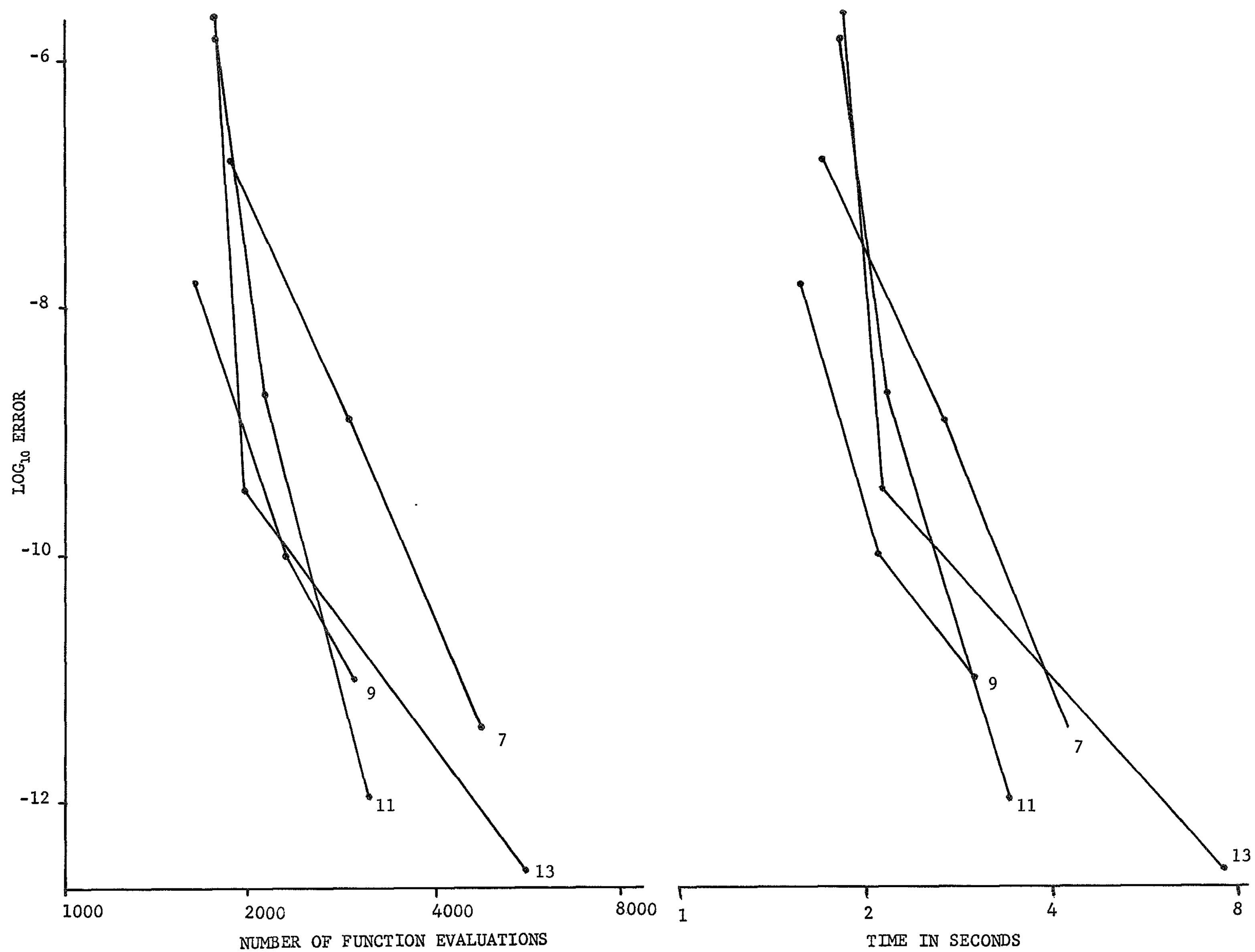

Figure 9. Error Behavior for Cowell's Method.

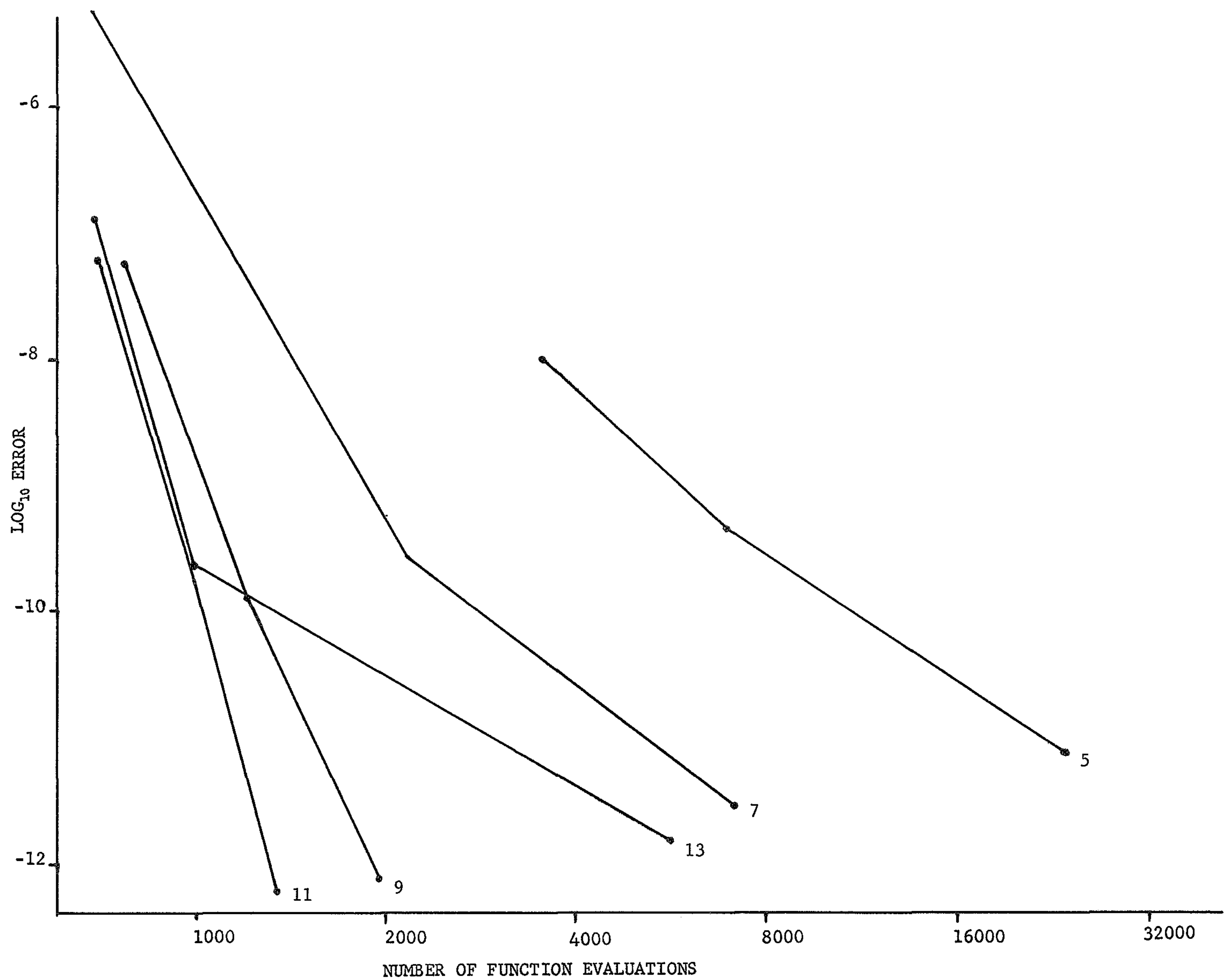

Figure 10. Error Behavior for Butcher's Formulas.

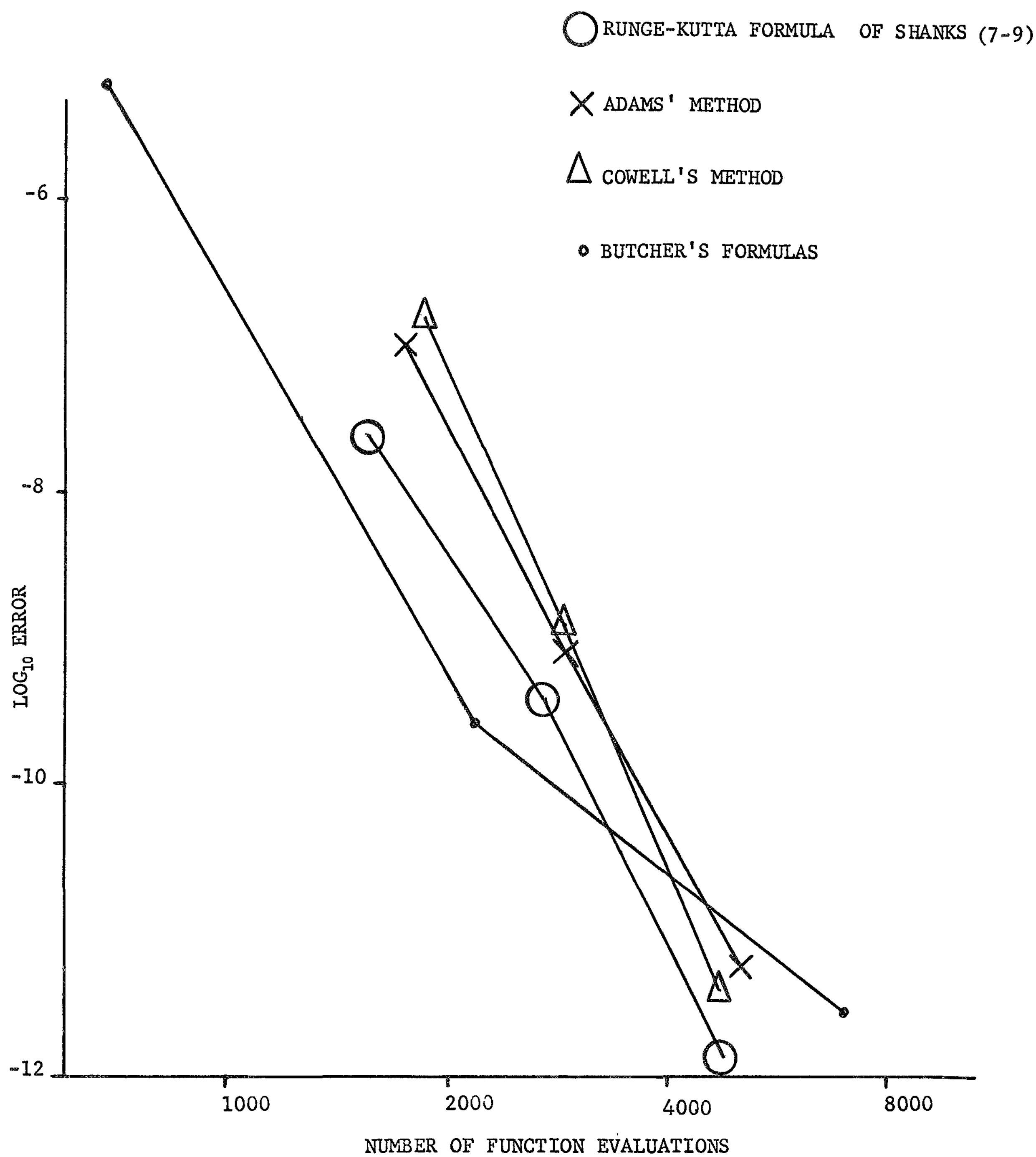

Figure 11. Error Behavior for the 7th Order Formulas.

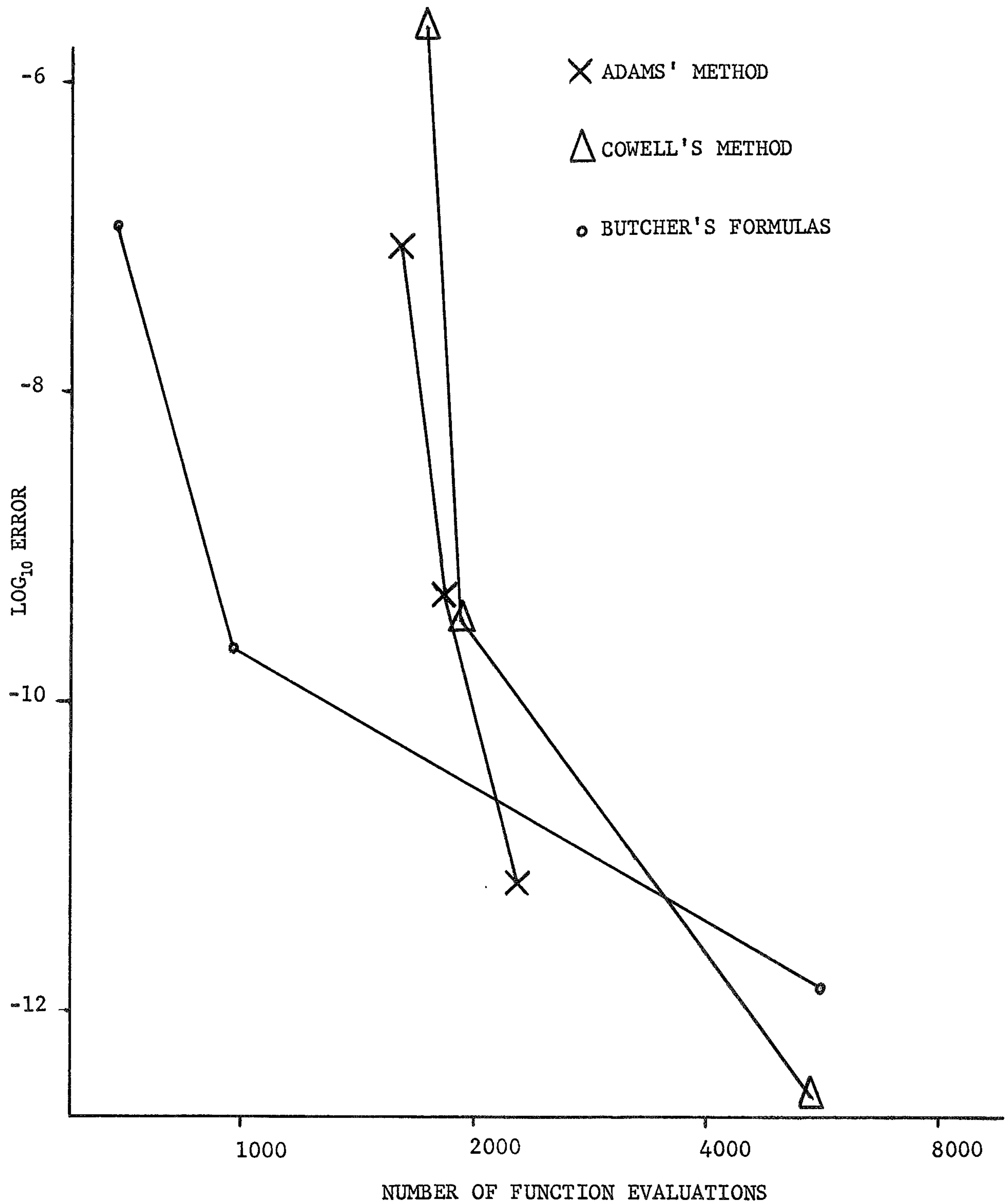

Figure 12. Error Behavior for the 13th Order Formulas.

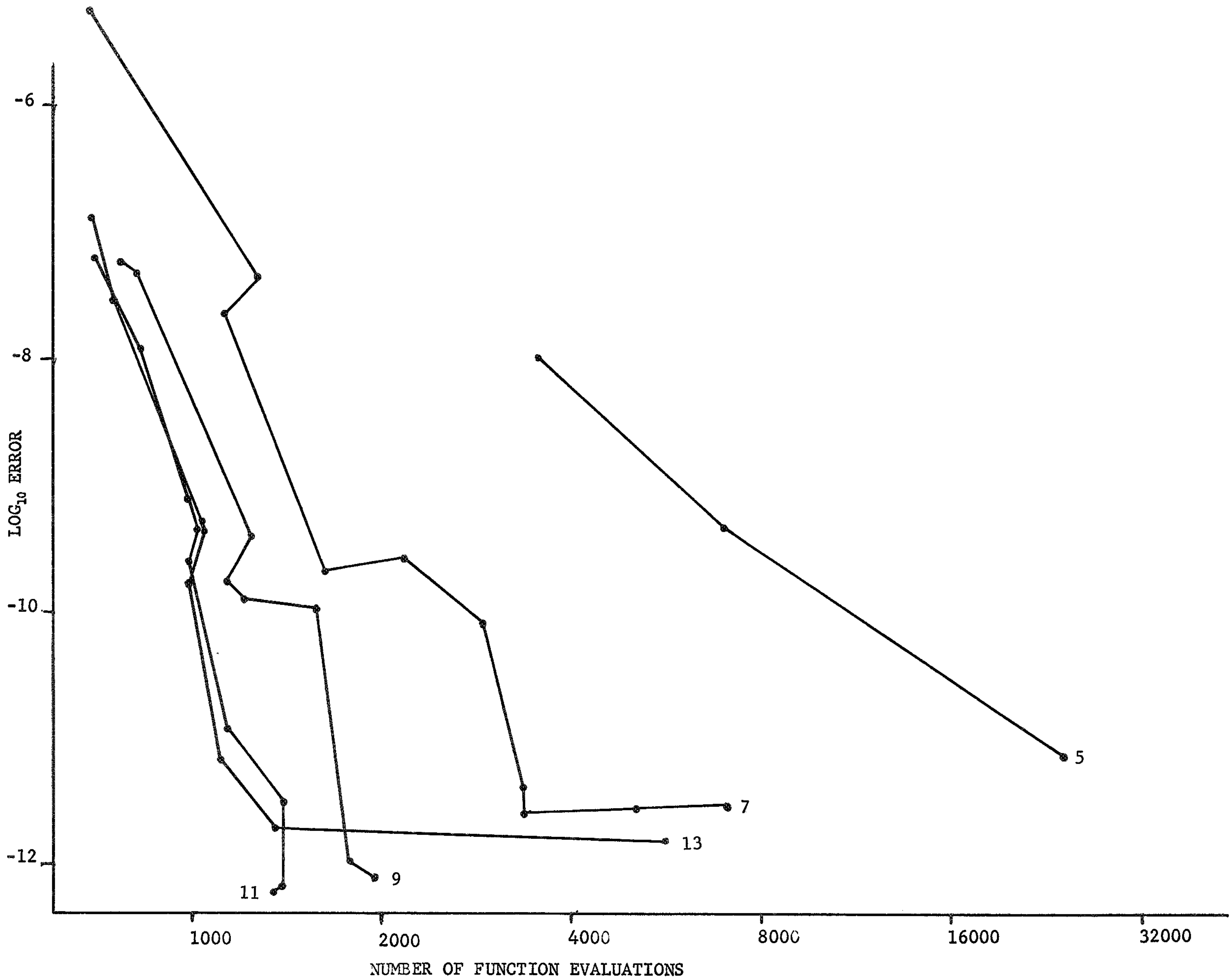

Figure 13. Detail Error Behavior for Butcher's Formulas.

C. Conclusions

From the above results one sees that all of the methods tested are fairly effective. At higher accuracies the higher order methods are more effective than the lower order methods, at least until the round off error region is reached. The region of round off error is determined primarily by the number of significant figures carried by the particular machine but is also somewhat dependent both on the method and order of the integration scheme.

For a fixed order (figure 11, 7th order; figure 12, 13th order) one sees that the Butcher formula is more effective than the others, at least until the rounding region is reached, but apparently Butcher enters the rounding error region sooner (produces larger rounding errors) than the others.

At 7th order the Shanks formula appears somewhat better than either the Cowell or Adams methods. No Shanks formulas of order higher than 8 were available at the time this work was being done, but the results here (figure 7) suggest that higher order Shanks formulas would be effective in reducing the number of function evaluations. Whether higher order Shanks formulas would take less time or not would depend on the application.

Whether number of function evaluations or computer time is more relevant in these tests depends on the application one has in mind. In the test problem used here the function evaluation time was relatively small compared to the rest of the arithmetic for any of the methods. In this case the time measurements show the additional time taken by the added complexity of the higher order methods. For example, Figure 7 shows that, while the number of function evaluations for the Shanks 8-12 formula is less than for the 7-9 or 8-10 formulas, the time taken for each is almost identical. Thus the 8-12 formula is seen to be more advantageous only if the function evaluations are quite complex and time consuming.

One might also question whether the results for the test problem are representative of the overall behavior of these methods. The error at the end of one complete revolution might not be representative of maximum error over the orbit since error cancelling is known to take place when integrating a periodic system. The absolute error, however, is not the relevant measure here but rather the error of one method relative to another. If error cancelling takes place because of the geometric properties of the orbit, this cancelling should be the same for each method and not affect the validity of comparing the relative performance of the methods. However, one cannot rule out entirely the possibility that other type problems might show a different relative effectiveness of these methods. For example it is known that special highly stable methods out-perform those tested here when applied to very "stiff" equations [27].

D. Suggestions for Further Study

The most obvious need for further study is an examination of the performance of the subroutines for other types of problems, in particular non-orbit type problems.

Other integration formulas should be tested. In particular Shanks has now made available (private communication) a 9-16 formula and a 10-21 formula, (i.e. a 9th order formula with 16 function evaluations per step, and a 10th order with 21 function evaluations per step.) The results here suggest that at high enough accuracies these new high order Shanks formulas might be advantageous.

Gear [27] has published some highly stable methods that are reputed to be especially good when used with very "stiff" equations. Comparisons of the formulas tested here using the stiff equations and comparisons with Gear's methods should be made.

Applications of the methods studied here to the solution of other type problems (non-initial value problems) could be made. In particular these methods can be used to generate the Green's functions needed to solve differential equation problems with split boundary conditions, i.e. rendezvous type orbit problems.

IV. BIBLIOGRAPHY

References on the Adams Method

1. Krogh, Fred T., J. ACM, 13, (1966) 374-385.
2. Bashforth, F., and Adams, J. C., Theories of Capillary Action, Cambridge University Press, 1883.
3. Dahlquist, Germund, Math. Scan., 4, (1956) 33.
4. Dahlquist, Germund, Stockholm Tekniska Hogskolan, No. 130, (1959), 1-87.
5. Henrici, Peter, Discrete Variable Methods in Ordinary Differential Equations, John Wiley and Sons, New York, 1962, 191-199, 200, 203-204, 224, 241, 258, 272-273, 274.
6. Newberry, A. C. R., Math. Comp. 17, 84, (1963) 452-455.

References on Cowell (Constant Nth Order Difference) Method

7. Cowell, P. H., and Crommelin, A. C. D., "Investigation of the Motion of Halley's Comet from 1759 to 1910." Appendix to Greenwich Observations for 1909, Edinburgh, 1910, 84.
8. Herrick, S., "Step-by Step Integration of $\dot{x} = f(x, y, z, t)$ without a Corrector," Mathematical Tables and Other Aids to Computation, No. 34, April 1951, 61-67.
9. Durham, H. L., Jr., et al, "Study of Methods for Numerical Solution of Ordinary Differential Equations," Final Report, Contract NAS8-11129, Project A-740, Engineering Experiment Station, Georgia Institute of Technology, Atlanta, Georgia, 1964.
10. Currie, J. C., et al, "Study of Satellite Orbit Computation Methods and An Error Analysis of the Mathematical Procedures," Technical Report No. 3, Prepared for Department of the Air Force Contract No. AF29(600)-1756, Project No. 1770, Headquarters Air Force Missile Development Center, Air Research and Development Command, Holloman Air Force Base, New Mexico, August 1959.
11. Francis, O. B., Jr., et al, "Study of the Methods for the Numerical Solution of Ordinary Differential Equations," Final Report, Project A-831, Contract NAS8-20014, Engineering Experiment Station, Georgia Institute of Technology, Atlanta, Georgia, 1966.

References on the Stetter-Gragg-Butcher Method

12. Gear, C. W., "Hybrid Methods for Initial Value Problems in Ordinary Differential Equations," J.SIAM (B) 2 (1965) 69.

13. Butcher, J. C., "A Modified Multistep Method for the Numerical Integration of Ordinary Differential Equations," J. ACM, 12 (1965) 124-135.

14. Gragg, W. B. and Stetter, H. J., "Generalized Multistep Predictor-Corrector Methods," J. ACM, 11 (1964) 188-209.

References on the Runge-Kutta-Shanks Method

15. Shanks, E. B., "Formulas for Obtaining Solutions of Differential Equations by Evaluations of Functions," Presented at the summer meeting of the American Mathematical Society in Boulder, Colorado, August 1963.

16. Shanks, E. B., "Performance Characteristics of Higher Order Approximations of Runge-Kutta Type," Internal Note, George C. Marshall Space Flight Center, Huntsville, Alabama, March 1963, (6 pages).

17. Shanks, E. B., "Formulas for Obtaining Solutions of Differential Equations by Evaluation of Functions," pp. 69-81, Effective Use of ADP in the NASA, National Aeronautics and Space Administration, Washington, D. C., August 1964.

18. Shanks, E. B., "Higher Order Approximations of Runge-Kutta Type," NASA Technical Note D-2920, National Aeronautics and Space Administration, Washington, D. C., September 1965, (23 pages).

19. Shanks, E. B., "Solutions of Differential Equations by Evaluations of Functions," Mathematics of Computation (published by the American Mathematical Society) Vol. 20, No. 93, 21-38. January 1966.

Other References

20. Ralston, A., "Runge-Kutta Methods with Minimum Error Bounds," Math. Comput., 16, (1962) 431-437.

21. Brush, D. G., Kohfeld, J. J., and Thompson, G. T., "Solution of Ordinary Differential Equations Using Two 'Off-Step' Points," J. ACM, 14, (1967) 84-89.

22. Scraton, R. E., "Estimation of the Truncation Error in Runge-Kutta and Allied Processes," The Computer Journal, 7 (1964) 246-248.

23. Butcher, J. C., "A Multistep Generalization of Runge-Kutta Methods with Four or Five Stages," J. ACM, 14 (1967) 84-89.

Other References (Cont.)

24. Kohfeld, J. J. and Thompson, G., T., "Multistep Methods With Modified Predictors and Correctors," J. ACM, 14 (1967) 155-166.

25. Gallaher, L. J., et al, "Study of the Methods for the Numerical Solution of Ordinary Differential Equations, " Final Report Project A-831, Contract NAS8-20014, Engineering Experiment Station, Georgia Institute of Technology, Atlanta, Georgia, 1967 and 1968.

26. Gear, C. W., "The Numerical Integration of Ordinary Differential Equations, " Math. Comp. 21, (1967), 146-156.

27. Gear, C. W., "The Automatic Integration of Stiff Ordinary Differential Equations," Proceedings of 1968 IFIPS Congress.

28. Nordsieck, A., "On Numerical Integration of Ordinary Differential Equations," Math. Comp. 16, (1962), 22-49.

29. Ratliff, K., "A Comparison of Techniques for Numerical Integration of Ordinary Differential Equations," Report No. 274, Dept. of Computer Science, University of Illinois, Urbana, Illinois, 1968.

30. Gear, C. W., "The Control of Parameters in Automatic Integration of Ordinary Differential Equations," Document C00-1469-0077, Dept. of Computer Science, University of Illinois, Urbana, Illinois, 1968.

31. Dill, C., Ellis, C. A., Gear, C. W., Ratliff, K., "The Automatic Integration Package for Ordinary Differential Equations," Document C00-1469-0103, Department of Computer Science, University of Illinois, Urbana, Illinois, 1968.

APPENDIX A

PROGRAM LISTINGS

In this appendix is listed each of the programs described in the body of the report. Also listed here are a skeleton main program and FUNCTI subroutine. The order of the listings is as follows:

1. A skeleton main program
2. The Runge-Kutta-Shanks subroutine SHANKS
3. The Adams subroutine ADAMS
4. The Stetter-Gragg-Butcher subroutine BUTCHR
5. The Cowell subroutine COWELL
6. The starting routine START
7. An auxiliary routine to START called RUNKUT
8. An auxiliary routine to START called COMP
9. A skeleton FUNCTI subroutine

```
GALLAHER-L-J*TPF$.A1167
        C      PROGRAM MAIN                                                          00001000
               DEFINE FDBLE(I) = DBLE(FLOAT(I))                                      00002000
        C      PARAMETER NP1
               COMMON/NN/NN
               COMMON/COMMON/FA(35,NP1)
               DOUBLE PRECISION FA,X, Y1, Y2, XI1
               DIMENSION Y(100),EA(100),ER(100),ADMSCF(91),RKSCFF(91)
              X,Y1(100), Y2(100)
               DIMENSION BTCHCF(91), CWLLCF(91)
               DOUBLE PRECISION EA,ER,Y,ADMSCF,RKSCFF,BTCHCF,CWLLCF
               INTEGER A,B,C,Q,O,F
               END                                                                   00003000
        C      BLOCK  2                                                              00004000


@PRT      .SHANKS
```

```
GALLAHER-L-J*TPF$.SHANKS
      SUBROUTINE SHANKS(N, XIV,XF,YV,IM,ORDER,CF,P,EA,ER,DXV)
      DOUBLE PRECISION XIV, DXV
      INTEGER N,M,ORDER                                                 01542000
      DOUBLE PRECISION XI,XF,P,DX                                       01543000
      DOUBLE PRECISION CF                                               01544000
      DIMENSION CF(91)                                                  01545000
      DOUBLE PRECISION YV,EA,ER                                         01546000
      DIMENSION YV(100),EA(100),ER(100)                                 01547000
      INTEGER I,J,K,L,COUNT,COUNT2,II,NCF                               01548000
      INTEGER NCF1                                                      01549000
      INTEGER DKTR                                                      01550000
      DOUBLE PRECISION EFACT                                            01551000
      DOUBLE PRECISION BETA,DCOUNT,DXD,DXH,DXT,EFACTR,ERANGE,ES,GAMMA,X,01552000
     1XM                                                                01553000
      LOGICAL CFSW,DSW                                                  01554000
      DOUBLE PRECISION CFD                                              01555000
      DIMENSION CFD(180)
      DOUBLE PRECISION FV
      COMMON/COMMON/FV(35,1)
      DOUBLE PRECISION TEMP9,GV,YC,YM,YP                                01559000
      DIMENSION TEMP9(100),GV(100),YC(100),YM(100),YP(100)              01560000
      INTEGER II1                                                       01561000
      INTEGER STEPR,STEPS                                               01562000
      DEFINE FDBLE(I) = DBLE(FLOAT(I))                                  01563000
      M= IM
      XI = XIV
      DX = DXV
      M = M - 1                                                         01564000
      STEPR = 0                                                         01565000
      STEPS = STEPR                                                     01566000
      DXT = XF - XI                                                     01567000
      DXD = DXT                                                         01568000
      IF (DXT .EQ. 0) GO TO 625                                         01569000
      IF (DX .NE. 0) GO TO 628                                          01570000
      DX = DXD                                                          01571000
  628 CONTINUE                                                          01572000
  627 COUNT = 1                                                         01573000
  629 IF (ABS(DX) .GE. ABS(DXD)) GO TO 630                              01574000
      COUNT = COUNT + COUNT                                             01575000
```

```
      DXD = DXT / FDBLE(COUNT)
      GO TO 629
  630 COUNT2 = COUNT + COUNT
      DXH = DXT / FDBLE(COUNT2)
      DCOUNT = COUNT
      EFACT = 1
      I = 1
      GO TO 633
  634 I = I + 1
  633 IF (I .LE. ORDER) GO TO 631
      GO TO 632
  631 EFACT = EFACT + EFACT
      GO TO 634
  632 ERANGE = .125 / EFACT
      EFACT = 4.0D0 / EFACT
      EFACTR = (COUNT ** P) * EFACT
      DKTR = 0
      NCF1 = ((((M * M) + M) / 2) + M) + M
      NCF = NCF1 + 1
      CFSW = .FALSE.
      I = 0
      GO TO 637
  638 I = I + 1
  637 IF (I .LE. NCF1) GO TO 635
      GO TO 636
  635 CFD(I + 1) = CF(I + 1) * DXD
      III2 = I + NCF
      CFD(III2 + 1) = CF(I + 1) * DXH
      GO TO 638
  636 X = XI
      XM = XI + DXH
  623 DSW = .TRUE.
      CALL FUNCTI(N, X, YV, GV)
      IF ( .NOT. CFSW) GO TO 640
      L = NCF1
      GO TO 639
  640 L =  - 1
  639 I = 1
      GO TO 643
  644 I = I + 1
```

```
  643 IF (I .LE. M) GO TO 641                                    01616000
      GO TO 642                                                  01617000
  641 II1 = I - 1                                                01618000
      L = L + 1                                                  01619000
      BETA = CFD(L + 1)                                          01620000
      K = 1                                                      01621000
      GO TO 647                                                  01622000
  648 K = K + 1                                                  01623000
  647 IF (K .LE. N) GO TO 645                                    01624000
      GO TO 646                                                  01625000
  645 YP(K + 1) = (GV(K + 1) * BETA) + YV(K + 1)                 01626000
      GO TO 648                                                  01627000
  646 J = 1                                                      01628000
      GO TO 651                                                  01629000
  652 J = J + 1                                                  01630000
  651 IF (J .LE. II1) GO TO 649                                  01631000
      GO TO 650                                                  01632000
  649 L = L + 1                                                  01633000
      BETA = CFD(L + 1)                                          01634000
      K = 1                                                      01635000
      GO TO 655                                                  01636000
  656 K = K + 1                                                  01637000
  655 IF (K .LE. N) GO TO 653                                    01638000
      GO TO 654                                                  01639000
  653 YP(K + 1) = (FV(J + 1, K + 1) * BETA) + YP(K + 1)          01640000
      GO TO 656                                                  01641000
  654 GO TO 652                                                  01642000
  650 L = L + 1                                                  01643000
      CALL FUNCTI(N, CFD(L + 1) + X, YP, TEMP9)                  01644000
      II1 = 0                                                    01645000
      GO TO 659                                                  01646000
  660 II1 = II1 + 1                                              01647000
  659 IF (II1 .LE. N) GO TO 657                                  01648000
      GO TO 658                                                  01649000
  657 FV(I + 1, II1 + 1) = TEMP9(II1 + 1)                        01650000
      GO TO 660                                                  01651000
  658 GO TO 644                                                  01652000
  642 L = L + 1                                                  01653000
      GAMMA = CFD(L + 1)                                         01654000
      K = 1                                                      01655000
```

```
      GO TO 663                                                         01656000
  664 K = K + 1                                                         01657000
  663 IF (K .LE. N) GO TO 661                                           01658000
      GO TO 662                                                         01659000
  661 YP(K + 1) = (GV(K + 1) * GAMMA) + YV(K + 1)                       01660000
      GO TO 664                                                         01661000
  662 I = 1                                                             01662000
      GO TO 667                                                         01663000
  668 I = I + 1                                                         01664000
  667 IF (I .LE. M) GO TO 665                                           01665000
      GO TO 666                                                         01666000
  665 L = L + 1                                                         01667000
      GAMMA = CFD(L + 1)                                                01668000
      K = 1                                                             01669000
      GO TO 671                                                         01670000
  672 K = K + 1                                                         01671000
  671 IF (K .LE. N) GO TO 669                                           01672000
      GO TO 670                                                         01673000
  669 YP(K + 1) = (FV(I + 1, K + 1) * GAMMA) + YP(K + 1)                01674000
      GO TO 672                                                         01675000
  670 GO TO 668                                                         01676000
  666 CONTINUE                                                          01677000
  624 IF ( .NOT. CFSW) GO TO 674                                        01678000
      L =  - 1                                                          01679000
  674 CONTINUE                                                          01680000
  673 I = 1                                                             01681000
      GO TO 677                                                         01682000
  678 I = I + 1                                                         01683000
  677 IF (I .LE. M) GO TO 675                                           01684000
      GO TO 676                                                         01685000
  675 II1 = I - 1                                                       01686000
      L = L + 1                                                         01687000
      BETA = CFD(L + 1)                                                 01688000
      K = 1                                                             01689000
      GO TO 681                                                         01690000
  682 K = K + 1                                                         01691000
  681 IF (K .LE. N) GO TO 679                                           01692000
      GO TO 680                                                         01693000
  679 YM(K + 1) = (GV(K + 1) * BETA) + YV(K + 1)                        01694000
      GO TO 682                                                         01695000
```

```
  680 J = 1
      GO TO 685
  686 J = J + 1
  685 IF (J .LE. II1) GO TO 683
      GO TO 684
  683 L = L + 1
      BETA = CFD(L + 1)
      K = 1
      GO TO 689
  690 K = K + 1
  689 IF (K .LE. N) GO TO 687
      GO TO 688
  687 YM(K + 1) = (FV(J + 1, K + 1) * BETA) + YM(K + 1)
      GO TO 690
  688 GO TO 686
  684 L = L + 1
      CALL FUNCTI(N, CFD(L + 1) + X, YM, TEMP9)
      II1 = 0
      GO TO 693
  694 II1 = II1 + 1
  693 IF (II1 .LE. N) GO TO 691
      GO TO 692
  691 FV(I + 1, II1 + 1) = TEMP9(II1 + 1)
      GO TO 694
  692 GO TO 678
  676 L = L + 1
      GAMMA = CFD(L + 1)
      K = 1
      GO TO 697
  698 K = K + 1
  697 IF (K .LE. N) GO TO 695
      GO TO 696
  695 YM(K + 1) = (GV(K + 1) * GAMMA) + YV(K + 1)
      GO TO 698
  696 I = 1
      GO TO 701
  702 I = I + 1
  701 IF (I .LE. M) GO TO 699
      GO TO 700
  699 L = L + 1
```

```
      GAMMA = CFD(L + 1)
      K = 1
      GO TO 705
  706 K = K + 1
  705 IF (K .LE. N) GO TO 703
      GO TO 704
  703 YM(K + 1) = (FV(I + 1, K + 1) * GAMMA) + YM(K + 1)
      GO TO 706
  704 GO TO 702
  700 CALL FUNCTI(N, XM, YM, TEMP9)
      II1 = 0
      GO TO 709
  710 II1 = II1 + 1
  709 IF (II1 .LE. N) GO TO 707
      GO TO 708
  707 FV(1, II1 + 1) = TEMP9(II1 + 1)
      GO TO 710
  708 IF ( .NOT. CFSW) GO TO 712
      L =  - 1
      GO TO 711
  712 L = NCF1
  711 I = 1
      GO TO 715
  716 I = I + 1
  715 IF (I .LE. M) GO TO 713
      GO TO 714
  713 II1 = I - 1
      K = 1
      GO TO 719
  720 K = K + 1
  719 IF (K .LE. N) GO TO 717
      GO TO 718
  717 YC(K + 1) = YM(K + 1)
      GO TO 720
  718 J = 0
      GO TO 723
  724 J = J + 1
  723 IF (J .LE. II1) GO TO 721
      GO TO 722
  721 L = L + 1
```

```
      BETA = CFD(L + 1)                                          01776000
      K = 1                                                      01777000
      GO TO 727                                                  01778000
  728 K = K + 1                                                  01779000
  727 IF (K .LE. N) GO TO 725                                    01780000
      GO TO 726                                                  01781000
  725 YC(K + 1) = (FV(J + 1, K + 1) * BETA) + YC(K + 1)          01782000
      GO TO 728                                                  01783000
  726 GO TO 724                                                  01784000
  722 L = L + 1                                                  01785000
      CALL FUNCTI(N, CFD(L + 1) + XM, YC, TEMP9)                 01786000
      II1 = 0                                                    01787000
      GO TO 731                                                  01788000
  732 II1 = II1 + 1                                              01789000
  731 IF (II1 .LE. N) GO TO 729                                  01790000
      GO TO 730                                                  01791000
  729 FV(I + 1, II1 + 1) = TEMP9(II1 + 1)                        01792000
      GO TO 732                                                  01793000
  730 GO TO 716                                                  01794000
  714 K = 1                                                      01795000
      GO TO 735                                                  01796000
  736 K = K + 1                                                  01797000
  735 IF (K .LE. N) GO TO 733                                    01798000
      GO TO 734                                                  01799000
  733 YC(K + 1) = YM(K + 1)                                      01800000
      GO TO 736                                                  01801000
  734 I = 0                                                      01802000
      GO TO 739                                                  01803000
  740 I = I + 1                                                  01804000
  739 IF (I .LE. M) GO TO 737                                    01805000
      GO TO 738                                                  01806000
  737 L = L + 1                                                  01807000
      GAMMA = CFD(L + 1)                                         01808000
      K = 1                                                      01809000
      GO TO 743                                                  01810000
  744 K = K + 1                                                  01811000
  743 IF (K .LE. N) GO TO 741                                    01812000
      GO TO 742                                                  01813000
  741 YC(K + 1) = (FV(I + 1, K + 1) * GAMMA) + YC(K + 1)         01814000
      GO TO 744                                                  01815000
```

```
742 GO TO 740
738 K = 1
    GO TO 747
748 K = K + 1
747 IF (K .LE. N) GO TO 745
    GO TO 746
745 ES = ABS(YC(K + 1) - YP(K + 1)) * EFACTR
    IF (ES .EQ. 0) GO TO 750
    IF (ES .LT. EA(K + 1)) GO TO 752
    IF (ES .LT. ABS(YC(K + 1)) * ER(K + 1)) GO TO 753
    DSW = .FALSE.
    STEPR = STEPR + 1
    COUNT = COUNT2
    COUNT2 = COUNT + COUNT
    DCOUNT = DCOUNT + DCOUNT
    DXD = DXH
    DXH = DXT / FDBLE(COUNT2)
    EFACTR = (COUNT ** P) * EFACT
    IF ( .NOT. CFSW) GO TO 755
    I = 0
    GO TO 758
759 I = I + 1
758 IF (I .LE. NCF1) GO TO 756
    GO TO 757
756 III3 = I + NCF
    CFD(III3 + 1) = CF(I + 1) * DXH
    GO TO 759
757 CFSW = .FALSE.
    GO TO 754
755 I = 0
    GO TO 762
763 I = I + 1
762 IF (I .LE. NCF1) GO TO 760
    GO TO 761
760 CFD(I + 1) = CF(I + 1) * DXH
    GO TO 763
761 CFSW = .TRUE.
754 XM = ((((FDBLE(COUNT2 + 1)) - DCOUNT) - DCOUNT) * DXH) + XI
    K = 1
    GO TO 766
```

```
      767 K = K + 1                                                     01856000
      766 IF (K .LE. N) GO TO 764                                       01857000
          GO TO 765                                                     01858000
      764 YP(K + 1) = YM(K + 1)                                         01859000
          GO TO 767                                                     01860000
      765 GO TO 624                                                     01861000
      753 CONTINUE                                                      01862000
      752 CONTINUE                                                      01863000
      751 IF ( .NOT. DSW) GO TO 769                                     01864000
          IF (ES .LT. EA(K + 1) * ERANGE) GO TO 770                     01865000
          IF (ES .LT. (ABS(YC(K + 1)) * ER(K + 1)) * ERANGE) GO TO 771  01866000
          DSW = .FALSE.                                                 01867000
      771 CONTINUE                                                      01868000
      770 CONTINUE                                                      01869000
      769 CONTINUE                                                      01870000
      768 CONTINUE                                                      01871000
      750 CONTINUE                                                      01872000
      749 GO TO 748                                                     01873000
      746 DCOUNT = DCOUNT - 1.0D0                                       01874000
          STEPS = STEPS + 1                                             01875000
          K = 1                                                         01876000
          GO TO 774                                                     01877000
      775 K = K + 1                                                     01878000
      774 IF (K .LE. N) GO TO 772                                       01879000
          GO TO 773                                                     01880000
      772 YV(K + 1) = YC(K + 1)                                         01881000
          GO TO 775                                                     01882000
      773 IF (DCOUNT .EQ. 0) GO TO 625                                  01883000
          X = ((FDBLE(COUNT) - DCOUNT) * DXD) + XI                      01884000
          IF (DCOUNT .GE. 2) GO TO 778                                  01885000
          IF (DCOUNT .NE. 1) GO TO 780                                  01886000
          DSW = .FALSE.                                                 01887000
      780 CONTINUE                                                      01888000
      779 IF ((DCOUNT .GE. 1) .AND.  .NOT. DSW) GO TO 782               01889000
          IF (DCOUNT .LE. 1) GO TO 784                                  01890000
          DKTR = DKTR + 1                                               01891000
      784 CONTINUE                                                      01892000
      783 DCOUNT = 1                                                    01893000
          COUNT = DCOUNT + .5                                           01894000
          COUNT2 = 2                                                    01895000
```

```
      EFACTR = EFACT
      XI = X
      DXT = XF - XI
      DXD = DXT
      DXH = DXD / 2.0D0
      XM = XI + DXH
      CFSW = .FALSE.
      I = 0
      GO TO 787
  788 I = I + 1
  787 IF (I .LE. NCF1) GO TO 785
      GO TO 786
  785 CFD(I + 1) = CF(I + 1) * DXD
      III2 = I + NCF
      CFD(III2 + 1) = CF(I + 1) * DXH
      GO TO 788
  786 GO TO 623
  782 CONTINUE
  781 CONTINUE
  778 CONTINUE
  777 IF ( .NOT. DSW) GO TO 790
      DKTR = DKTR + 1
      COUNT2 = COUNT
      COUNT = COUNT / 2
      DCOUNT = DCOUNT / 2.0D0
      DXH = DXD
      DXD = DXT / FDBLE(COUNT)
      EFACTR = (COUNT ** P) * EFACT
      IF ( .NOT. CFSW) GO TO 792
      I = 0
      GO TO 795
  796 I = I + 1
  795 IF (I .LE. NCF1) GO TO 793
      GO TO 794
  793 CFD(I + 1) = CF(I + 1) * DXD
      GO TO 796
  794 CFSW = .FALSE.
      GO TO 791
  792 I = 0
      GO TO 799
```

```
800 I = I + 1
799 IF (I .LE. NCF1) GO TO 797
    GO TO 798
797 III2 = I + NCF
    CFD(III2 + 1) = CF(I + 1) * DXD
    GO TO 800
798 CFSW = .TRUE.
791 CONTINUE
790 CONTINUE
789 XM = ((((FDBLE(COUNT2 + 1)) - DCOUNT) - DCOUNT) * DXH) + XI
    GO TO 623
625 CONTINUE
    RETURN
    END
```

@PRT .ADAMS

```
GALLAHER-L-J*TPF$.ADAMS
      SUBROUTINE ADAMS(N, XI, XF, Y, P, Q, DXV, EA, ER, ADMSCF, RKSFNS,
     1RXSRDR, RKSCFF)
      DOUBLE PRECISION DXV
      COMMON/COMMON/FH(35,1)
      DOUBLE PRECISION XI,XF,P,DX                                       00488000
      INTEGER N,Q,RKSFNS,RKSRDR                                         00489000
      DOUBLE PRECISION Y,EA,ER                                          00490000
      DIMENSION Y(100),EA(100),ER(100)                                  00491000
      DOUBLE PRECISION ADMSCF                                           00492000
      DIMENSION ADMSCF(39)                                              00493000
      DOUBLE PRECISION RKSCFF                                           00494000
      DIMENSION RKSCFF(91)                                              00495000
      INTEGER II                                                        00496000
      DOUBLE PRECISION FH                                               00497000
      DOUBLE PRECISION B,BS,HB,HBS                                      00499000
      DIMENSION B(16),BS(16),HB(16),HBS(16)                             00500000
      DOUBLE PRECISION TEMP9,C,YP,YC,YD,FP,FC,EAU,EAL,ERU,ERL,HAL,HRL   00501000
      DIMENSION TEMP9(100),C(100),YP(100),YC(100),YD(100),FP(100),FC(10000502000
     1),EAU(100),EAL(100),ERU(100),ERL(100),HAL(100),HRL(100)           00503000
      DOUBLE PRECISION H,X,CU,C2,GR,YCI,BSQZ,FHJI,FMUI,HBMU,M1UP,HBSMU, 00504000
     1HBSQZ,ERROR,CHANGE,C2MQP5,INTRVL                                  00505000
      INTEGER I,J,K,C1,DC,PC,MU,MULT,JZERO,QT2M1,QMINS1,QTIMS2          00506000
      LOGICAL BGOOD,FLIPPD,TOSMLL                                       00507000
      DEFINE FDBLE(I) = DBLE(FLOAT(I))                                  00508000
      DX = DXV
      C2MQP5 = 1.0D0 / (FDBLE(2 ** (Q + 5)))                            00509000
      QMINS1 = Q - 1                                                    00510000
      QTIMS2 = Q + Q                                                    00511000
      QT2M1 = QTIMS2 - 1                                                00512000
      MU = 0                                                            00513000
      GO TO 204                                                         00514000
  205 MU = MU + 1                                                       00515000
  204 IF (MU .LE. QMINS1) GO TO 202                                     00516000
      GO TO 203                                                         00517000
  202 B(MU + 1) = ADMSCF(MU + 1)                                        00518000
      III2 = MU + Q                                                     00519000
      BS(MU + 1) = ADMSCF(III2 + 1)                                     00520000
      GO TO 205                                                         00521000
  203 BSQZ = ADMSCF(QTIMS2 + 1)                                         00522000
```

```
      GR = ADMSCF(QTIMS2 + 2)                                         00523000
      C1 = 1                                                          00524000
      INTRVL = XF - XI                                                00525000
      H = INTRVL                                                      00526000
      DX = ABS(DX)                                                    00527000
  206 IF ((ABS(H) .LE. DX) .AND. C1 .GE. Q) GO TO 207                 00528000
      C1 = C1 + C1                                                    00529000
      H = INTRVL / FDBLE(C1)                                          00530000
      GO TO 206                                                       00531000
 207  C2 = FDBLE(C1)
      J = 0                                                           00533000
      JZERO = J                                                       00534000
      CALL FUNCTI(N, XI, Y, TEMP9)                                    00535000
      II = 0                                                          00536000
      GO TO 210                                                       00537000
  211 II = II + 1                                                     00538000
  210 IF (II .LE. N) GO TO 208                                        00539000
      GO TO 209                                                       00540000
  208 FH(1, II + 1) = TEMP9(II + 1)                                   00541000
      GO TO 211                                                       00542000
  209 X = XI                                                          00543000
      I = 1                                                           00544000
      GO TO 214                                                       00545000
  215 I = I + 1                                                       00546000
  214 IF (I .LE. N) GO TO 212                                         00547000
      GO TO 213                                                       00548000
  212 Y(I + 1) = Y(I + 1)                                             00549000
      GO TO 215                                                       00550000
  213 BGOOD = .TRUE.                                                  00551000
  194 IF (BGOOD) GO TO 217                                            00552000
      I = 1                                                           00553000
      GO TO 220                                                       00554000
  221 I = I + 1                                                       00555000
  220 IF (I .LE. N) GO TO 218                                         00556000
      GO TO 219                                                       00557000
  218 Y(I + 1) = YD(I + 1)                                            00558000
      GO TO 221                                                       00559000
  219 CONTINUE                                                        00560000
  217 CONTINUE                                                        00561000
  216 X = XF - (INTRVL * (C2 / FDBLE(C1)))                            00562000
```

```
      MULT = START(N, XI, XF, C1, EA, ER, QMINS1, X, Y, FH, FH, Y, J,
     1QT2M1, 0, P, RKSFNS, RKSCFF)
      I = 1
      GO TO 224
  225 I = I + 1
  224 IF (I .LE. N) GO TO 222
      GO TO 223
  222 YP(I + 1) = Y(I + 1)
      C(I + 1) = YP(I + 1)
      GO TO 225
  223 BGOOD = .TRUE.
      C1 = C1 * MULT
      C2 = (C2 * FDBLE(MULT)) - FDBLE(Q)
      J = JZERO - 1
      IF (J .GE. 0) GO TO 227
      J = J + QT2M1
  227 CONTINUE
  226 DC = 0
      PC = Q
      CU = (C1 ** ( - P)) * GR
      H = INTRVL / FDBLE(C1)
      MU = 0
      GO TO 230
  231 MU = MU + 1
  230 IF (MU .LE. QMINS1) GO TO 228
      GO TO 229
  228 HB(MU + 1) = B(MU + 1) * H
      HBS(MU + 1) = BS(MU + 1) * H
      GO TO 231
  229 HBSQZ = BSQZ * H
      I = 1
      GO TO 234
  235 I = I + 1
  234 IF (I .LE. N) GO TO 232
      GO TO 233
  232 EAU(I + 1) = EA(I + 1) * CU
      EAL(I + 1) = EAU(I + 1) * C2MQP5
      ERU(I + 1) = ER(I + 1) * CU
      ERL(I + 1) = ERU(I + 1) * C2MQP5
      HAL(I + 1) = ABS(EAL(I + 1) / HBSQZ)
```

```
      HRL(I + 1) = ABS(ERL(I + 1) / HBSQZ)
      GO TO 235
  233 CONTINUE
  195 X = XF - (C2 * H)
      MU = 0
      GO TO 238
  239 MU = MU + 1
  238 IF (MU .LE. QMINS1) GO TO 236
      GO TO 237
  236 J = J + 1
      IF (J .NE. QT2M1) GO TO 241
      J = 0
  241 CONTINUE
  240 HBMU = HB(MU + 1)
      HBSMU = HBS(MU + 1)
      I = 1
      GO TO 244
  245 I = I + 1
  244 IF (I .LE. N) GO TO 242
      GO TO 243
  242 FMUI = FH(J + 1, I + 1)
      YP(I + 1) = (FMUI * HBMU) + YP(I + 1)
      C(I + 1) = (HBSMU * FMUI) + C(I + 1)
      GO TO 245
  243 GO TO 239
  237 CALL FUNCTI(N, X, YP, FP)
  196 I = 1
      GO TO 248
  249 I = I + 1
  248 IF (I .LE. N) GO TO 246
      GO TO 247
  246 YC(I + 1) = (FP(I + 1) * HBSQZ) + C(I + 1)
      GO TO 249
  247 CALL FUNCTI(N, X, YC, FC)
      I = 1
      GO TO 252
  253 I = I + 1
  252 IF (I .LE. N) GO TO 250
      GO TO 251
  250 CHANGE = ABS(FC(I + 1) - FP(I + 1))
```

```
      IF (CHANGE .LE. HAL(I + 1)) GO TO 255
      IF (CHANGE .GT. ABS(HRL(I + 1) * FC(I + 1))) GO TO 197
  255 CONTINUE
  254 GO TO 253
  251 FLIPPD = .TRUE.
      GO TO 198
  197 I = 1
      GO TO 258
  259 I = I + 1
  258 IF (I .LE. N) GO TO 256
      GO TO 257
  256 YC(I + 1) = (FC(I + 1) * HBSQZ) + C(I + 1)
      GO TO 259
  257 CALL FUNCTI(N, X, YC, FP)
      I = 1
      GO TO 262
  263 I = I + 1
  262 IF (I .LE. N) GO TO 260
      GO TO 261
  260 CHANGE = ABS(FP(I + 1) - FC(I + 1))
      IF (CHANGE .LE. HAL(I + 1)) GO TO 265
      IF (CHANGE .GT. ABS(HRL(I + 1) * FP(I + 1))) GO TO 196
  265 CONTINUE
  264 GO TO 263
  261 FLIPPD = .FALSE.
  198 J = J + 1
      IF (J .NE. QT2M1) GO TO 267
      J = 0
  267 CONTINUE
  266 TOSMLL = .TRUE.
      I = 1
      GO TO 270
  271 I = I + 1
  270 IF (I .LE. N) GO TO 268
      GO TO 269
  268 IF ( .NOT. FLIPPD) GO TO 273
      FH(J + 1, I + 1) = FC(I + 1)
      GO TO 272
  273 FH(J + 1, I + 1) = FP(I + 1)
  272 FHJI = FH(J + 1, I + 1)
```

```
      YCI= FHJI * HBSQZ + C(1+I)
      ERROR = ABS(YP(I+1) - YCI)
      C (I+1) = YCI
      YP(I+1) = YCI
      IF ( .NOT. BGOOD) GO TO 275
      YD(I + 1) = YCI
      GO TO 274
  275 Y(I + 1) = YCI
  274 YCI = ABS(YCI)
      IF (ERROR .LE. EAU(I + 1)) GO TO 277
      IF (ERROR .GT. ERU(I + 1) * YCI) GO TO 200
  277 CONTINUE
  276 IF (ERROR .LE. EAL(I + 1)) GO TO 279
      IF (ERROR .LE. ERL(I + 1) * YCI) GO TO 280
      TOSMLL = .FALSE.
  280 CONTINUE
  279 CONTINUE
  278 GO TO 271
  269 PC = PC + 1
      IF ( .NOT. BGOOD) GO TO 282
      BGOOD = .FALSE.
      GO TO 281
  282 BGOOD = .TRUE.
  281 IF (C2 .LT. 1) GO TO 284
      C2 = C2 - 1.0D0
      GO TO 283
  284 GO TO 201
  283 IF ( .NOT. TOSMLL) GO TO 286
      DC = DC + 1
      IF (DC .LT. 3) GO TO 288
      IF (PC .LT. QT2M1) GO TO 289
      IF (C2 .GE. 1) GO TO 199
  289 CONTINUE
  288 CONTINUE
  287 J = JZERO
      JZERO = J + 1
      IF (JZERO .NE. QT2M1) GO TO 291
      JZERO = 0
  291 CONTINUE
  290 GO TO 195
```

```
  286 CONTINUE
  285 DC = 0
      J = JZERO
      JZERO = J + 1
      IF (JZERO .NE. QT2M1) GO TO 293
      JZERO = 0
  293 CONTINUE
  292 GO TO 195
  199 C1 = C1 / 2
      C2 = (C2 - 1.0D0) / 2.0D0
      DC = 0
      PC = Q
      CU = (C1 ** ( - P)) * GR
      H = INTRVL / FDBLE(C1)
      MU = 0
      GO TO 296
  297 MU = MU + 1
  296 IF (MU .LE. QMINS1) GO TO 294
      GO TO 295
  294 HB(MU + 1) = B(MU + 1) * H
      HBS(MU + 1) = BS(MU + 1) * H
      GO TO 297
  295 HBSQZ = BSQZ * H
      I = 1
      GO TO 300
  301 I = I + 1
  300 IF (I .LE. N) GO TO 298
      GO TO 299
  298 EAU(I + 1) = EA(I + 1) * CU
      EAL(I + 1) = EAU(I + 1) * C2MQP5
      ERU(I + 1) = ER(I + 1) * CU
      ERL(I + 1) = ERU(I + 1) * C2MQP5
      HAL(I + 1) = ABS(EAL(I + 1) / HBSQZ)
      HRL(I + 1) = ABS(ERL(I + 1) / HBSQZ)
      GO TO 301
  299 K = J
      MU = 1
      GO TO 304
  305 MU = MU + 1
  304 IF (MU .LE. QMINS1) GO TO 302
```

```
      GO TO 303
  302 J = J - 1
      IF (J .GE. 0) GO TO 307
      J = J + QT2M1
  307 CONTINUE
  306 K = K - 2
      IF (K .GE. 0) GO TO 309
      K = K + QT2M1
  309 CONTINUE
  308 I = 1
      GO TO 312
  313 I = I + 1
  312 IF (I .LE. N) GO TO 310
      GO TO 311
  310 FH(J + 1, I + 1) = FH(K + 1, I + 1)
      GO TO 313
  311 GO TO 305
  303 JZERO = J
      J = JZERO - 1
      IF (J .GE. 0) GO TO 315
      J = J + QT2M1
  315 CONTINUE
  314 GO TO 195
  200 J = J - 1
      IF (J .GE. 0) GO TO 317
      J = J + QT2M1
  317 CONTINUE
  316 JZERO = J
      C1 = C1 + C1
      C2 = (C2 + C2) + 2.0D0
      IF (C2 .LT. Q) GO TO 201
      GO TO 194
  201 IF (FLIPPD) GO TO 320
      I = 1
      GO TO 323
  324 I = I + 1
  323 IF (I .LE. N) GO TO 321
      GO TO 322
  321 FC(I + 1) = FP(I + 1)
      GO TO 324
```

```
  322 CONTINUE                                                              00803000
  320 CONTINUE                                                              00804000
  319 IF (BGOOD) GO TO 326                                                  00805000
      I = 1                                                                 00806000
      GO TO 329                                                             00807000
  330 I = I + 1                                                             00808000
  329 IF (I .LE. N) GO TO 327                                               00809000
      GO TO 328                                                             00810000
  327 Y(I + 1) = YD(I + 1)                                                  00811000
      GO TO 330                                                             00812000
  328 CONTINUE                                                              00813000
  326 CONTINUE                                                              00814000
  325 X = XF - (INTRVL * (C2 / FDBLE(C1)))                                  00815000
      IF (C2 .EQ. 0) GO TO 332                                              00816000
      CALL SHANKS(N, X, XF, Y, RKSFNS, RKSRDR, RKSCFF, P, EA, ER, XF - X00817000
     1)                                                                     00818000
  332 CONTINUE                                                              00819000
      RETURN                                                                00828000
      END                                                                   00829000
C 334 BLOCK  8                                                               00830000


@PRT     .BUTCHR
```

```
GALLAHER-L-J*TPF$.BUTCHR
      SUBROUTINE BUTCHR(N, XI, XF, K, EA, ER, DXV, CON, EX, RKC, YIV,
     1RKSNF, RKSODR)
      DOUBLE PRECISION DXV
      COMMON/COMMON/F
      DOUBLE PRECISION YIV,EA,ER                                        00833000
      DIMENSION YIV(100),EA(100),ER(100)                                00834000
      INTEGER RKSNF,RKSODR                                              00835000
      INTEGER N,K                                                       00836000
      DOUBLE PRECISION XI,XF,DX,EX                                      00837000
      DOUBLE PRECISION RKC                                              00838000
      DIMENSION RKC(91)                                                 00839000
      DOUBLE PRECISION CON                                              00840000
      DIMENSION CON(37)                                                 00841000
      DOUBLE PRECISION Y,F                                              00842000
      DIMENSION Y(35,1),F(35, 1)
      DOUBLE PRECISION SC1,X                                            00844000
      DOUBLE PRECISION DX2                                              00845000
      DOUBLE PRECISION DX1,COA,COB,COLA,COLB,COGA,COGB,TEST,TEMPY,TEMPF,00846000
     1A1,A2,A3,C2                                                       00847000
      INTEGER I,J,CYL,INDEX,C1,M                                        00848000
      INTEGER CYL3                                                      00849000
      DOUBLE PRECISION SUMYIP,SUMYP,SUMYC,FV1                           00850000
      DIMENSION SUMYIP(100),SUMYP(100),SUMYC(100),FV1(100)              00851000
      DOUBLE PRECISION P2,T1,T2                                         00852000
      INTEGER COUNT,TOTCNT,CYL1,CYL2,M1                                 00853000
      DOUBLE PRECISION COO                                              00854000
      DIMENSION COO(19)                                                 00855000
      INTEGER CYO                                                       00856000
      INTEGER COUNTR                                                    00857000
      INTEGER KM1,K6,K61,K62                                            00858000
      DOUBLE PRECISION OMT                                              00859000
      DOUBLE PRECISION INTV                                             00860000
      INTEGER KM3,J2,J3,J6                                              00861000
      DOUBLE PRECISION XDXT,XDX                                         00862000
      DOUBLE PRECISION RE,AE                                            00863000
      DIMENSION RE(100),AE(100)                                         00864000
      DOUBLE PRECISION TEMP7,TEMP9                                      00865000
      DIMENSION TEMP7(100),TEMP9(100)                                   00866000
      INTEGER II                                                        00867000
```

```
      DEFINE FDBLE(I) = DBLE(FLOAT(I))                                      00868000
      EQUIVALENCE(F(18,1), Y(1, 1))
      DX = DXV
      I = 1                                                                 00869000
      GO TO 343                                                             00870000
  344 I = I + 1                                                             00871000
  343 IF (I .LE. N) GO TO 341                                               00872000
      GO TO 342                                                             00873000
  341 Y(1, I + 1) = YIV(I + 1)                                              00874000
      GO TO 344                                                             00875000
  342 IF (((K .NE. 1) .AND. K .NE. 2) .AND. K .NE. 3) GO TO 346             00876000
      OMT = .5                                                              00877000
      GO TO 345                                                             00878000
  346 OMT = 2.0D0 / 3.0D0                                                   00879000
  345 K6 = 6 * K                                                            00880000
      K61 = (6 * K) + 1                                                     00881000
      K62 = (6 * K) + 2                                                     00882000
      KM1 = K - 1                                                           00883000
      INTV = XF - XI                                                        00884000
      X = XI                                                                00885000
      C1 = 1                                                                00886000
  347 IF ((C1 .GE. K + 1) .AND. ABS(INTV) / FDBLE(C1) .LE. ABS(DX)) GO      00887000
     1TO 348                                                                00888000
      C1 = C1 + C1                                                          00889000
      GO TO 347                                                             00890000
  348 C2 = C1                                                               00891000
      P2 = 1.0D0 / (FDBLE(2 ** ((2 * K) + 4)))                              00892000
      CYL = 0                                                               00893000
      CYO = 0                                                               00894000
      TOTCNT = 0                                                            00895000
      II = 0                                                                00896000
      GO TO 351                                                             00897000
  352 II = II + 1                                                           00898000
  351 IF (II .LE. N) GO TO 349                                              00899000
      GO TO 350                                                             00900000
  349 TEMP7(II + 1) = Y(1, II + 1)                                          00901000
      GO TO 352                                                             00902000
  350 CALL FUNCTI(N, XI, TEMP7, TEMP9)                                      00903000
      II = 0                                                                00904000
      GO TO 355                                                             00905000
```

```
  356 II = II + 1                                                           00906000
  355 IF (II .LE. N) GO TO 353                                              00907000
      GO TO 354                                                             00908000
  353 F(1, II + 1) = TEMP9(II + 1)                                          00909000
      GO TO 356                                                             00910000
  354 CONTINUE                                                              00911000
  338 COUNTR = KM1                                                          00912000
      I = START(N, XI, XF, C1, EA, ER, KM1, X, YIV, Y, F, YIV, CYO, 16,     00913000
     12, EX, RKSNF, RKC)                                                    00914000
      C1 = C1 * I                                                           00915000
      C2 = (C2 * FDBLE(I)) - FDBLE(KM1)                                     00916000
      CYL = CYO                                                             00917000
  340 DX = INTV / FDBLE(C1)                                                 00918000
      KM3 = 3 * KM1                                                         00919000
      J = 0                                                                 00920000
      GO TO 359                                                             00921000
  360 J = J + 3                                                             00922000
  359 IF (J .LE. KM3) GO TO 357                                             00923000
      GO TO 358                                                             00924000
  357 J2 = 2 * J                                                            00925000
      COO(J + 1) = CON(J2 + 2) * DX                                         00926000
      COO(J + 2) = CON(J2 + 4) * DX                                         00927000
      COO(J + 3) = CON(J2 + 6) * DX                                         00928000
      GO TO 360                                                             00929000
 358  SC1 = 3.0D-4*(C1**EX)
      IF(K.LE.3) SC1 = SC1*10
      I = 1                                                                 00931000
      GO TO 363                                                             00932000
  364 I = I + 1                                                             00933000
  363 IF (I .LE. N) GO TO 361                                               00934000
      GO TO 362                                                             00935000
  361 AE(I + 1) = EA(I + 1) / SC1                                           00936000
      RE(I + 1) = ER(I + 1) / SC1                                           00937000
      GO TO 364                                                             00938000
  362 A1 = CON(K6 + 1) * DX                                                 00939000
      A2 = CON(K61 + 1) * DX                                                00940000
      A3 = CON(K62 + 1) * DX                                                00941000
  337 XDXT = X + DX * OMT                                                   00942000
      XDX = X + DX                                                          00943000
      I = 1                                                                 00944000
```

```
      GO TO 367
  368 I = I + 1
  367 IF (I .LE. N) GO TO 365
      GO TO 366
  365 SUMYC(I + 1) = 0
      SUMYP(I + 1) = SUMYC(I + 1)
      SUMYIP(I + 1) = SUMYP(I + 1)
      GO TO 368
  366 J = 0
      GO TO 371
  372 J = J + 1
  371 IF (J .LE. KM1) GO TO 369
      GO TO 370
  369 CYL3 = MOD((KM1 - J) + CYL,16)
      J3 = 3 * J
      J6 = 6 * J
      COA = CON(J6 + 1)
      COB = COO(J3 + 1)
      COLA = CON(J6 + 3)
      COLB = COO(J3 + 2)
      COGA = CON(J6 + 5)
      COGB = COO(J3 + 3)
      I = 1
      GO TO 375
  376 I = I + 1
  375 IF (I .LE. N) GO TO 373
      GO TO 374
  373 TEMPY = Y(CYL3 + 1, I + 1)
      TEMPF = F(CYL3 + 1, I + 1)
      SUMYIP(I + 1) = (SUMYIP(I + 1) + COA * TEMPY) + COB * TEMPF
      SUMYP(I + 1) = (SUMYP(I + 1) + COLA * TEMPY) + COLB * TEMPF
      SUMYC(I + 1) = (SUMYC(I + 1) + COGA * TEMPY) + COGB * TEMPF
      GO TO 376
  374 GO TO 372
  370 CALL FUNCTI(N, XDXT, SUMYIP, FV1)
      I = 1
      GO TO 379
  380 I = I + 1
  379 IF (I .LE. N) GO TO 377
      GO TO 378
```

```
377 TEMPF = FV1(I + 1)
    SUMYP(I + 1) = SUMYP(I + 1) + A1 * TEMPF
    SUMYC(I + 1) = SUMYC(I + 1) + A2 * TEMPF
    GO TO 380
378 CALL FUNCTI(N, XDX, SUMYP, FV1)
    CYL = MOD(CYL + 1,16)
    CYO = MOD(CYL + KM1,16)
    COUNT = 0
    I = 1
    GO TO 383
384 I = I + 1
383 IF (I .LE. N) GO TO 381
    GO TO 382
381 TEMPY = SUMYC(I + 1) + A3 * FV1(I + 1)
    T1 = AE(I + 1)
    T2 = ABS(RE(I + 1) * TEMPY)
    TEST = ABS(TEMPY - SUMYP(I + 1))
    IF ((TEST .LE. T1) .OR. TEST .LE. T2) GO TO 386
    C2 = C2 + C2
    CYL = MOD(CYL + 15,16)
    CYO = MOD(CYL + KM1,16)
    IF (C2 .GE. KM1) GO TO 388
    II = 0
    GO TO 391
392 II = II + 1
391 IF (II .LE. N) GO TO 389
    GO TO 390
389 TEMP7(II + 1) = Y(CYO + 1, II + 1)
    GO TO 392
390 CALL SHANKS(N, X, XF, TEMP7, RKSNF, RKSODR, RKC, EX, EA, ER, DX)
    II = 0
    GO TO 395
396 II = II + 1
395 IF (II .LE. N) GO TO 393
    GO TO 394
393 Y(CYO + 1, II + 1) = TEMP7(II + 1)
    GO TO 396
394 GO TO 339
388 CONTINUE
387 C1 = C1 + C1
```

```
      GO TO 338
  386 CONTINUE
  385 Y(CYO + 1, I + 1) = TEMPY
      IF ((TEST .GE. P2 * T1) .AND. TEST .GE. P2 * T2) GO TO 398
      COUNT = COUNT + 1
  398 CONTINUE
  397 GO TO 384
  382 C2 = C2 - 1.0D0
      X = XF - (DX * C2)
      IF (C2 .EQ. 0) GO TO 339
      IF (C2 .GE. 1) GO TO 401
      II = 0
      GO TO 404
  405 II = II + 1
  404 IF (II .LE. N) GO TO 402
      GO TO 403
  402 TEMP7(II + 1) = Y(CYO + 1, II + 1)
      GO TO 405
  403 CALL SHANKS(N, X, XF, TEMP7, RKSNF, RKSODR, RKC, EX, EA, ER, DX)
      II = 0
      GO TO 408
  409 II = II + 1
  408 IF (II .LE. N) GO TO 406
      GO TO 407
  406 Y(CYO + 1, II + 1) = TEMP7(II + 1)
      GO TO 409
  407 GO TO 339
  401 CONTINUE
  400 II = 0
      GO TO 412
  413 II = II + 1
  412 IF (II .LE. N) GO TO 410
      GO TO 411
  410 TEMP7(II + 1) = Y(CYO + 1, II + 1)
      GO TO 413
  411 CALL FUNCTI(N, X, TEMP7, TEMP9)
      II = 0
      GO TO 416
  417 II = II + 1
  416 IF (II .LE. N) GO TO 414
```

```
      GO TO 415
  414 F(CYO + 1, II + 1) = TEMP9(II + 1)
      GO TO 417
  415 IF (COUNT .NE. N) GO TO 419
      TOTCNT = TOTCNT + 1
      IF (TOTCNT .LT. 3) GO TO 421
      IF (COUNTR .LT. 2 * K) GO TO 423
      COUNTR = 0
      C2 = C2 / 2.0D0
      C1 = C1 / 2
      IF (C2 .GE. 1) GO TO 425
      II = 0
      GO TO 428
  429 II = II + 1
  428 IF (II .LE. N) GO TO 426
      GO TO 427
  426 TEMP7(II + 1) = Y(CYO + 1, II + 1)
      GO TO 429
  427 CALL SHANKS(N, X, XF, TEMP7, RKSNF, RKSODR, RKC, EX, EA, ER, UX)
      II = 0
      GO TO 432
  433 II = II + 1
  432 IF (II .LE. N) GO TO 430
      GO TO 431
  430 Y(CYO + 1, II + 1) = TEMP7(II + 1)
      GO TO 433
  431 GO TO 339
  425 CONTINUE
  424 TOTCNT = 0
      I = 1
      GO TO 436
  437 I = I + 1
  436 IF (I .LE. N) GO TO 434
      GO TO 435
  434 J = 1
      GO TO 440
  441 J = J + 1
  440 IF (J .LE. KM1) GO TO 438
      GO TO 439
  438 CYL1 = MOD((CYO + 16) - J,16)
```

```
          CYL2 = MOD((CYO + 16) - (2 * J),16)
          Y(CYL1 + 1, I + 1) = Y(CYL2 + 1, I + 1)
          F(CYL1 + 1, I + 1) = F(CYL2 + 1, I + 1)
          GO TO 441
      439 GO TO 437
      435 GO TO 340
      423 CONTINUE
      422 CONTINUE
      421 CONTINUE
      420 CONTINUE
      419 CONTINUE
      418 COUNTR = COUNTR + 1
          GO TO 337
      339 I = 1
          GO TO 444
      445 I = I + 1
      444 IF (I .LE. N) GO TO 442
          GO TO 443
      442 YIV(I + 1) = Y(CYO + 1, I + 1)
          GO TO 445
      443 CONTINUE
          RETURN
          END
    C 443 BLOCK  9

@PRT     .COWELL
```

```
GALLAHER-L-J*TPF$.COWELL
      SUBROUTINE COWELL(N, XI, XF, Y, EA, ER ,P, DXV, RKSFN, RKSRDR,
     1RKSCFF, Q, CWLLCF)
      DOUBLE PRECISION DXV
      COMMON/COMMON/FH(35,1)
      INTEGER N,RKSFN,RKSRDR,Q                                          01131000
      DOUBLE PRECISION XI,XF,P,DX                                       01132000
      DOUBLE PRECISION Y,EA,ER                                          01133000
      DIMENSION Y(100),EA(100),ER(100)                                  01134000
      DOUBLE PRECISION RKSCFF                                           01135000
      DIMENSION RKSCFF(91)                                              01136000
      DOUBLE PRECISION CWLLCF                                           01137000
      DIMENSION CWLLCF(52)                                              01138000
      INTEGER C1,M,MM1,QP1,TQP1,INDX,I1,I2,I3,I,J,K,CYI                 01139000
      INTEGER CRRCTC                                                    01140000
      DOUBLE PRECISION INT,C2,DFACTR,X,H,T1,T2,T3,T4,T5,T6              01141000
      LOGICAL DFLAG,PFLAG                                               01142000
      INTEGER II                                                        01143000
      DOUBLE PRECISION FH                                               01144000
      DOUBLE PRECISION YMID1,YP,YC,YM,CS,FP,HDM1F,TEMP9,HDM1FM,EAV,ERV, 01146000
     1EAVD,ERVD                                                         01147000
      DIMENSION YMID1(100),YP(100),YC(100),YM(100),CS(100),FP(100),HDM1F01148000
     1(100),TEMP9(100),HDM1FM(100),EAV(100),ERV(100),EAVD(100),ERVD(100)01149000
      DOUBLE PRECISION PCOEFF,CCOEFF,MCOEFF                             01150000
      DIMENSION PCOEFF(18),CCOEFF(18),MCOEFF(18)                        01151000
      DEFINE FDBLE(I) = DBLE(FLOAT(I))                                  01152000
      DX = DXV
      INT = XF - XI                                                     01153000
      C1 = 1                                                            01154000
  446 C1 = C1 + C1                                                      01155000
      IF (C1 .LT. Q) GO TO 446                                          01156000
      IF (DX .GE. 0) GO TO 459                                          01157000
      DX =  - DX                                                        01158000
  459 CONTINUE                                                          01159000
  458 IF (DX .EQ. 0) GO TO 461                                          01160000
  447 IF (ABS(INT) / FDBLE(C1) .LE. DX) GO TO 463                       01161000
      C1 = C1 + C1                                                      01162000
      GO TO 447                                                         01163000
  463 CONTINUE                                                          01164000
  462 CONTINUE                                                          01165000
```

```
  461 CONTINUE                                                                01166000
  460 C2 = C1                                                                 01167000
      M = Q / 2                                                               01168000
      MM1 = M - 1                                                             01169000
      QP1 = Q + 1                                                             01170000
      TQP1 = QP1 + Q                                                          01171000
      DFACTR = 2 ** (Q + 3)                                                   01172000
      INDX = 0                                                                01173000
      X = XI                                                                  01174000
      CALL FUNCTI(N, X, Y, TEMP9)                                             01175000
      II = 0                                                                  01176000
      GO TO 466                                                               01177000
  467 II = II + 1                                                             01178000
  466 IF (II .LE. N) GO TO 464                                                01179000
      GO TO 465                                                               01180000
  464 FH(1, II + 1) = TEMP9(II + 1)                                           01181000
      GO TO 467                                                               01182000
  465 CONTINUE                                                                01183000
  452 I1 = START(N, XI, XF, C1, EA, ER, Q, X, Y, FH, FH, YMID1, INDX,        01184000
     1TQP1, 1, P, RKSFN, RKSCFF)                                              01185000
      C1 = I1 * C1                                                            01186000
      C2 = (C2 * FDBLE(I1)) - FDBLE(Q)                                        01187000
      INDX = MOD(INDX + Q,TQP1)                                               01188000
      IF (C2 .LT. M) GO TO 456                                                01189000
  453 H = INT / FDBLE(C1)                                                     01190000
      K = 0                                                                   01191000
      GO TO 471                                                               01192000
  472 K = K + 1                                                               01193000
  471 IF (K .LE. Q) GO TO 469                                                 01194000
      GO TO 470                                                               01195000
  469 PCOEFF(K + 1) = CWLLCF(K + 1) * H                                       01196000
      I1 = K + QP1                                                            01197000
      CCOEFF(K + 1) = CWLLCF(I1 + 1) * H                                      01198000
      III2 = I1 + QP1                                                         01199000
      MCOEFF(K + 1) = CWLLCF(III2 + 1) * H                                    01200000
      GO TO 472                                                               01201000
  470 T1 = (C1 ** P) * 10.0D0                                                 01202000
      J = 1                                                                   01203000
      GO TO 475                                                               01204000
  476 J = J + 1                                                               01205000
```

```
475 IF (J .LE. N) GO TO 473
    GO TO 474
473 EAV(J + 1) = EA(J + 1) / T1
    EAVD(J + 1) = EAV(J + 1) / DFACTR
    ERV(J + 1) = ER(J + 1) / T1
    ERVD(J + 1) = ERV(J + 1) / DFACTR
    GO TO 476
474 T1 = MCOEFF(1)
    J = 1
    GO TO 479
480 J = J + 1
479 IF (J .LE. N) GO TO 477
    GO TO 478
477 HDM1F(J + 1) = YMID1(J + 1) - (FH(INDX + 1, J + 1) * T1)
    GO TO 480
478 CYI = INDX + TQP1
    I3 = CYI - QP1
    K = 1
    GO TO 483
484 K = K + 1
483 IF (K .LE. M) GO TO 481
    GO TO 482
481 I1 = MOD(CYI - K,TQP1)
    I2 = MOD(I3 + K,TQP1)
    T1 = MCOEFF(K + 1) - H
    III2 = QP1 - K
    T2 = MCOEFF(III2 + 1)
    J = 1
    GO TO 487
488 J = J + 1
487 IF (J .LE. N) GO TO 485
    GO TO 486
485 HDM1F(J + 1) = (HDM1F(J + 1) - (FH(I1 + 1, J + 1) * T1)) - (FH(I2
   1+ 1, J + 1) * T2)
    GO TO 488
486 GO TO 484
482 J = 1
    GO TO 491
492 J = J + 1
491 IF (J .LE. N) GO TO 489
```

```
      GO TO 490                                         01246000
  489 HDM1FM(J + 1) = HDM1F(J + 1)                      01247000
      GO TO 492                                         01248000
  490 DFLAG = .FALSE.                                   01249000
  454 I = 1                                             01250000
      GO TO 495                                         01251000
  496 I = I + 1                                         01252000
  495 IF (I .LE. M) GO TO 493                           01253000
      GO TO 494                                         01254000
  493 INDX = MOD(INDX + 1,TQP1)                         01255000
      CYI = INDX + TQP1                                 01256000
      X = XF - ((C2 - FDBLE(I)) * H)                    01257000
      I1 = MOD(CYI - 1,TQP1)                            01258000
      T1 = PCOEFF(1)                                    01259000
      T2 = CCOEFF(2)                                    01260000
      J = 1                                             01261000
      GO TO 499                                         01262000
  500 J = J + 1                                         01263000
  499 IF (J .LE. N) GO TO 497                           01264000
      GO TO 498                                         01265000
  497 T3 = FH(I1 + 1, J + 1)                            01266000
      HDM1F(J + 1) = HDM1F(J + 1) + H * T3              01267000
      T4 = HDM1F(J + 1)                                 01268000
      YP(J + 1) = T4 + T1 * T3                          01269000
      CS(J + 1) = T4 + T2 * T3                          01270000
      GO TO 500                                         01271000
  498 I3 = CYI - QP1                                    01272000
      K = 2                                             01273000
      GO TO 503                                         01274000
  504 K = K + 1                                         01275000
  503 IF (K .LE. M) GO TO 501                           01276000
      GO TO 502                                         01277000
  501 I1 = MOD(CYI - K,TQP1)                            01278000
      I2 = MOD(I3 + K,TQP1)                             01279000
      T1 = PCOEFF(K)                                    01280000
      T2 = CCOEFF(K + 1)                                01281000
      III3 = Q - K                                      01282000
      T3 = PCOEFF(III3 + 1)                             01283000
      III3 = QP1 - K                                    01284000
      T4 = CCOEFF(III3 + 1)                             01285000
```

```
      J = 1
      GO TO 507
  508 J = J + 1
  507 IF (J .LE. N) GO TO 505
      GO TO 506
  505 T5 = FH(I1 + 1, J + 1)
      T6 = FH(I2 + 1, J + 1)
      YP(J + 1) = (YP(J + 1) + T1 * T5) + T3 * T6
      CS(J + 1) = (CS(J + 1) + T2 * T5) + T4 * T6
      GO TO 508
  506 GO TO 504
  502 I1 = MOD(CYI - Q,TQP1)
      I2 = MOD(CYI - QP1,TQP1)
      T1 = PCOEFF(Q)
      T2 = CCOEFF(Q + 1)
      T3 = PCOEFF(Q + 1)
      J = 1
      GO TO 511
  512 J = J + 1
  511 IF (J .LE. N) GO TO 509
      GO TO 510
  509 T4 = FH(I1 + 1, J + 1)
      YP(J + 1) = (YP(J + 1) + T1 * T4) + T3 * FH(I2 + 1, J + 1)
      CS(J + 1) = CS(J + 1) + T2 * T4
      GO TO 512
  510 T2 = CCOEFF(1)
      CRRCTC = 1
  455 CALL FUNCTI(N, X, YP, FP)
      J = 1
      GO TO 515
  516 J = J + 1
  515 IF (J .LE. N) GO TO 513
      GO TO 514
  513 YC(J + 1) = CS(J + 1) + T2 * FP(J + 1)
      T3 = YC(J + 1)
      T1 = ABS(T3 - YP(J + 1))
      IF (T1 .LE. EAV(J + 1)) GO TO 518
      IF (T1 .LE. ERV(J + 1) * ABS(T3)) GO TO 520
      J = J + 1
      GO TO 523
```

```
  524 J = J + 1
  523 IF (J .LE. N) GO TO 521
      GO TO 522
  521 YC(J + 1) = CS(J + 1) + T2 * FP(J + 1)
      GO TO 524
  522 CALL FUNCTI(N, X, YC, FP)
      J = 1
      GO TO 527
  528 J = J + 1
  527 IF (J .LE. N) GO TO 525
      GO TO 526
  525 YP(J + 1) = CS(J + 1) + T2 * FP(J + 1)
      T3 = YP(J + 1)
      T1 = ABS(T3 - YC(J + 1))
      IF (T1 .LE. EAV(J + 1)) GO TO 530
      IF (T1 .LE. ERV(J + 1) * ABS(T3)) GO TO 532
      J = J + 1
      GO TO 535
  536 J = J + 1
  535 IF (J .LE. N) GO TO 533
      GO TO 534
  533 YP(J + 1) = CS(J + 1) + T2 * FP(J + 1)
      GO TO 536
  534 CRRCTC = CRRCTC + 2
      IF (CRRCTC .LE. 8) GO TO 538
      INDX = MOD(CYI - I,TQP1)
      GO TO 451
  538 CONTINUE
  537 GO TO 455
  532 CONTINUE
  531 CONTINUE
  530 CONTINUE
  529 GO TO 528
  526 CALL FUNCTI(N, X, YP, TEMP9)
      II = 0
      GO TO 541
  542 II = II + 1
  541 IF (II .LE. N) GO TO 539
      GO TO 540
  539 FH(INDX + 1, II + 1) = TEMP9(II + 1)
```

```
      GO TO 542
  540 PFLAG = .TRUE.
      CRRCTC = CRRCTC + 1
      GO TO 448
  520 CONTINUE
  519 CONTINUE
  518 CONTINUE
  517 GO TO 516
  514 CALL FUNCTI(N, X, YC, TEMP9)
      II = 0
      GO TO 545
  546 II = II + 1
  545 IF (II .LE. N) GO TO 543
      GO TO 544
  543 FH(INDX + 1, II + 1) = TEMP9(II + 1)
      GO TO 546
  544 PFLAG = .FALSE.
  448 GO TO 496
  494 I1 = MOD(CYI - M,TQP1)
      T1 = MCOEFF(M + 1)
      J = 1
      GO TO 549
  550 J = J + 1
  549 IF (J .LE. N) GO TO 547
      GO TO 548
  547 YM(J + 1) = HDM1FM(J + 1) + T1 * FH(I1 + 1, J + 1)
      GO TO 550
  548 I3 = CYI - Q
      K = 0
      GO TO 553
  554 K = K + 1
  553 IF (K .LE. MM1) GO TO 551
      GO TO 552
  551 I1 = MOD(CYI - K,TQP1)
      I2 = MOD(I3 + K,TQP1)
      T1 = MCOEFF(K + 1)
      III2 = Q - K
      T2 = MCOEFF(III2 + 1)
      J = 1
      GO TO 557
```

```
558 J = J + 1
557 IF (J .LE. N) GO TO 555
    GO TO 556
555 YM(J + 1) = (YM(J + 1) + T1 * FH(I1 + 1, J + 1)) + T2 * FH(I2 + 1,
   1 J + 1)
    GO TO 558
556 GO TO 554
552 IF ( .NOT. DFLAG) GO TO 560
    J = 1
    GO TO 563
564 J = J + 1
563 IF (J .LE. N) GO TO 561
    GO TO 562
561 T3 = Y(J + 1)
    T2 = ABS(T3 - YM(J + 1))
    IF (T2 .LE. EAVD(J + 1)) GO TO 566
    IF (T2 .LE. ERVD(J + 1) * ABS(T3)) GO TO 568
    IF (T2 .LE. EAV(J + 1)) GO TO 570
    IF (T2 .GT. ERV(J + 1) * ABS(T3)) GO TO 450
570 CONTINUE
569 GO TO 449
568 CONTINUE
567 CONTINUE
566 CONTINUE
565 GO TO 564
562 C2 = C2 - FDBLE(M)
    C2 = C2 / 2.0D0
    IF ( .NOT. PFLAG) GO TO 573
    J = 1
    GO TO 576
577 J = J + 1
576 IF (J .LE. N) GO TO 574
    GO TO 575
574 Y(J + 1) = YP(J + 1)
    GO TO 577
575 GO TO 572
573 J = 1
    GO TO 580
581 J = J + 1
580 IF (J .LE. N) GO TO 578
```

```
      GO TO 579                                          01446000
  578 Y(J + 1) = YC(J + 1)                               01447000
      GO TO 581                                          01448000
  579 CONTINUE                                           01449000
  572 C1 = C1 / 2                                        01450000
      IF (C2 .LT. M) GO TO 456                           01451000
      INDX = INDX + 1                                    01452000
      K = 1                                              01453000
      GO TO 585                                          01454000
  586 K = K + 1                                          01455000
  585 IF (K .LE. Q) GO TO 583                            01456000
      GO TO 584                                          01457000
  583 INDX = MOD(INDX + 1,TQP1)                          01458000
      I1 = MOD(INDX + K,TQP1)                            01459000
      J = 1                                              01460000
      GO TO 589                                          01461000
  590 J = J + 1                                          01462000
  589 IF (J .LE. N) GO TO 587                            01463000
      GO TO 588                                          01464000
  587 FH(INDX + 1, J + 1) = FH(I1 + 1, J + 1)            01465000
      GO TO 590                                          01466000
  588 GO TO 586                                          01467000
  584 GO TO 453                                          01468000
  560 CONTINUE                                           01469000
  559 J = 0                                              01470000
  449 J = J + 1                                          01471000
      GO TO 593                                          01472000
  594 J = J + 1                                          01473000
  593 IF (J .LE. N) GO TO 591                            01474000
      GO TO 592                                          01475000
  591 T3 = Y(J + 1)                                      01476000
      T2 = ABS(T3 - YM(J + 1))                           01477000
      IF (T2 .LE. EAV(J + 1)) GO TO 596                  01478000
      IF (T2 .LE. ERV(J + 1) * ABS(T3)) GO TO 598        01479000
  450 INDX = MOD(CYI - M,TQP1)                           01480000
  451 C1 = C1 + C1                                       01481000
      X = XF - (C2 * H)                                  01482000
      C2 = C2 + C2                                       01483000
      GO TO 452                                          01484000
  598 CONTINUE                                           01485000
```

```
  597 CONTINUE
  596 CONTINUE
  595 GO TO 594
  592 C2 = C2 - FDBLE(M)
      IF (C2 .LT. M) GO TO 600
      IF ( .NOT. PFLAG) GO TO 602
      J = 1
      GO TO 605
  606 J = J + 1
  605 IF (J .LE. N) GO TO 603
      GO TO 604
  603 YMID1(J + 1) = Y(J + 1)
      Y(J + 1) = YP(J + 1)
      HDM1FM(J + 1) = HDM1F(J + 1)
      GO TO 606
  604 GO TO 601
  602 J = 1
      GO TO 609
  610 J = J + 1
  609 IF (J .LE. N) GO TO 607
      GO TO 608
  607 YMID1(J + 1) = Y(J + 1)
      Y(J + 1) = YC(J + 1)
      HDM1FM(J + 1) = HDM1F(J + 1)
      GO TO 610
  608 CONTINUE
  601 DFLAG = .TRUE.
      GO TO 454
  600 CONTINUE
  599 IF ( .NOT. PFLAG) GO TO 612
      J = 1
      GO TO 615
  616 J = J + 1
  615 IF (J .LE. N) GO TO 613
      GO TO 614
  613 Y(J + 1) = YP(J + 1)
      GO TO 616
  614 GO TO 611
  612 J = 1
      GO TO 619
```

```
      620 J = J + 1
      619 IF (J .LE. N) GO TO 617
          GO TO 618
      617 Y(J + 1) = YC(J + 1)
          GO TO 620
      618 CONTINUE
      611 CONTINUE
      456 IF (C2 .LE. 0) GO TO 622
          CALL SHANKS(N, X, XF, Y, RKSFN, RKSRDR, RKSCFF, P, EA, ER, ABS(INT
         1) / FDBLE(C1))
      622 CONTINUE
      621 CONTINUE
          RETURN
          END
    C 621 BLOCK   10


@PRT     .START
```

```
GALLAHER-L-J*TPF$.START
               FUNCTION START(N, XI, XF,IC1, EA, ER, M, X, YIV, YH, FH,
     1YFV, CYI, CYM, PA, P, FNEVAL, RKSCNS)
      LOGICAL COMP
      INTEGER N,C1,M,CYI,CYM,PA,FNEVAL                                  00027000
      DOUBLE PRECISION XI,XF,X,P                                        00028000
      DOUBLE PRECISION EA,ER,YIV,YFV                                    00029000
      DIMENSION EA(100),ER(100),YIV(100),YFV(100)                       00030000
      DOUBLE PRECISION RKSCNS                                           00031000
      DIMENSION RKSCNS(91)                                              00032000
      DOUBLE PRECISION YH,FH                                            00033000
      DIMENSION YH(35,100),FH(35,100)                                   00034000
      INTEGER I,J,K,L,CFFCNT,FNMAX,INDX,NINDX,CNTR                      00035000
      DOUBLE PRECISION INT,H,TWOH,T1                                    00036000
      DOUBLE PRECISION HC,TWOHC                                         00037000
      DIMENSION HC(91),TWOHC(91)                                        00038000
      DOUBLE PRECISION EAV,ERV,Y1,Y2,Y3,Y4                              00039000
      DIMENSION EAV(100),ERV(100),Y1(100),Y2(100),Y3(100),Y4(100)       00040000
      DOUBLE PRECISION G                                                00041000
      DIMENSION G(28,100)                                               00042000
      DOUBLE PRECISION TEMP7,TEMP8,TEMP9                                00043000
      DIMENSION TEMP7(100),TEMP8(100),TEMP9(100)                        00044000
      INTEGER II                                                        00045000
      DEFINE FDBLE(I) = DBLE(FLOAT(I))                                  00046000
      C1 = IC1
      CNTR = 1                                                          00047000
      IF (M .EQ. 0) GO TO 49                                            00048000
      CFFCNT = (((FNEVAL + 3) * FNEVAL) / 2) - 2                        00049000
      FNMAX = FNEVAL - 2                                                00050000
      INT = XF - XI                                                     00051000
      TWOH = (INT + INT) / FDBLE(C1)                                    00052000
      H = INT / FDBLE(C1)                                               00053000
      I = 0                                                             00054000
      GO TO 52                                                          00055000
   53 I = I + 1                                                         00056000
   52 IF (I .LE. CFFCNT) GO TO 50                                       00057000
      GO TO 51                                                          00058000
   50 T1 = RKSCNS(I + 1)                                                00059000
      HC(I + 1) = T1 * H                                                00060000
      TWOHC(I + 1) = T1 * TWOH                                          00061000
```

```
      GO TO 53
   51 INDX = CYI -(CYI/CYM) * CYM
      T1 = (FDBLE(C1) / 2.0D0) ** P
      J = 1
      GO TO 56
   57 J = J + 1
   56 IF (J .LE. N) GO TO 54
      GO TO 55
   54 EAV(J + 1) = EA(J + 1) / T1
      ERV(J + 1) = ER(J + 1) / T1
      GO TO 57
   55 IF (PA .NE. 2) GO TO 59
      NINDX = MOD(INDX + 1,CYM)
      K = 0
      GO TO 3
   59 CONTINUE
   58 IF (PA .NE. 1) GO TO 61
      L = M / 2
   61 CONTINUE
   60 II = 0
      GO TO 64
   65 II = II + 1
   64 IF (II .LE. N) GO TO 62
      GO TO 63
   62 TEMP9(II + 1) = FH(INDX + 1, II + 1)
      GO TO 65
   63 CALL RUNKUT(N, X, FNMAX, TWOHC, YIV, Y1, TEMP9, G)
      GO TO 4
    1 TWOH = H
      CNTR = CNTR + CNTR
      C1 = C1 + C1
      H = INT / FDBLE(C1)
      I = 0
      GO TO 68
   69 I = I + 1
   68 IF (I .LE. CFFCNT) GO TO 66
      GO TO 67
   66 TWOHC(I + 1) = HC(I + 1)
      HC(I + 1) = RKSCNS(I + 1) * H
      GO TO 69
```

```
   67 INDX = CYI -(CYI/CYM) * CYM
      T1 = (FDBLE(C1) / 2.0D0) ** P
      IF (PA .NE. 2) GO TO 71
      K = 0
      NINDX = MOD(INDX + 1,CYM)
      J = 1
      GO TO 74
   75 J = J + 1
   74 IF (J .LE. N) GO TO 72
      GO TO 73
   72 EAV(J + 1) = EA(J + 1) / T1
      ERV(J + 1) = ER(J + 1) / T1
      Y1(J + 1) = YH(NINDX + 1, J + 1)
      GO TO 75
   73 CONTINUE
    2 II = 0
      GO TO 78
   79 II = II + 1
   78 IF (II .LE. N) GO TO 76
      GO TO 77
   76 TEMP7(II + 1) = YH(INDX + 1, II + 1)
      TEMP9(II + 1) = FH(INDX + 1, II + 1)
      GO TO 79
   77 CALL RUNKUT(N, (FDBLE(K) * H) + X, FNMAX, HC, TEMP7, TEMP8, TEMP9,
     1 G)
      II = 0
      GO TO 82
   83 II = II + 1
   82 IF (II .LE. N) GO TO 80
      GO TO 81
   80 YH(NINDX + 1, II + 1) = TEMP8(II + 1)
      GO TO 83
   81 K = K + 1
      INDX = NINDX
      NINDX = MOD(INDX + 1,CYM)
      II = 0
      GO TO 86
   87 II = II + 1
   86 IF (II .LE. N) GO TO 84
      GO TO 85
```

```
   84 TEMP7(II + 1) = YH(INDX + 1, II + 1)
      GO TO 87
   85 CALL FUNCTI(N, (FDBLE(K) * H) + X, TEMP7, TEMP9)
      II = 0
      GO TO 90
   91 II = II + 1
   90 IF (II .LE. N) GO TO 88
      GO TO 89
   88 FH(INDX + 1, II + 1) = TEMP9(II + 1)
      GO TO 91
   89 CALL RUNKUT(N, (FDBLE(K) * H) + X, FNMAX, HC, TEMP7, TEMP8, TEMP9,
     1 G)
      II = 0
      GO TO 94
   95 II = II + 1
   94 IF (II .LE. N) GO TO 92
      GO TO 93
   92 YH(NINDX + 1, II + 1) = TEMP8(II + 1)
      GO TO 95
   93 IF ( .NOT. COMP(N, EAV, ERV, Y1, TEMP8)) GO TO 97
      K = K + 1
      IF (K .GE. M) GO TO 99
      INDX = NINDX
      NINDX = MOD(INDX + 1,CYM)
      II = 0
      GO TO 102
  103 II = II + 1
  102 IF (II .LE. N) GO TO 100
      GO TO 101
  100 TEMP7(II + 1) = YH(INDX + 1, II + 1)
      GO TO 103
  101 CALL FUNCTI(N, (FDBLE(K) * H) + X, TEMP7, TEMP9)
      II = 0
      GO TO 106
  107 II = II + 1
  106 IF (II .LE. N) GO TO 104
      GO TO 105
  104 FH(INDX + 1, II + 1) = TEMP9(II + 1)
      GO TO 107
  105 CONTINUE
```

```
    3 II = 0
      GO TO 110
  111 II = II + 1
  110 IF (II .LE. N) GO TO 108
      GO TO 109
  108 TEMP7(II + 1) = YH(INDX + 1, II + 1)
      TEMP9(II + 1) = FH(INDX + 1, II + 1)
      GO TO 111
  109 CALL RUNKUT(N, (FDBLE(K) * H) + X, FNMAX, TWOHC, TEMP7, Y1, TEMP9,
     1 G)
      GO TO 2
   99 CONTINUE
   98 IF (K .NE. M) GO TO 113
      INDX = NINDX
      II = 0
      GO TO 116
  117 II = II + 1
  116 IF (II .LE. N) GO TO 114
      GO TO 115
  114 TEMP7(II + 1) = YH(INDX + 1, II + 1)
      GO TO 117
  115 CALL FUNCTI(N, (FDBLE(K) * H) + X, TEMP7, TEMP9)
      II = 0
      GO TO 120
  121 II = II + 1
  120 IF (II .LE. N) GO TO 118
      GO TO 119
  118 FH(INDX + 1, II + 1) = TEMP9(II + 1)
      GO TO 121
  119 CONTINUE
  113 CONTINUE
  112 GO TO 96
   97 GO TO 1
   96 GO TO 70
   71 J = 1
      GO TO 124
  125 J = J + 1
  124 IF (J .LE. N) GO TO 122
      GO TO 123
  122 EAV(J + 1) = EA(J + 1) / T1
```

```
      ERV(J + 1) = ER(J + 1) / T1
      Y1(J + 1) = Y2(J + 1)
      GO TO 125
  123 CONTINUE
    4 II = 0
      GO TO 128
  129 II = II + 1
  128 IF (II .LE. N) GO TO 126
      GO TO 127
  126 TEMP9(II + 1) = FH(INDX + 1, II + 1)
      GO TO 129
  127 CALL RUNKUT(N, X, FNMAX, HC, YIV, Y2, TEMP9, G)
      INDX = MOD(INDX + 1,CYM)
      CALL FUNCTI(N, X + H, Y2, TEMP9)
      II = 0
      GO TO 132
  133 II = II + 1
  132 IF (II .LE. N) GO TO 130
      GO TO 131
  130 FH(INDX + 1, II + 1) = TEMP9(II + 1)
      GO TO 133
  131 CALL RUNKUT(N, X + H, FNMAX, HC, Y2, Y4, TEMP9, G)
      K = 2
      IF (PA .NE. 0) GO TO 135
    5 IF ( .NOT. COMP(N, EAV, ERV, Y1, Y4)) GO TO 137
      IF (K .GE. M) GO TO 139
      INDX = MOD(INDX + 1,CYM)
      CALL FUNCTI(N, (FDBLE(K) * H) + X, Y4, TEMP9)
      II = 0
      GO TO 142
  143 II = II + 1
  142 IF (II .LE. N) GO TO 140
      GO TO 141
  140 FH(INDX + 1, II + 1) = TEMP9(II + 1)
      GO TO 143
  141 CALL RUNKUT(N, (FDBLE(K) * H) + X, FNMAX, TWOHC, Y4, Y1, TEMP9, G)
      CALL RUNKUT(N, (FDBLE(K) * H) + X, FNMAX, HC, Y4, Y3, TEMP9, G)
      K = K + 1
      INDX = MOD(INDX + 1,CYM)
      CALL FUNCTI(N, (FDBLE(K) * H) + X, Y3, TEMP9)
```

```
      II = 0
      GO TO 146
  147 II = II + 1
  146 IF (II .LE. N) GO TO 144
      GO TO 145
  144 FH(INDX + 1, II + 1) = TEMP9(II + 1)
      GO TO 147
  145 CALL RUNKUT(N, (FDBLE(K) * H) + X, FNMAX, HC, Y3, Y4, TEMP9, G)
      K = K + 1
      GO TO 5
  139 CONTINUE
  138 IF (K .NE. M) GO TO 149
      INDX = MOD(INDX + 1,CYM)
      CALL FUNCTI(N, (FDBLE(K) * H) + X, Y4, TEMP9)
      II = 0
      GO TO 152
  153 II = II + 1
  152 IF (II .LE. N) GO TO 150
      GO TO 151
  150 FH(INDX + 1, II + 1) = TEMP9(II + 1)
      GO TO 153
  151 J = 1
      GO TO 156
  157 J = J + 1
  156 IF (J .LE. N) GO TO 154
      GO TO 155
  154 YFV(J + 1) = Y4(J + 1)
      GO TO 157
  155 GO TO 148
  149 J = 1
      GO TO 160
  161 J = J + 1
  160 IF (J .LE. N) GO TO 158
      GO TO 159
  158 YFV(J + 1) = Y3(J + 1)
      GO TO 161
  159 CONTINUE
  148 GO TO 136
  137 GO TO 1
  136 GO TO 134
```

```
 135 CONTINUE
   6 IF ( .NOT. COMP(N, EAV, ERV, Y1, Y4)) GO TO 163
     INDX = MOD(INDX + 1,CYM)
     CALL FUNCTI(N, (FDBLE(K) * H) + X, Y4, TEMP9)
     II = 0
     GO TO 166
 167 II = II + 1
 166 IF (II .LE. N) GO TO 164
     GO TO 165
 164 FH(INDX + 1, II + 1) = TEMP9(II + 1)
     GO TO 167
 165 IF (K .GE. M) GO TO 169
     IF (K .NE. L) GO TO 171
     J = 1
     GO TO 174
 175 J = J + 1
 174 IF (J .LE. N) GO TO 172
     GO TO 173
 172 YFV(J + 1) = Y4(J + 1)
     GO TO 175
 173 CONTINUE
 171 CONTINUE
 170 II = 0
     GO TO 178
 179 II = II + 1
 178 IF (II .LE. N) GO TO 176
     GO TO 177
 176 TEMP9(II + 1) = FH(INDX + 1, II + 1)
     GO TO 179
 177 CALL RUNKUT(N, (FDBLE(K) * H) + X, FNMAX, TWOHC, Y4, Y1, TEMP9, G)
     CALL RUNKUT(N, (FDBLE(K) * H) + X, FNMAX, HC, Y4, Y3, TEMP9, G)
     K = K + 1
     INDX = MOD(INDX + 1,CYM)
     CALL FUNCTI(N, (FDBLE(K) * H) + X, Y3, TEMP9)
     II = 0
     GO TO 182
 183 II = II + 1
 182 IF (II .LE. N) GO TO 180
     GO TO 181
 180 FH(INDX + 1, II + 1) = TEMP9(II + 1)
```

```
              GO TO 183
          181 IF (K .NE. L) GO TO 185
              J = 1
              GO TO 188
          189 J = J + 1
          188 IF (J .LE. N) GO TO 186
              GO TO 187
          186 YFV(J + 1) = Y3(J + 1)
              GO TO 189
          187 CONTINUE
          185 CONTINUE
          184 CALL RUNKUT(N, (FDBLE(K) * H) + X, FNMAX, HC, Y3, Y4, TEMP9, G)
              K = K + 1
              GO TO 6
          169 CONTINUE
          168 J = 1
              GO TO 192
          193 J = J + 1
          192 IF (J .LE. N) GO TO 190
              GO TO 191
          190 YIV(J + 1) = Y4(J + 1)
              GO TO 193
          191 GO TO 162
          163 GO TO 1
          162 CONTINUE
          134 CONTINUE
           70 X = (FDBLE(M) * H) + X
           49 CONTINUE
           48 START = CNTR
              RETURN
              END
       C   48 BLOCK   5

@PRT      .RUNKUT
```

GALLAHER-L-J*TPF$.RUNKUT

```
      SUBROUTINE RUNKUT(N, X, FNMAX, COEFF, YIV, YFV, FV, G)
      INTEGER N,FNMAX
      DOUBLE PRECISION X
      DOUBLE PRECISION COEFF
      DIMENSION COEFF(91)
      DOUBLE PRECISION YIV,YFV,FV
      DIMENSION YIV(100),YFV(100),FV(100)
      DOUBLE PRECISION G
      DIMENSION G(28,100)
      DOUBLE PRECISION TEMP9
      DIMENSION TEMP9(100)
      INTEGER II
      INTEGER I,J,K,CCNT
      DOUBLE PRECISION TEMP
      DEFINE FDBLE(I) = DBLE(FLOAT(I))
      J = 0
      I = J
      CCNT = 0
      TEMP = COEFF(CCNT + 1)
      I = 0
      GO TO 9
   10 I = I + 1
    9 IF (I .LE. FNMAX) GO TO 7
      GO TO 8
    7 J = 1
      GO TO 13
   14 J = J + 1
   13 IF (J .LE. N) GO TO 11
      GO TO 12
   11 YFV(J + 1) = (FV(J + 1) * TEMP) + YIV(J + 1)
      GO TO 14
   12 III3 = I - 1
      K = 0
      GO TO 17
   18 K = K + 1
   17 IF (K .LE. III3) GO TO 15
      GO TO 16
   15 CCNT = CCNT + 1
      TEMP = COEFF(CCNT + 1)
```

```
      J = 1                                                      00413000
      GO TO 21                                                   00414000
   22 J = J + 1                                                  00415000
   21 IF (J .LE. N) GO TO 19                                     00416000
      GO TO 20                                                   00417000
   19 YFV(J + 1) = (G(K + 1, J + 1) * TEMP) + YFV(J + 1)         00418000
      GO TO 22                                                   00419000
   20 GO TO 18                                                   00420000
   16 CCNT = CCNT + 1                                            00421000
      TEMP = X + COEFF(CCNT + 1)                                 00422000
      CALL FUNCTI(N, TEMP, YFV, TEMP9)                           00423000
      II = 0                                                     00424000
      GO TO 25                                                   00425000
   26 II = II + 1                                                00426000
   25 IF (II .LE. N) GO TO 23                                    00427000
      GO TO 24                                                   00428000
   23 G(I + 1, II + 1) = TEMP9(II + 1)                           00429000
      GO TO 26                                                   00430000
   24 CCNT = CCNT + 1                                            00431000
      TEMP = COEFF(CCNT + 1)                                     00432000
      GO TO 10                                                   00433000
    8 J = 1                                                      00434000
      GO TO 29                                                   00435000
   30 J = J + 1                                                  00436000
   29 IF (J .LE. N) GO TO 27                                     00437000
      GO TO 28                                                   00438000
   27 YFV(J + 1) = (FV(J + 1) * TEMP) + YIV(J + 1)               00439000
      GO TO 30                                                   00440000
   28 K = 0                                                      00441000
      GO TO 33                                                   00442000
   34 K = K + 1                                                  00443000
   33 IF (K .LE. FNMAX) GO TO 31                                 00444000
      GO TO 32                                                   00445000
   31 CCNT = CCNT + 1                                            00446000
      TEMP = COEFF(CCNT + 1)                                     00447000
      J = 1                                                      00448000
      GO TO 37                                                   00449000
   38 J = J + 1                                                  00450000
   37 IF (J .LE. N) GO TO 35                                     00451000
      GO TO 36                                                   00452000
```

```
   35 YFV(J + 1) = (G(K + 1, J + 1) * TEMP) + YFV(J + 1)                00453000
      GO TO 38                                                          00454000
   36 GO TO 34                                                          00455000
   32 CONTINUE                                                          00456000
      RETURN                                                            00457000
      END                                                               00458000
C  32 BLOCK  6                                                          00459000

@PRT      FUNCTI
```

```
GALLAHER-L-J*TPF$.COMP
            LOGICAL FUNCTION COMP(N, EAV, ERV, Y, Z)                          00460000
            INTEGER N                                                         00461000
            DOUBLE PRECISION EAV,ERV,Y,Z                                      00462000
            DIMENSION EAV(100),ERV(100),Y(100),Z(100)                         00463000
            INTEGER J                                                         00464000
            DOUBLE PRECISION T1                                               00465000
            DEFINE FDBLE(I) = DBLE(FLOAT(I))                                  00466000
            J = 1                                                             00467000
            GO TO 42                                                          00468000
         43 J = J + 1                                                         00469000
         42 IF (J .LE. N) GO TO 40                                            00470000
            GO TO 41                                                          00471000
         40 T1 = ABS(Y(J + 1) - Z(J + 1))                                     00472000
            IF (T1 .LE. EAV(J + 1)) GO TO 45                                  00473000
            IF (T1 .LE. ERV(J + 1) * ABS(Z(J + 1))) GO TO 47                  00474000
            COMP = .FALSE.                                                    00475000
            GO TO 39                                                          00476000
         47 CONTINUE                                                          00477000
         46 CONTINUE                                                          00478000
         45 CONTINUE                                                          00479000
         44 GO TO 43                                                          00480000
         41 COMP = .TRUE.                                                     00481000
         39 CONTINUE                                                          00482000
            RETURN                                                            00483000
            END                                                               00484000
      C  39 BLOCK  7                                                          00485000

@HDG,N   X,M,66,6,3.S,PLEASE PUT STANDARD PAPER IN PR1 THANKS.
```

```
GALLAHER-L-J*TPF$.FUNCTI                                        00005000
              SUBROUTINE FUNCTI(N, X, Y, FV)                    00006000
              DOUBLE PRECISION X                                00007000
              INTEGER N                                         00008000
              DOUBLE PRECISION Y,FV                             00009000
              DIMENSION Y(100),FV(100)                          00010000
              DEFINE FDBLE(I) = DBLE(FLOAT(I))
              COMMON/NN/NN
              NN=NN+1                                           00011000
              RETURN                                            00012000
              END                                               00024000
       C      BLOCK  4


@PRT     .COMP
```

APPENDIX B

LISTING OF COEFFICIENTS

This appendix lists sets of coefficients for the four methods described in this report. Given here are:

first, the Adams coefficients, β and β^*, for q = 3 through 18;

then come the Butcher coefficients, A, B, a, b, α and β, for k = 1 through 6;

then come the Cowell coefficients P, C, and M, for m = 2 through 8;

last are the Shanks coefficients, α, β and γ for the formulas 4-4, 5-5, 6-6, 7-7, 7-9, 8-10, and 8-12. (The γs in each case are the last set of B's given).

```
  BETA  ( 2, 2) =          5/12
  BETA  ( 2, 1) =         -4/3
  BETA  ( 2, 0) =         23/12
 BETA*  ( 3, 3) =          1/24
 BETA*  ( 3, 2) =         -5/24
 BETA*  ( 3, 1) =         19/24
 BETA*  ( 3, 0) =          3/8
Q = 2 RATIO ( 4) =       270/19
  BETA  ( 3, 3) =         -3/8
  BETA  ( 3, 2) =         37/24
  BETA  ( 3, 1) =        -59/24
  BETA  ( 3, 0) =         55/24
 BETA*  ( 4, 4) =        -19/720
 BETA*  ( 4, 3) =         53/360
 BETA*  ( 4, 2) =        -11/30
 BETA*  ( 4, 1) =        323/360
 BETA*  ( 4, 0) =        251/720
Q = 3 RATIO ( 5) =       502/27
  BETA  ( 4, 4) =        251/720
  BETA  ( 4, 3) =       -637/360
  BETA  ( 4, 2) =        109/30
  BETA  ( 4, 1) =      -1387/360
  BETA  ( 4, 0) =       1901/720
 BETA*  ( 5, 5) =          3/160
 BETA*  ( 5, 4) =       -173/1440
 BETA*  ( 5, 3) =        241/720
 BETA*  ( 5, 2) =       -133/240
 BETA*  ( 5, 1) =       1427/1440
 BETA*  ( 5, 0) =         95/288
Q = 4 RATIO ( 6) =     19950/863
  BETA  ( 5, 5) =        -95/288
  BETA  ( 5, 4) =        959/480
  BETA  ( 5, 3) =      -3649/720
  BETA  ( 5, 2) =       4991/720
  BETA  ( 5, 1) =      -2641/480
  BETA  ( 5, 0) =       4277/1440
 BETA*  ( 6, 6) =       -863/60480
 BETA*  ( 6, 5) =        263/2520
```

```
   BETA* ( 6, 4) =            -6737/20160
   BETA* ( 6, 3) =              586/945
   BETA* ( 6, 2) =           -15487/20160
   BETA* ( 6, 1) =             2713/2520
   BETA* ( 6, 0) =            19087/60480
Q = 5 RATIO ( 7) =            38174/1375
    BETA ( 6, 6) =            19087/60480
    BETA ( 6, 5) =            -5603/2520
    BETA ( 6, 4) =           135713/20160
    BETA ( 6, 3) =           -10754/945
    BETA ( 6, 2) =           235183/20160
    BETA ( 6, 1) =           -18637/2520
    BETA ( 6, 0) =           198721/60480
   BETA* ( 7, 7) =              275/24192
   BETA* ( 7, 6) =           -11351/120960
   BETA* ( 7, 5) =             1537/4480
   BETA* ( 7, 4) =           -88547/120960
   BETA* ( 7, 3) =           123133/120960
   BETA* ( 7, 2) =            -4511/4480
   BETA* ( 7, 1) =           139849/120960
   BETA* ( 7, 0) =             5257/17280
Q = 6 RATIO ( 8) =          1103970/33953
    BETA ( 7, 7) =            -5257/17280
    BETA ( 7, 6) =            32863/13440
    BETA ( 7, 5) =          -115747/13440
    BETA ( 7, 4) =          2102243/120960
    BETA ( 7, 3) =          -296053/13440
    BETA ( 7, 2) =           242653/13440
    BETA ( 7, 1) =         -1152169/120960
    BETA ( 7, 0) =            16083/4480
   BETA* ( 8, 8) =           -33953/3628800
   BETA* ( 8, 7) =           156437/1814400
   BETA* ( 8, 6) =          -645607/1814400
   BETA* ( 8, 5) =          1573169/1814400
   BETA* ( 8, 4) =           -31457/22680
   BETA* ( 8, 3) =          2797679/1814400
   BETA* ( 8, 2) =         -2302297/1814400
   BETA* ( 8, 1) =          2233547/1814400
   BETA* ( 8, 0) =          1070017/3628800
Q = 7 RATIO ( 9) =          2140034/57281
```

```
   BETA ( 8, 8) =          1070017/3628800
   BETA ( 8, 7) =         -4832053/1814400
   BETA ( 8, 6) =         19416743/1814400
   BETA ( 8, 5) =        -45586321/1814400
   BETA ( 8, 4) =           862303/22680
   BETA ( 8, 3) =        -69927631/1814400
   BETA ( 8, 2) =         47738393/1814400
   BETA ( 8, 1) =        -21562603/1814400
   BETA ( 8, 0) =         14097247/3628800
  BETA* ( 9, 9) =             8183/1036800
  BETA* ( 9, 8) =          -116687/1451520
  BETA* ( 9, 7) =           335983/907200
  BETA* ( 9, 6) =          -462127/453600
  BETA* ( 9, 5) =          6755041/3628800
  BETA* ( 9, 4) =         -8641823/3628800
  BETA* ( 9, 3) =           200029/90720
  BETA* ( 9, 2) =         -1408913/907200
  BETA* ( 9, 1) =          9449717/7257600
  BETA* ( 9, 0) =            25713/89600
Q = 8 RATIO (10) =       137461698/3250433
   BETA ( 9, 9) =           -25713/89600
   BETA ( 9, 8) =         20884811/7257600
   BETA ( 9, 7) =         -2357683/181440
   BETA ( 9, 6) =         15788639/453600
   BETA ( 9, 5) =       -222386081/3628800
   BETA ( 9, 4) =        269181919/3628800
   BETA ( 9, 3) =        -28416361/453600
   BETA ( 9, 2) =          6648317/181440
   BETA ( 9, 1) =       -104995189/7257600
   BETA ( 9, 0) =          4325321/1036800
  BETA* (10,10) =         -3250433/479001600
  BETA* (10, 9) =          9071219/119750400
  BETA* (10, 8) =        -12318413/31933440
  BETA* (10, 7) =         23643791/19958400
  BETA* (10, 6) =        -21677723/8870400
  BETA* (10, 5) =          2227571/623700
  BETA* (10, 4) =        -33765029/8870400
  BETA* (10, 3) =         12051709/3991680
  BETA* (10, 2) =       -296725183/159667200
  BETA* (10, 1) =        164046413/119750400
```

```
   BETA* (10, 0) =          26842253/95800320
Q = 9 RATIO (11) =          53684506/1135053
    BETA (10,10) =          26842253/95800320
    BETA (10, 9) =         -52841941/17107200
    BETA (10, 8) =        2472634817/159667200
    BETA (10, 7) =        -186080291/3991680
    BETA (10, 6) =        2492064913/26611200
    BETA (10, 5) =         -82260679/623700
    BETA (10, 4) =        3539798831/26611200
    BETA (10, 3) =       -1921376209/19958400
    BETA (10, 2) =        1572737587/31933440
    BETA (10, 1) =       -2067948781/119750400
    BETA (10, 0) =        2132509567/479001600
   BETA* (11,11) =              4671/788480
   BETA* (11,10) =         -68928781/958003200
   BETA* (11, 9) =         384709327/958003200
   BETA* (11, 8) =         -87064741/63866880
   BETA* (11, 7) =         501289903/159667200
   BETA* (11, 6) =         -91910491/17740800
   BETA* (11, 5) =        1007253581/159667200
   BETA* (11, 4) =        -102212233/17740800
   BETA* (11, 3) =          36465037/9123840
   BETA* (11, 2) =         -99642413/45619200
   BETA* (11, 1) =        1374799219/958003200
   BETA* (11, 0) =           4777223/17418240
Q =10 RATIO (12) =      717300033450/13695779093
    BETA (11,11) =          -4777223/17418240
    BETA (11,10) =          30082309/9123840
    BETA (11, 9) =      -17410248271/958003200
    BETA (11, 8) =         923636629/15206400
    BETA (11, 7) =        -625551749/4561920
    BETA (11, 6) =       35183928883/159667200
    BETA (11, 5) =      -41290273229/159667200
    BETA (11, 4) =       35689892561/159667200
    BETA (11, 3) =      -15064372973/106444800
    BETA (11, 2) =       12326645437/191600640
    BETA (11, 1) =       -6477936721/319334400
    BETA (11, 0) =        4527766399/958003200
   BETA* (12,12) =      -13695779093/2615348736000
   BETA* (12,11) =        2724891251/39626496000
```

```
     BETA* (12,10) =        -30336027563/72648576000
     BETA* (12, 9) =        406332786317/261534873600
     BETA* (12, 8) =       -229882484333/58118860800
     BETA* (12, 7) =        529394045911/72648576000
     BETA* (12, 6) =         -4874320027/486486000
     BETA* (12, 5) =         84400835489/8072064000
     BETA* (12, 4) =       -485500845331/58118860800
     BETA* (12, 3) =       1346577425651/261534873600
     BETA* (12, 2) =       -551368413119/217945728000
     BETA* (12, 1) =          6595204069/4402944000
     BETA* (12, 0) =        703604254357/2615348736000
Q =11 RATIO (13) =        1407208508714/24466579093
      BETA (12,12) =        703604254357/2615348736000
      BETA (12,11) =       -169639834921/48432384000
      BETA (12,10) =       4588414555201/217945728000
      BETA (12, 9) =     -20232291373837/261534873600
      BETA (12, 8) =       2253957198793/11623772160
      BETA (12, 7) =      -2826800577631/8072064000
      BETA (12, 6) =        228133014533/486486000
      BETA (12, 5) =     -34266367915049/72648576000
      BETA (12, 4) =      20730767690131/58118860800
      BETA (12, 3) =     -10498491598103/52306974720
      BETA (12, 2) =       5963794194517/72648576000
      BETA (12, 1) =       -931781102989/39626496000
      BETA (12, 0) =      13064406523627/2615348736000
     BETA* (13,13) =          2224234463/475517952000
     BETA* (13,12) =        -69091417279/1046139494400
     BETA* (13,11) =          1636420501/3773952000
     BETA* (13,10) =      -4590817802567/2615348736000
     BETA* (13, 9) =       5124051955567/1046139494400
     BETA* (13, 8) =      -5797545653629/581188608000
     BETA* (13, 7) =       1335017017153/87178291200
     BETA* (13, 6) =      -7866111554491/435891456000
     BETA* (13, 5) =       9575580965507/581188608000
     BETA* (13, 4) =      -1748248003751/149448499200
     BETA* (13, 3) =      16964495066809/2615348736000
     BETA* (13, 2) =       -168235945379/58118860800
     BETA* (13, 1) =        741197087471/475517952000
     BETA* (13, 0) =        106364763817/402361344000
Q =12 RATIO (14) =        8296451577726/1322822840127
```

```
BETA (13,13) =      -106364763817/402361344000
BETA (13,12) =      6460951197929/1743565824000
BETA (13,11) =     -2102916211365l/871782912000
BETA (13,10) =      722265915994 9/74724249600
BETA (13, 9) =     -10320787460413/38745907200
BETA (13, 8) =     310429955875453/581188608000
BETA (13, 7) =    -3503793271 27877/435891456000
BETA (13, 6) =     134046425652457/145297152000
BETA (13, 5) =     -31457535950413/38745907200
BETA (13, 4) =     570885914358161/1046139494400
BETA (13, 3) =     -34412222659093/124540416000
BETA (13, 2) =      89541175419277/871782912000
BETA (13, 1) =    -140970750679621/5230697472000
BETA (13, 0) =          905730205/172204032
BETA* (14,14) =     -132282840127/31384184832000
BETA* (14,13) =      124922452271/1961511552000
BETA* (14,12) =    -14110480969927/31384184832000
BETA* (14,11) =      137855863153/7005398400 0
BETA* (14,10) =   -187504936597931/31384184832000
BETA* (14, 9) =     26159487787579/1961511552000
BETA* (14, 8) =   -236770944732449/10461394944000
BETA* (14, 7) =        38029005269/1277025750
BETA* (14, 6) =   -321201800274911/10461394944000
BETA* (14, 5) =     48869476129477/1961511552000
BETA* (14, 4) =    -71363886250691/4483454976000
BETA* (14, 3) =      3933201478249/490377888000
BETA* (14, 2) =   -102885148956217/31384184832000
BETA* (14, 1) =      3173185470929/1961511552000
BETA* (14, 0) =      1166309819657/4483454976000
Q =13 RATIO (15) =   18660957114512/274523709512
BETA (14,14) =       1166309819657/4483454976000
BETA (14,13) =       -696561442637/178319232000
BETA (14,12) =     85923647668 4231/31384184832000
BETA (14,11) =     -58262613384023/490377888000
BETA (14,10) =    1600835679073597/4483454976000
BETA (14, 9) =   -1544031478475483/1961511552000
BETA (14, 8) =   13760072112094753/10461394944000
BETA (14, 7) =      -2166615342637/1277025750
BETA (14, 6) =   17823675553313503/10461394944000
BETA (14, 5) =   -2614079370781733/1961511552000
```

```
      BETA (14, 4) =          2529891033708l429/31384184832000
      BETA (14, 3) =             =25990262345039/70053984000
      BETA (14, 2) =            3966421670215481/31384184832000
      BETA (14, 1) =             =600076791502 57/1961511552000
      BETA (14, 0) =              13325653738373/2414168064000
     BETA* (15,15) =                  2639651053/689762304000
     BETA* (15,14) =             =3867689367599/62768369664000
     BETA* (15,13) =              29219384284087/62768369664000
     BETA* (15,12) =            =137515713789319/62768369664000
     BETA* (15,11) =              64486158419069/8966909952000
     BETA* (15,10) =          =1096355235402331/62768369664000
     BETA* (15, 9) =            679781959848881/20922789888000
     BETA* (15, 8) =           =988788576755233/20922789888000
     BETA* (15, 7) =           1138313909617631/20922789888000
     BETA* (15, 6) =          =3129453071993581/62768369664000
     BETA* (15, 5) =           2285168598349733/62768369664000
     BETA* (15, 4) =           =189568380436867/8966909952000
     BETA* (15, 3) =            612744541065337/62768369664000
     BETA* (15, 2) =            =20999287611259/5706215424000
     BETA* (15, 1) =            105145058757073/62768369664000
     BETA* (15, 0) =                  25221445/98402304
 Q =14 RATIO (16) =          820494590698125O/111956703448001
      BETA (15,15) =                 =25221445/98402304
      BETA (15,14) =              36807182273689/8966909952000
      BETA (15,13) =          =1934443196892599/62768369664000
      BETA (15,12) =           9038571752734087/62768369664000
      BETA (15,11) =          =2674355537386529/5706215424000
      BETA (15,10) =          1010347879754906 9/8966909952000
      BETA (15, 9) =        =129930094104237331/62768369664000
      BETA (15, 8) =          62029181421198881/20922789888000
      BETA (15, 7) =         =70006862970773983/20922789888000
      BETA (15, 6) =          62487713370967631/20922789888000
      BETA (15, 5) =        =131963191940828581/62768369664000
      BETA (15, 4) =          72558117072259733/62768369664000
      BETA (15, 3) =          =4372481980074367/8966909952000
      BETA (15, 2) =            740161300731949/4828336128000
      BETA (15, 1) =          =2161567671248849/62768369664000
      BETA (15, 0) =            362555126427073/62768369664000
     BETA* (16,16) =          =111956703448001/320118685286400000
     BETA* (16,15) =           956906730268873/1600593426432000 0
```

```
BETA* (16,14) =        -171192511013729/355687428096000
BETA* (16,13) =         596904922428961/246245142528000
BETA* (16,12) =      -27389421430791451/3201186852864000
BETA* (16,11) =         5708273541404323/254062448640000
BETA* (16,10) =      -727845225633390409/16005934264320000
BETA* (16, 9) =       232085108601391937/3201186852864000
BETA* (16, 8) =         -42733352080603/463134672000
BETA* (16, 7) =       302240496916010687/3201186852864000
BETA* (16, 6) =     -1246285173964159159/16005934264320000
BETA* (16, 5) =        91914603656624011/1778437140480000
BETA* (16, 4) =       -12578861691928243/457312407552000
BETA* (16, 3) =        37519546987420243/3201186852864000
BETA* (16, 2) =        -1458231199032479/355687428096000
BETA* (16, 1) =        27707643610637623/16005934264320000
BETA* (16, 0) =         809298920353249/32011868528640000
Q =15 RATIO (17) =     16185978407066498/205804074290625
  BETA (16,16) =        809298920353249/32011868528640000
  BETA (16,15) =      -68846386581756617/16005934264320000
  BETA (16,14) =         942359269351333/27360571392000
  BETA (16,13) =     -551863998439384493/3201186852864000
  BETA (16,12) =      386778238886497951/640237370572800
  BETA (16,11) =    -2797406189209536629/1778437140480000
  BETA (16,10) =     7205576917796031023/2286562037760000
  BETA (16, 9) =     -324179886697104913/65330343936000
  BETA (16, 8) =        2879939505554213/463134672000
  BETA (16, 7) =    -3993885936674091251/640237370572800
  BETA (16, 6) =   80207429499737366711/16005934264320000
  BETA (16, 5) =    -5702855818380878219/1778437140480000
  BETA (16, 4) =     5173388005728297701/3201186852864000
  BETA (16, 3) =     -22133884200927593/35177877504000
  BETA (16, 2) =        2612634723678583/14227497123840
  BETA (16, 1) =     -55994879072429317/1455084933120000
  BETA (16, 0) =      14845854129333883/2462451425280000
BETA* (17,17) =                50188465/15613165568
BETA* (17,16) =      -3722582669836627/64023737057280000
BETA* (17,15) =        1136320750878589/2286562037760000
BETA* (17,14) =         -72974966880383/27360571392000
BETA* (17,13) =        8062612208040217/800296713216000
BETA* (17,12) =       -45532601008155413/1600593426432000
BETA* (17,11) =        15815540301010573/254062448640000
```

```
BETA* (17,10) =     =17284646348344409159/16005934264320000
BETA* (17, 9) =       964479921803293249/640237370572800 0
BETA* (17, 8) =      =40409465324546861/237124952064000
BETA* (17, 7) =      502364378756214437/3201186852864000
BETA* (17, 6) =     =1883042979819352909/16005934264320000
BETA* (17, 5) =       63645018657622943/889218570240000
BETA* (17, 4) =      =40193827387172 17/114328101888000
BETA* (17, 3) =      445168855133014 93/3201186852864000
BETA* (17, 2) =       =146702510065339/32335220736000
BETA* (17, 1) =     11432924370549111 7/64023737057280000
BETA* (17, 0) =        85455477715379/342372925440000
Q =16 RATIO (18) =  1275217911755514665 4/151711881512390095
BETA (17,17) =         =85455477715379/342372925440000
BETA (17,16) =        2460247368070567/547211427840000
BETA (17,15) =      =612172313896136299/16005934264320000
BETA (17,14) =       653581961828485643/3201186852864000
BETA (17,13) =       =454352730377724 7/5928123801600
BETA (17,12) =      171953928324833621 53/80029671321600 00
BETA (17,11) =     =7461931508849438072 3/16005934264320000
BETA (17,10) =      14237182892280945743/1778437140480000
BETA (17, 9) =     =7061743269929442873 7/64023737057280000
BETA (17, 8) =       3146403501110383511/256094948229120
BETA (17, 7) =     =19726972891423175089/1778437140480000
BETA (17, 6) =     12965008888534591777 3/16005934264320000
BETA (17, 5) =     =3802351602911608975 1/8002967132160000
BETA (17, 4) =        6550952547526506 1/29640619008000
BETA (17, 3) =       =51150187791975812 9/640237370572800
BETA (17, 2) =       49750571306468365 1/22865620377600 00
BETA (17, 1) =        =394724046586447 3/92386344960000
BETA (17, 0) =        57424625956493833/9146248151040000
BETA* (18,18) =        =2334028946344463/78601449494937600 0
BETA* (18,17) =        160835862139274 47/28383856762060800 0
BETA* (18,16) =     =87275129473084376 27/170303140572364800 00
BETA* (18,15) =      621611859048922958 9/212878925715456000 0
BETA* (18,14) =      =256626484009824989/218337359708160 00
BETA* (18,13) =      216021971923743546 1/608225502044160 00
BETA* (18,12) =       =230524780940305 07/27583922995200 0
BETA* (18,11) =    111229455689198952379/70959641905152000 0
BETA* (18,10) =   =2025977937601766635423/851515702861824000 0
BETA* (18, 9) =       735975656988412429/24946749107280 00
```

```
    BETA* (18, 8) =    -17050257677828924942 3/5676771352412 16000
```

```
K = 3  CAP A (2) =      200/128
K = 3  CAP B (2) =      300/128
K = 3   LC A (2) =     -297/31
K = 3   LC B (2) =    -2130/155
K = 3  ALPHA (2) =     -135/617
K = 3   BETA (2) =     -495/3085
K = 3  CAP A (3) =      153/128
K = 3  CAP B (3) =       45/128
K = 3   LC A (3) =     -212/31
K = 3   LC B (3) =     -309/155
K = 3  ALPHA (3) =      -31/617
K = 3   BETA (3) =      -39/3085
K = 3       LC B =      384/155
K = 3       BETA =     2304/3085
K = 3   BETA (0) =      465/3085
K = 4  CAP A (1) =  -629200/59049
K = 4  CAP B (1) =    96800/19683
K = 4   LC A (1) =     3796/61
K = 4   LC B (1) =   -75240/3355
K = 4  ALPHA (1) =     2248/3943
K = 4   BETA (1) =   129360/216865
K = 4  CAP A (2) =  -418176/59049
K = 4  CAP B (2) =   348480/19683
K = 4   LC A (2) =     2700/61
K = 4   LC B (2) =  -327096/3355
K = 4  ALPHA (2) =     1080/3943
K = 4   BETA (2) =    64152/216865
K = 4  CAP A (3) =   898425/59049
K = 4  CAP B (3) =   217800/19683
K = 4   LC A (3) =    -5204/61
K = 4   LC B (3) =  -210705/3355
K = 4  ALPHA (3) =      568/3943
K = 4   BETA (3) =    14190/216865
K = 4  CAP A (4) =   208000/59049
K = 4  CAP B (4) =    17600/19683
K = 4   LC A (4) =    -1231/61
K = 4   LC B (4) =   -17220/3355
K = 4  ALPHA (4) =       47/3943
K = 4   BETA (4) =      570/216865
K = 4       LC B =     6561/3355
```

```
K = 4       BETA =            118098/216865
K = 4   BETA (0) =             20130/216865
K = 5  CAP A (1) =          -9486400/531441
K = 5  CAP B (1) =           1185800/177147
K = 5   LC A (1) =            454750/4503
K = 5   LC B (1) =         -44017050/1386924
K = 5  ALPHA (1) =            230875/479327
K = 5   BETA (1) =          46719750/73816358
K = 5  CAP A (2) =         -24285184/531441
K = 5  CAP B (2) =           7589120/177147
K = 5   LC A (2) =           1181300/4503
K = 5   LC B (2) =        -328685280/1386924
K = 5  ALPHA (2) =             97000/479327
K = 5   BETA (2) =          32062800/73816358
K = 5  CAP A (3) =          12006225/531441
K = 5  CAP B (3) =          10672200/177147
K = 5   LC A (3) =           -558150/4503
K = 5   LC B (3) =        -472227525/1386924
K = 5  ALPHA (3) =            117500/479327
K = 5   BETA (3) =          13369125/73816358
K = 5  CAP A (4) =          20070400/531441
K = 5  CAP B (4) =           3449600/177147
K = 5   LC A (4) =           -965725/4503
K = 5   LC B (4) =        -153728400/1386924
K = 5  ALPHA (4) =             32375/479327
K = 5   BETA (4) =           1879500/73816358
K = 5  CAP A (5) =           2226400/531441
K = 5  CAP B (5) =            169400/177147
K = 5   LC A (5) =           -107672/4503
K = 5   LC B (5) =          -7573830/1386924
K = 5  ALPHA (5) =              1577/479327
K = 5   BETA (5) =             49170/73816358
K = 5       LC B =           2657205/1386924
K = 5       BETA =          39858075/73816358
K = 5   BETA (0) =           6934620/73816358
K = 6  CAP A (1) =        -630561008/23914845
K = 6  CAP B (1) =          13707848/1594323
K = 6   LC A (1) =          21459012/142985
K = 6   LC B (1) =       -1599082254/37433473
K = 6  ALPHA (1) =           1485084/3044563
```

```
K = 6   BETA (1) =          2514772260/3985332967
K = 6  CAP A (2) =        -3221344280/23914845
K = 6  CAP B (2) =          137078480/1594323
K = 6   LC A (2) =          110802000/142985
K = 6   LC B (2) =       -18173488410/37433473
K = 6  ALPHA (2) =             694875/3044563
K = 6   BETA (2) =          1660531950/3985332967
K = 6  CAP A (3) =        -1499295875/23914845
K = 6  CAP B (3) =          342696200/1594323
K = 6   LC A (3) =           52489000/142985
K = 6   LC B (3) =       -46096009575/37433473
K = 6  ALPHA (3) =             774000/3044563
K = 6   BETA (3) =           559924750/3985332967
K = 6  CAP A (4) =         3511928000/23914845
K = 6  CAP B (4) =          249233600/1594323
K = 6   LC A (4) =         -120483375/142985
K = 6   LC B (4) =       -33650498700/37433473
K = 6  ALPHA (4) =             121875/3044563
K = 6   BETA (4) =            -9906750/3985332967
K = 6  CAP A (5) =         1748450000/23914845
K = 6  CAP B (5) =           48956600/1594323
K = 6   LC A (5) =          -60172140/142985
K = 6   LC B (5) =        -6618347010/37433473
K = 6  ALPHA (5) =             -28812/3044563
K = 6   BETA (5) =           -18412020/3985332967
K = 6  CAP A (6) =          114738008/23914845
K = 6  CAP B (6) =            1612688/1594323
K = 6   LC A (6) =           -3951512/142985
K = 6   LC B (6) =         -218118054/37433473
K = 6  ALPHA (6) =              -2459/3044563
K = 6   BETA (6) =             -675290/3985332967
K = 6       LC B =           71744535/37433473
K = 6       BETA =          2152336050/3985332967
K = 6   BETA (0) =           374334730/3985332967
M = 2     P ( 0) =                2837/1440
M = 2     P ( 1) =               -2543/720
M = 2     P ( 2) =                  17/5
M = 2     P ( 3) =               -1201/720
M = 2     P ( 4) =                  95/288
M = 2     C ( 0) =                  95/288
```

M = 2	C (1) =	77/240
M = 2	C (2) =	-7/30
M = 2	C (3) =	73/720
M = 2	C (4) =	-3/160
M = 2	M (0) =	11/1440
M = 2	M (1) =	-41/720
M = 2	M (2) =	1/2
M = 2	M (3) =	41/720
M = 2	M (4) =	-11/1440
M = 3	P (0) =	11603/4480
M = 3	P (1) =	-104861/15120
M = 3	P (2) =	1344989/120960
M = 3	P (3) =	-20617/1890
M = 3	P (4) =	156551/24192
M = 3	P (5) =	-32371/15120
M = 3	P (6) =	5257/17280
M = 3	C (0) =	5257/17280
M = 3	C (1) =	6961/15120
M = 3	C (2) =	-66109/120960
M = 3	C (3) =	33/70
M = 3	C (4) =	-31523/120960
M = 3	C (5) =	1247/15120
M = 3	C (6) =	-275/24192
M = 3	M (0) =	-191/120960
M = 3	M (1) =	211/15120
M = 3	M (2) =	-7843/120960
M = 3	M (3) =	1/2
M = 3	M (4) =	7843/120960
M = 3	M (5) =	-211/15120
M = 3	M (6) =	191/120960
M = 4	P (0) =	3288521/1036800
M = 4	P (1) =	-40987771/3628800
M = 4	P (2) =	10219841/403200
M = 4	P (3) =	-135352319/3628800
M = 4	P (4) =	167287/4536
M = 4	P (5) =	-9839609/403200
M = 4	P (6) =	5393233/518400
M = 4	P (7) =	-9401029/3628800
M = 4	P (8) =	25713/89600
M = 4	C (0) =	25713/89600

M = 4	C (1) =	427487/725760
M = 4	C (2) =	-3498217/3628800
M = 4	C (3) =	500327/403200
M = 4	C (4) =	-6467/5670
M = 4	C (5) =	2616161/3628800
M = 4	C (6) =	-24019/80640
M = 4	C (7) =	263077/3628800
M = 4	C (8) =	-8183/1036800
M = 4	M (0) =	2497/7257600
M = 4	M (1) =	-1469/403200
M = 4	M (2) =	68119/3628800
M = 4	M (3) =	-252769/3628800
M = 4	M (4) =	1/2
M = 4	M (5) =	252769/3628800
M = 4	M (6) =	-68119/3628800
M = 4	M (7) =	1469/403200
M = 4	M (8) =	-2497/7257600
M = 5	P (0) =	3569763199/958003200
M = 5	P (1) =	-3966011741/239500800
M = 5	P (2) =	1695154823/35481600
M = 5	P (3) =	-1247363563/13305600
M = 5	P (4) =	4144305961/31933440
M = 5	P (5) =	-495967/3850
M = 5	P (6) =	2087883637/22809600
M = 5	P (7) =	-86656259/1900800
M = 5	P (8) =	76795519/5068800
M = 5	P (9) =	-144794759/47900160
M = 5	P (10) =	4777223/17418240
M = 5	C (0) =	4777223/17418240
M = 5	C (1) =	8089801/11404800
M = 5	C (2) =	-67283209/45619200
M = 5	C (3) =	14380247/5702400
M = 5	C (4) =	-517263181/159667200
M = 5	C (5) =	76561/24948
M = 5	C (6) =	-337204019/159667200
M = 5	C (7) =	41021471/39916800
M = 5	C (8) =	-107151937/319334400
M = 5	C (9) =	15813379/239500800
M = 5	C (10) =	-4671/788480
M = 5	M (0) =	-14797/191600640

M = 5 M (1) = 230371/239500800
M = 5 M (2) = -203257/35481600
M = 5 M (3) = 299093/13305600
M = 5 M (4) = -11639731/159667200
M = 5 M (5) = 1/2
M = 5 M (6) = 11639731/159667200
M = 5 M (7) = -299093/13305600
M = 5 M (8) = 203257/35481600
M = 5 M (9) = -230371/239500800
M = 5 M (10) = 14797/191600640
M = 6 P (0) = 733526173/172204032
M = 6 P (1) = -59344946587373/2615348736000
M = 6 P (2) = 104639289835229/1307674368000
M = 6 P (3) = -102675619234099/523069747200
M = 6 P (4) = 121844891963321/348713164800
M = 6 P (5) = -40318232897599/87178291200
M = 6 P (6) = 31975145483/69498000
M = 6 P (7) = -149831214658501/435891456000
M = 6 P (8) = 66393001798471/348713164800
M = 6 P (9) = -13247042672623/174356582400
M = 6 P (10) = 491703913717/23775897600
M = 6 P (11) = -9000055832083/2615348736000
M = 6 P (12) = 106364763817/402361344000
M = 6 C (0) = 106364763817/402361344000
M = 6 C (1) = 307515172843/373621248000
M = 6 C (2) = -2709005666077/1307674368000
M = 6 C (3) = 2309296746931/523069747200
M = 6 C (4) = -507942835493/69742632960
M = 6 C (5) = 4007043002299/435891456000
M = 6 C (6) = -2215533/250250
M = 6 C (7) = 2816016533573/435891456000
M = 6 C (8) = -175102023617/49816166400
M = 6 C (9) = 144690945961/104613949440
M = 6 C (10) = -486772076771/1307674368000
M = 6 C (11) = 160495253651/2615348736000
M = 6 C (12) = -2224234463/475517952000
M = 6 M (0) = 92427157/5230697472000
M = 6 M (1) = -132822967/523069747200
M = 6 M (2) = 2274524387/1307674368000
M = 6 M (3) = -4013113421/523069747200

```
M = 6     M ( 4) =        8855328071/348713164800
M = 6     M ( 5) =      -32793164357/435891456000
M = 6     M ( 6) =                1/2
M = 6     M ( 7) =       32793164357/435891456000
M = 6     M ( 8) =      -8855328071/348713164800
M = 6     M ( 9) =       4013113421/523069747200
M = 6     M (10) =      -2274524387/1307674368000
M = 6     M (11) =         132822967/523069747200
M = 6     M (12) =         -92427157/5230697472000
M = 7     P ( 0) =    29978675676307 3/62768369664000
M = 7     P ( 1) =   -1163613071553 61/3923023104000
M = 7     P ( 2) =   2586771998343187/20922789888000
M = 7     P ( 3) =  -356985279148297/980755776000
M = 7     P ( 4) =   1988442368270749/2510734786560
M = 7     P ( 5) =  -571195366208749/435891456000
M = 7     P ( 6) =   5010047970421097/2988969984000
M = 7     P ( 7) =     -2132356395131/1277025750
M = 7     P ( 8) =   9030884747790859/6974263296000
M = 7     P ( 9) =  -1216303284352 99/156920924160
M = 7     P (10) =   2006565473520353/5706215424000
M = 7     P (11) =    -3478073249303/29719872000
M = 7     P (12) =   1302216192466 27/4828336128000
M = 7     P (13) =    -2156804881129/560431872000
M = 7     P (14) =          25221445/98402304
M = 7     C ( 0) =          25221445/98402304
M = 7     C ( 1) =     3654051145153/3923023104000
M = 7     C ( 2) =  -17252734540140 1/62768369664000
M = 7     C ( 3) =     2292797894083/326918592000
M = 7     C ( 4) =  -886761467394133/62768369664000
M = 7     C ( 5) =     3496017827389/156920924160
M = 7     C ( 6) =  -19233843789310 9/6974263296000
M = 7     C ( 7) =         349581097/13030875
M = 7     C ( 8) =  -427489980816929/20922789888000
M = 7     C ( 9) =     5256082896499/435891456000
M = 7     C (10) =   -13579171932259/2510734786560
M = 7     C (11) =     1748809541047/980755776000
M = 7     C (12) =    -8530634387437/20922789888000
M = 7     C (13) =      226717570111/3923023104000
M = 7     C (14) =       -2639651053/689762304000
M = 7     M ( 0) =         -36740617/8966909952000
```

```
M = 7    M ( 1) =        87402869/1307674368000
M = 7    M ( 2) =      -1306229471/2510734786560
M = 7    M ( 3) =       2541742327/980755776000
M = 7    M ( 4) =    -197301894457/20922789888000
M = 7    M ( 5) =     108816780203/3923023104000
M = 7    M ( 6) =   -1610849246753/20922789888000
M = 7    M ( 7) =              1/2
M = 7    M ( 8) =    1610849246753/20922789888000
M = 7    M ( 9) =    -108816780203/3923023104000
M = 7    M (10) =     197301894457/20922789888000
M = 7    M (11) =      -2541742327/980755776000
M = 7    M (12) =       1306229471/2510734786560
M = 7    M (13) =        -87402869/1307674368000
M = 7    M (14) =         36740617/8966909952000
M = 8    P ( 0) =  482783778054538333/9146248151040000
M = 8    P ( 1) = -171249214157564497/4573124075520000
M = 8    P ( 2) =     469380177761711/2605768704000
M = 8    P ( 3) = -3961751682437057363/6402373705728000
M = 8    P ( 4) = 10188305820220195813/6402373705728000
M = 8    P ( 5) = -3746390185754199257/1185624760320000
M = 8    P ( 6) = 2259252039361835080l/4573124075520000
M = 8    P ( 7) = -39387573858057739199/6402373705728000
M = 8    P ( 8) =      11690920326343/1905904000
M = 8    P ( 9) = -31344919029592580161/6402373705728000
M = 8    P (10) =   904951790119037477 9/291016986624 0000
M = 8    P (11) =  -1840516046810912551/1185624760320000
M = 8    P (12) =    29365597024675091 9/492490285056000
M = 8    P (13) =    -1414911525807356 9/8314771046400 0
M = 8    P (14) =     8062298103159499/237124952064000
M = 8    P (15) =  -1359343838657402 33/32011868528640000
M = 8    P (16) =      854554777153 79/342372925440000
M = 8    C ( 0) =      854554777153 79/342372925440000
M = 8    C ( 1) =     736507566455411/711374856192000
M = 8    C ( 2) =   -2490947654982047/711374856192000
M = 8    C ( 3) =   6661524213176456 3/6402373705728000
M = 8    C ( 4) =  -1760777990262666 21/711374856192000
M = 8    C ( 5) =  1665410794991586 67/3556874280960000
M = 8    C ( 6) = -4534432488292555 63/6402373705728000
M = 8    C ( 7) =    2041798180308049 3/237124952064000
M = 8    C ( 8) =       -610091660201/7236479250
```

```
M = 8     C ( 9) =       4247098667237013l3/6402373705728000
M = 8     C (10) =     -1481533262278l2417/3556874280960000
M = 8     C (11) =         1332077054297011/64670441472000
M = 8     C (12) =      -5025477565721756З/6402373705728000
M = 8     C (13) =         1582902445233797/711374856192000
M = 8     C (14) =         -2858606305965l/64670441472000
M = 8     C (15) =        1758389297773001/32011868528640000
M = 8     C (16) =             -50188465/15613165568
M = 8     M ( 0) =           61430943169/6402373705728000
M = 8     M ( 1) =          -80168657839/457312407552000
M = 8     M ( 2) =             660473357/4311362764800
M = 8     M ( 3) =        -1096193632393/128047474114560
M = 8     M ( 4) =          4436541947807/128047474114560
M = 8     M ( 5) =       -13043845962023/1185624760320000
M = 8     M ( 6) =        949437300568649/32011868528640000
M = 8     M ( 7) =       -14334414125131/182924963020800
M = 8     M ( 8) =                      1/2
M = 8     M ( 9) =        14334414125131/182924963020800
M = 8     M (10) =      -949437300568649/32011868528640000
M = 8     M (11) =        13043845962023/1185624760320000
M = 8     M (12) =        -4436541947807/128047474114560
M = 8     M (13) =         1096193632393/128047474114560
M = 8     M (14) =            -660473357/4311362764800
M = 8     M (15) =           80168657839/457312407552000
M = 8     M (16) =          -61430943169/6402373705728000
  (4,4)   B(1,0)                    1/100
  (4,4)     A(1)                    1/100
  (4,4)   B(2,0)                -4278/245
  (4,4)   B(2,1)                 4425/245
  (4,4)     A(2)                      3/5
  (4,4)   B(3,0)              524746/8791
  (4,4)   B(3,1)             -532125/8791
  (4,4)   B(3,2)               16170/8791
  (4,4)     A(3)                      1/1
  (4,4)   B(4,0)            -179124/70092
  (4,4)   B(4,1)             200000/70092
  (4,4)   B(4,2)              40425/70092
  (4,4)   B(4,3)               8791/70092
(5,5)     B(1,0)                   1/1000
(5,5)       A(1)                   1/1000
```

(5,5)	B(2,0)	-447/10
(5,5)	B(2,1)	450/10
(5,5)	A(2)	3/10
(5,5)	B(3,0)	2684721/9568
(5,5)	B(3,1)	-2695500/9568
(5,5)	B(3,2)	17955/9568
(5,5)	A(3)	3/4
(5,5)	B(4,0)	-30872049/24219
(5,5)	B(4,1)	31009500/24219
(5,5)	B(4,2)	-146720/24219
(5,5)	B(4,3)	33488/24219
(5,5)	A(4)	1/1
(5,5)	B(5,0)	105/1134
(5,5)	B(5,1)	0/1134
(5,5)	B(5,2)	500/1134
(5,5)	B(5,3)	448/1134
(5,5)	B(5,4)	81/1134
(6,6)	B(1,0)	1/300
(6,6)	A(1)	1/300
(6,6)	B(2,0)	-29/5
(6,6)	B(2,1)	30/5
(6,6)	A(2)	1/5
(6,6)	B(3,0)	323/5
(6,6)	B(3,1)	-330/5
(6,6)	B(3,2)	10/5
(6,6)	A(3)	3/5
(6,6)	B(4,0)	-510104/810
(6,6)	B(4,1)	521640/810
(6,6)	B(4,2)	-12705/810
(6,6)	B(4,3)	1925/810
(6,6)	A(4)	14/15
(6,6)	B(5,0)	-417923/77
(6,6)	B(5,1)	427350/77
(6,6)	B(5,2)	-10605/77
(6,6)	B(5,3)	1309/77
(6,6)	B(5,4)	-54/77
(6,6)	A(5)	1/1
(6,6)	B(6,0)	198/3696
(6,6)	B(6,1)	0/3696
(6,6)	B(6,2)	1225/3696

(6,6)	B(6,3)	1540/3696
(6,6)	B(6,4)	810/3696
(6,6)	B(6,5)	-77/3696
(7,7)	B(1,0)	1/192
(7,7)	A(1)	1/192
(7,7)	B(2,0)	-15/6
(7,7)	B(2,1)	16/6
(7,7)	A(2)	1/6
(7,7)	B(3,0)	4867/186
(7,7)	B(3,1)	-5072/186
(7,7)	B(3,2)	298/186
(7,7)	A(3)	1/2
(7,7)	B(4,0)	-19995/31
(7,7)	B(4,1)	20896/31
(7,7)	B(4,2)	-1025/31
(7,7)	B(4,3)	155/31
(7,7)	A(4)	1/1
(7,7)	B(5,0)	-469805/5022
(7,7)	B(5,1)	490960/5022
(7,7)	B(5,2)	-22736/5022
(7,7)	B(5,3)	5580/5022
(7,7)	B(5,4)	186/5022
(7,7)	A(5)	5/6
(7,7)	B(6,0)	914314/2604
(7,7)	B(6,1)	-955136/2604
(7,7)	B(6,2)	47983/2604
(7,7)	B(6,3)	-6510/2604
(7,7)	B(6,4)	-558/2604
(7,7)	B(6,5)	2511/2604
(7,7)	A(6)	1/1
(7,7)	B(7,0)	14/300
(7,7)	B(7,1)	0/300
(7,7)	B(7,2)	81/300
(7,7)	B(7,3)	110/300
(7,7)	B(7,4)	0/300
(7,7)	B(7,5)	81/300
(7,7)	B(7,6)	14/300
(7,9)	B(1,0)	2/9
(7,9)	A(1)	2/9
(7,9)	B(2,0)	1/12

(7,9)	B(2,1)	3/12
(7,9)	A(2)	1/3
(7,9)	B(3,0)	1/8
(7,9)	B(3,1)	0/8
(7,9)	B(3,2)	3/8
(7,9)	A(3)	1/2
(7,9)	B(4,0)	23/216
(7,9)	B(4,1)	0/216
(7,9)	B(4,2)	21/216
(7,9)	B(4,3)	-8/216
(7,9)	A(4)	1/6
(7,9)	B(5,0)	-4136/729
(7,9)	B(5,1)	0/729
(7,9)	B(5,2)	-13584/729
(7,9)	B(5,3)	5264/729
(7,9)	B(5,4)	13104/729
(7,9)	A(5)	8/9
(7,9)	B(6,0)	105131/151632
(7,9)	B(6,1)	0/151632
(7,9)	B(6,2)	302016/151632
(7,9)	B(6,3)	-107744/151632
(7,9)	B(6,4)	-284256/151632
(7,9)	B(6,5)	1701/151632
(7,9)	A(6)	1/9
(7,9)	B(7,0)	-775229/1375920
(7,9)	B(7,1)	0/1375920
(7,9)	B(7,2)	-2770950/1375920
(7,9)	B(7,3)	1735136/1375920
(7,9)	B(7,4)	2547216/1375920
(7,9)	B(7,5)	81891/1375920
(7,9)	B(7,6)	328536/1375920
(7,9)	A(7)	5/6
(7,9)	B(8,0)	23569/251888
(7,9)	B(8,1)	0/251888
(7,9)	B(8,2)	-122304/251888
(7,9)	B(8,3)	-20384/251888
(7,9)	B(8,4)	695520/251888
(7,9)	B(8,5)	-99873/251888
(7,9)	B(8,6)	-466560/251888
(7,9)	B(8,7)	241920/251888

(7,9)	A(8)	1/1
(7,9)	B(9,0)	110201/2140320
(7,9)	B(9,1)	0/2140320
(7,9)	B(9,2)	0/2140320
(7,9)	B(9,3)	767936/2140320
(7,9)	B(9,4)	635040/2140320
(7,9)	B(9,5)	-59049/2140320
(7,9)	B(9,6)	-59049/2140320
(7,9)	B(9,7)	635040/2140320
(7,9)	B(9,8)	110201/2140320
(8,10)	B(1,0)	4/27
(8,10)	A(1)	4/27
(8,10)	B(2,0)	1/18
(8,10)	B(2,1)	3/18
(8,10)	A(2)	2/9
(8,10)	B(3,0)	1/12
(8,10)	B(3,1)	0/12
(8,10)	B(3,2)	3/12
(8,10)	A(3)	1/3
(8,10)	B(4,0)	1/8
(8,10)	B(4,1)	0/8
(8,10)	B(4,2)	0/8
(8,10)	B(4,3)	3/8
(8,10)	A(4)	1/2
(8,10)	B(5,0)	13/54
(8,10)	B(5,1)	0/54
(8,10)	B(5,2)	-27/54
(8,10)	B(5,3)	42/54
(8,10)	B(5,4)	8/54
(8,10)	A(5)	2/3
(8,10)	B(6,0)	389/4320
(8,10)	B(6,1)	0/4320
(8,10)	B(6,2)	-54/4320
(8,10)	B(6,3)	966/4320
(8,10)	B(6,4)	-824/4320
(8,10)	B(6,5)	243/4320
(8,10)	A(6)	1/6
(8,10)	B(7,0)	-231/20
(8,10)	B(7,1)	0/20
(8,10)	B(7,2)	81/20

(8,10)	B(7,3)	-1164/20
(8,10)	B(7,4)	656/20
(8,10)	B(7,5)	-122/20
(8,10)	B(7,6)	800/20
(8,10)	A(7)	1/1
(8,10)	B(8,0)	-127/288
(8,10)	B(8,1)	0/288
(8,10)	B(8,2)	18/288
(8,10)	B(8,3)	-678/288
(8,10)	B(8,4)	456/288
(8,10)	B(8,5)	-9/288
(8,10)	B(8,6)	576/288
(8,10)	B(8,7)	4/288
(8,10)	A(8)	5/6
(8,10)	B(9,0)	1481/820
(8,10)	B(9,1)	0/820
(8,10)	B(9,2)	-81/820
(8,10)	B(9,3)	7104/820
(8,10)	B(9,4)	-3376/820
(8,10)	B(9,5)	72/820
(8,10)	B(9,6)	-5040/820
(8,10)	B(9,7)	-60/820
(8,10)	B(9,8)	720/820
(8,10)	A(9)	1/1
(8,10)	B(10,0)	41/840
(8,10)	B(10,1)	0/840
(8,10)	B(10,2)	0/840
(8,10)	B(10,3)	27/840
(8,10)	B(10,4)	272/840
(8,10)	B(10,5)	27/840
(8,10)	B(10,6)	216/840
(8,10)	B(10,7)	0/840
(8,10)	B(10,8)	216/840
(8,10)	B(10,9)	41/840
(8,12)	B(1, 0)	1/9
(8,12)	A(1)	1/9
(8,12)	B(2, 0)	1/24
(8,12)	B(2, 1)	3/24
(8,12)	A(2)	1/6
(8,12)	B(3, 0)	1/16

(8,12)	B(3, 1)	0/16
(8,12)	B(3, 2)	3/16
(8,12)	A(3)	1/4
(8,12)	B(4, 0)	29/500
(8,12)	B(4, 1)	0/500
(8,12)	B(4, 2)	33/500
(8,12)	B(4, 3)	-12/500
(8,12)	A(4)	1/10
(8,12)	B(5, 0)	33/972
(8,12)	B(5, 1)	0/972
(8,12)	B(5, 2)	0/972
(8,12)	B(5, 3)	4/972
(8,12)	B(5, 4)	125/972
(8,12)	A(5)	1/6
(8,12)	B(6, 0)	-21/36
(8,12)	B(6, 1)	0/36
(8,12)	B(6, 2)	0/36
(8,12)	B(6, 3)	76/36
(8,12)	B(6, 4)	125/36
(8,12)	B(6, 5)	-162/36
(8,12)	A(6)	1/2
(8,12)	B(7, 0)	-30/243
(8,12)	B(7, 1)	0/243
(8,12)	B(7, 2)	0/243
(8,12)	B(7, 3)	-32/243
(8,12)	B(7, 4)	125/243
(8,12)	B(7, 5)	0/243
(8,12)	B(7, 6)	99/243
(8,12)	A(7)	2/3
(8,12)	B(8, 0)	1175/324
(8,12)	B(8, 1)	0/324
(8,12)	B(8, 2)	0/324
(8,12)	B(8, 3)	-3456/324
(8,12)	B(8, 4)	-6250/324
(8,12)	B(8, 5)	8424/324
(8,12)	B(8, 6)	242/324
(8,12)	B(8, 7)	-27/324
(8,12)	A(8)	1/3
(8,12)	B(9, 0)	293/324
(8,12)	B(9, 1)	0/324

(8,12)	B(9, 2)	0/324
(8,12)	B(9, 3)	-852/324
(8,12)	B(9, 4)	-1375/324
(8,12)	B(9, 5)	1836/324
(8,12)	B(9, 6)	-118/324
(8,12)	B(9, 7)	162/324
(8,12)	B(9, 8)	324/324
(8,12)	A(9)	5/6
(8,12)	B(10, 0)	1303/1620
(8,12)	B(10, 1)	0/1620
(8,12)	B(10, 2)	0/1620
(8,12)	B(10, 3)	-4260/1620
(8,12)	B(10, 4)	-6875/1620
(8,12)	B(10, 5)	9990/1620
(8,12)	B(10, 6)	1030/1620
(8,12)	B(10, 7)	0/1620
(8,12)	B(10, 8)	0/1620
(8,12)	B(10, 9)	162/1620
(8,12)	A(10)	5/6
(8,12)	B(11, 0)	-8595/4428
(8,12)	B(11, 1)	0/4428
(8,12)	B(11, 2)	0/4428
(8,12)	B(11, 3)	30720/4428
(8,12)	B(11, 4)	48750/4428
(8,12)	B(11, 5)	-66096/4428
(8,12)	B(11, 6)	378/4428
(8,12)	B(11, 7)	-729/4428
(8,12)	B(11, 8)	-1944/4428
(8,12)	B(11, 9)	-1296/4428
(8,12)	B(11,10)	3240/4428
(8,12)	A(11)	1/1
(8,12)	B(12, 0)	41/840
(8,12)	B(12, 1)	0/840
(8,12)	B(12, 2)	0/840
(8,12)	B(12, 3)	0/840
(8,12)	B(12, 4)	0/840
(8,12)	B(12, 5)	216/840
(8,12)	B(12, 6)	272/840
(8,12)	B(12, 7)	27/840
(8,12)	B(12, 8)	27/840

(8,12)	B(12, 9)	36/840
(8,12)	B(12,10)	180/840
(8,12)	B(12,11)	41/840