RATIONAL FUNCTION EXTRAPOLATION

by

Richard Joseph Gaydos

Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in the

Mathematics

Program

J. Douglas Faires — August 7, 1980

Advisor — Date

Dean Rand — August 18, 1980

Dean of the Graduate School — Date

YOUNGSTOWN STATE UNIVERSITY

August, 1980

ABSTRACT

RATIONAL FUNCTION EXTRAPOLATION

Richard Joseph Gaydos

Master of Science

Youngstown State University, 1980

In this thesis, I examine rational function extrapolation to solve the initial value problem in ordinary differential equations. Significant historical achievements leading up to rational function extrapolation are noted, and a thorough study is made of H. G. Hussel's computer implementation of the method. That Watfiv program is then compared to an Adams Predictor Corrector program over a series of problems with an error tolerance of .0001. The overall results of the computer runs show that although computer costs are at times more expensive, rational function extrapolation is more accurate.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to Dr. J. Douglas Faires for his help and technical suggestions, which aided me in the preparation of this thesis. Also, I would like to thank Pam Spon for her help in typing the paper.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

There are a multitude of methods that can be used to approximate the solution to the system of initial value problems

$$\begin{aligned} y_1' &= f_1(x, y_1, \ldots, y_n),\ y_1(a) = c_1 \\ &\vdots \\ y_n' &= f_n(x, y_1, \ldots, y_n),\ y_n(a) = c_n \end{aligned}$$

$$\text{for } a \leq x \leq b \qquad (1)$$

at specific points in the interval [a,b]. The solutions at other points can then be approximated by interpolation. Most notable are the one step Runge-Kutta and Runge-Kutta-Fehlberg methods, multistep methods such as the explicit Adams-Bashforth and implicit Adams-Moulton methods, the Adams Predictor Corrector method, and the extrapolation methods. Extrapolation methods use polynomials or rational functions.

When attempting to solve (1) by the numerical methods, one should be aware of what stiff systems of equations are. A stiff system of equations has no unstable solution component, that is, no eigenvalue with large positive real part , and at least some very stable component, that is, at least one eigenvalue with a large

negative real part. Stiff systems can cause severe problems in the above methods because rounding error in the computations may build up rapidly, producing erroneous results. However, the main concern of this paper is nonstiff equations without discontinuities that have no large oscillations in the starting interval, so that no additional features have to be built into the methods to solve (1).

The order of a particular method refers to the amount of accuracy that one expects to obtain from the method. For example, Runge-Kutta Four is on $O(h^4)$ method, that is, one expects the method to produce solutions accurate to some multiple of the step size h raised to the fourth power. Associated with each Runge-Kutta, Runge-Kutta-Fehlberg, Adams-Bashforth, Adams-Moulton, and Adams Predictor Corrector formula is some fixed order, so a fixed order method basically employs only one of the above named formulas. On the other hand, a variable-order method usually employs a series of formulas in such a way that if a solution to (1) relative to a particular error tolerance is not produced in a sufficient time, then a higher order formula is used. It will be shown that extrapolation can be a variable-order method.

Exhaustive tests to compare the methods were carried out at the University of Toronto, from which two basic conclusions were reached. The first is that variable-order methods are best to solve (1) if the

equations are not stiff. The other conclusion is that rational function extrapolation is the best overall method if function evaluations are not too costly, and that variable-order Adams methods are preferable if function evaluations are costly, since they take less function evaluations.[1] The break even point for a costly function is approximately twenty five arithmetic operations per evaluation.[2]

The actual implementation of the Burlirsch and Stoer rational function extrapolation algorithm that was tested in Toronto is called DESUB, and is due to P.A. Fox.[3] However, in a subsequent series of tests, it was found that DESUB had some problems, and that there was a better version of the Burlirsch and Stoer algorithm which was less sensitive to the choice of starting step size, and which was not as biased to the higher-order methods, hence more accurate at lower error tolerances.[4] This better version was written by H. G. Hussels and is

---

[1] T. E. Hull, et al, "Comparing Numerical Methods for Ordinary Differential Equations," SIAM Journal of Numerical Analysis, IX (4, December 1972), 605.

[2] Hull, 614.

[3] P. A. Fox, "DESUB: Integration of a First Order System of Ordinary Differential Equations," Mathematical Software, ed. by John Rice (New York: Academic Press, 1971), pp. 486-507.

[4] Wayne H. Enright and T. E. Hull, "Test Results on Initial Value Methods for Non-Stiff Ordinary Differential Equations," SIAM Journal of Numerical Analysis, XIII (6, December 1976), 959.

called DIFSY1.[5]

In a separate set of tests, L. F. Shampine, H. A. Watts, and S. M. Davenport reached the same conclusion, that the best methods to solve initial value problems for nonstiff ordinary differential equations are the extrapolation and variable-order Adams methods.[6] They were also in agreement about Hussels' program, in regards to DIFSY1's improvement over the original algorithm's behavior with respect to the choice of step size.[7] The results of all these tests were summarized by J. D. Lambert.[8]

A brief summary of the key historical extrapolation results is presented in Chapter II. In Chapter III, DIFSY1 is examined in detail. Chapter IV explains how DIFSY1 was compared to an Adams Predictor Corrector program, and the test results appear in Chapter V, while the conclusions appear in Chapter VI.

---

[5]H. G. Hussels, "Schrittweitenteuerung bei der Integration gewöhnlicher Differentialgleichungen mit Extrapolationverfahren," (unpublished M. Sc. Thesis, Universität Köln, 1973).

[6]L. F. Shampine, H. A. Watts, and S. M. Davenport, "Solving Nonstiff Ordinary Differential Equations - The State of The Art," SIAM Review, XVII (3, July 1976), 377.

[7]L. F. Shampine, H. A. Watts, and S. M. Davenport, 382.

[8]J. D. Lambert, "The Initial Value Problem for Ordinary Differential Equations," The State of The Art in Numerical Analysis, ed. by D. A. H. Jacobs (London: Academic Press, 1977), p. 479.

CHAPTER II

HISTORY OF EXTRAPOLATION METHODS

In this section, I will look at some of the key extrapolation ideas. Then the Bulirsch and Stoer rational function extrapolation method is examined. It is often possible to approximate some unknown quantity B, by a calculatable quantity $T(h)$, depending on some parameter $h > 0$, so that the limit as h approaches zero of $T(h)$ is B. For extrapolation to proceed, there must exist constants $a_1, a_2, \ldots, c_1, c_2, \ldots, g_1, g_2, \ldots$, and H so that for $j = 1, 2, 3, \ldots$ and for $h < H$, $T(h) = B + a_1 h^{g_1} + \ldots + a_j h^{g_j} + E(h)$, where $|E(h)| < c_j h^{g_{j+1}}$ is a bound on the truncation error. When this is possible, it is said that $T(h)$ admits an asymptotic expansion for h approaching zero. The idea of extrapolation is to speed the convergence of $T(h)$ to the unknown B by combining approximations obtained by some discretization method at two different step sizes to eliminate terms in the asymptotic expansion. D. C. Joyce cites the following example.[9] To find an approximation to the familiar geometric quantity π, one can examine two sequences $I(h)$ and $U(h)$, where $I(h)$

---

[9] D. C. Joyce, "Survey of Extrapolation Processes in Numerical Analysis," SIAM Review, XIII (4, October 1971), 436-7.

represents the perimeter of an n-sided polygon inscribed in a unit circle, and U(h) represents the perimeter of an n-sided polygon circumscribed around a unit circle, with $h = \frac{1}{n}$. Geometric arguments show that

$$I(h) = 2 \frac{\sin \pi h}{h}$$

and

$$U(h) = 2 \frac{\tan \pi h}{h}.$$

Let $M(h) = \frac{1}{2} I(h)$ and $N(h) = \frac{1}{2} U(h)$. Since the Taylor series expansions for sin x and tan x are respectively

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

and

$$\tan x = x + \frac{x^3}{3} + \frac{2x^5}{15} + \frac{17x^7}{315} + \ldots,$$

substitution gives

$$M(h) = \pi - \frac{\pi^3 h^2}{3!} + \frac{\pi^5 h^4}{5!} - \frac{\pi^7 h^6}{7!} + \ldots$$

and

$$N(h) = \pi + \frac{\pi^3 h^2}{3} + \frac{2\pi^5 h^4}{15} + \frac{17\pi^7 h^6}{315} + \ldots.$$

It can also be shown that M(h) is an increasing sequence of lower bounds for π, and that N(h) is a decreasing sequence of upper bounds for π. That M(h) admits an asymptotic expansion as h approaches zero can be seen since

$$A_j = \frac{(-1)^j \pi^{2j+1}}{(2j+1)!}, \text{ for } j = 0, 1, 2, \ldots$$

and

$$g_j = 2j, \text{ for } j = 1, 2, 3, \ldots$$

A similar result can be shown for N(h). In 1654, Huygens suggested using two new sequences

$$S(h) = \frac{4M(h) - M(2h)}{3}$$

and

$$V(h) = \frac{2N(h) + M(2h)}{3} .$$

He showed that

$$S(h) = \pi - \frac{\pi^5 h^4}{30} + \frac{\pi^7 h^6}{252} - \ldots$$

and that

$$V(h) = \pi + \frac{2\pi^5 h^4}{15} + \frac{2\pi^7 h^6}{63} + \ldots,$$

from which the reader can see that the h^2 terms have been eliminated from the sequences, making the error $O(h^4)$ as opposed to $O(h^2)$. These Huygens sequences are examples of extrapolation formulas. They accelerate the convergence of the sequences toward π, and more than double the accuracy. [10] L.F. Richardson produced his deferred approach to the limit theory to solve (1) in 1910. He showed that the h^2 term in the asymptotic expansion could be eliminated by combining approximations obtained from two different step sizes h_0 and h_1, by means of the formula

$$T(h_0,h_1) = \frac{h_1^2 T(h_0) - h_0^2 T(h_1)}{h_1^2 - h_0^2} ,$$

---

[10] D. C. Joyce, 437.

if central difference formulas were used in the descretization method to find T(h), since the asymptotic expansion would then contain only even powers of h.[11] In 1912, C. Runge used

$$\frac{T(2h) - T(h)}{15}$$

to estimate the error in T(h) when solving initial value problems.[12]

W. Romberg dealt with linear iterative extrapolation to integrate a function over an interval, in 1955, which dealt with discretization methods based on the trapezoidal rule, T(h), and the midpoint rule, U(h).[13] Let T_0^i be the trapezoidal value $T(h_i)$ and U_0^i is the midpoint value $U(h_i)$. If

$$T_m^i = T(h_i, h_{i+1}, \ldots, h_{i+m}),$$

and

$$U_m^i = U(h_i, h_{i+1}, \ldots, h_{i+m}),$$

where

$$h_{i+s} = \frac{h_i}{2^s}, \text{ for } s = 1, 2, \ldots, m \text{ and}$$

$$\text{for } i = 0, 1, 2, \ldots,$$

then we have for i = 0, 1, 2, ... and m = 1, 2, 3, ...

---

[11] D. C. Joyce, 442.

[12] D. C. Joyce, 442.

[13] D. C. Joyce, 452-3.

$$T_m^i = T_{m-1}^{i+1} + \frac{T_{m-1}^{i+1} - T_{m-1}^i}{4^m - 1} \tag{2}$$

and for i = 1, 2, 3, and m = 0, 1, 2, ...

$$U_m^i = U_{m-1}^{i+1} + \frac{U_{m-1}^{i+1} - U_{m-1}^i}{4^m - 1} \tag{3}$$

and

$$T_m^i = \frac{T_m^{i-1} + U_m^{i-1}}{2} . \tag{4}$$

With these formulas, Romberg had developed a method to accelerate convergence for the numerical integration problem, if the values T_0^0 and U_0^i for i = 0, 1, 2, ... were known. The T_i^0 for i = 1, 2, ..., were then calculated by (2), the T_j^i for i = 1, 2, ... and j = 0, 1, 2, ... were calculated by (4) , and the U_j^i for i = 0, 1, 2, ... and j = 1, 2, 3, ... were calculated by (3). As the reader will shortly see, (2), (3), and (4) are somewhat similar to the formulas Bulirsch and Stoer developed for rational function extrapolation.

In 1961, Bauer showed that T_m^0 converges superlinearly as $m \to \infty$. The difference in absolute value between T_m^0 and B, the value of the integral of the function f(x) over (0,1), is a constant divided by $2^{m(m+1)}$, if f(x) is 2^{m+2} times continuously differentiable on (0,1), and if $h_k = \frac{1}{2^k}$ is the stepsize sequence used.[14] Therefore, one gets superlinear convergence since

$$\lim_{m\to\infty} \left| \frac{T_{m+1}^0 - B}{T_m^0 - B} \right| = \lim_{m\to\infty} \frac{c_1 / 2^{(m+1)(m+2)}}{c_2 / 2^{m(m+1)}} = \lim_{m\to\infty} \frac{c_1/c_2}{4^{(m+1)}} = 0.$$

---

[14] D. C. Joyce, 456.

Roland Bulirsch generalized (2) to

$$T_m^i = T_{m-1}^{i+1} + \frac{T_{m-1}^{i+1} + T_{m-1}^{i}}{\left(\frac{h_i}{h_{i+m}}\right)^2 - 1}$$

$$= \frac{h_i^2 \, T_{m-1}^{i+1} - h_{i+m}^2 \, T_{m-1}^{i}}{h_i^2 - h_{i+m}^2}.^{15} \qquad (5)$$

He and Josef Stoer formalized this idea of polynomial extrapolation to solve (1) in 1964, stating that for a discretization method T(h) with asymptotic expansion as described above, a polynomial

$$P_m^i(h) = B + b_1 h^{g_1} + \ldots + b_m h^{g_m}$$

can be defined such that

$$P_m^i(h_k) = T(h_k), \text{ for } k = i, i+1, \ldots, i+m.$$

Then T(0) is approximated by $P_m^i(0) = B$, where T_m^i is defined iteratively as $T(h_i, h_{i+1}, \ldots, h_{i+m}) = B$.[16] If $G_I = GI$, then (5) generalizes to

$$T_m^i = \frac{T_{m-1}^{i+1} - T_{m-1}^{i}}{\left(\frac{h_i}{h_{i+m}}\right)^G - 1}.$$

After extensive testing, they concluded that rational

---

[15] D. C. Joyce, 458.

[16] D. C. Joyce, 464.

function extrapolation was preferable.[17]

A rational function is the quotient of two polynomials; that is, for rational function R(h),

$$R(h) = \frac{P(h)}{Q(h)}$$

where P(h) and Q(h) are polynomials in h. Suppose

$$R_k(h) = \frac{P_k(h)}{Q_k(h)}$$

$$= \frac{P_0 + P_1 h + \ldots + P_M h^M}{Q_0 + Q_1 h + \ldots + Q_N h^N},$$

where M is $[k/2]$ and N is $[(k + 1)/2]$, where $[x]$ indicates the greatest integer not exceeding x, with the stipulation that $R_k(h_j) = Y_j$, for $j = 0, 1, \ldots, k$, and for $k = 0, 1, \ldots, L$, where the Y_j are the approximations obtained from some discretization method. M and N are chosen this way to allow the sequence (0,0), (0,1), (1,1), (1,2), ... of rational functions to be used. The reader will shortly see that this gives the desired extrapolation formulas. In order to find these formulas, given $R_k(h)$ as defined above, let us find a similar rational function for $R_{k+1}(h)$. Clearly

$$T_k(h_j) = P_k(h_j) - Y_j Q_k(h_j) = 0,$$

for all h_j, $j = 0, 1, \ldots, k$ by the way $R_k(h_j)$ was defined.

---

[17] Roland Bulirsch and Josef Stoer, "Numerical Treatment of Ordinary Differential Equations by Extrapolation Methods," Numerische Mathematik, VIII (1966),1.

As a candidate for $R_{k+1}(h_j)$, let us look at

$$T_{k+1}(h) = AT_k(h) + B(h)T_{k-1}(h) \tag{6}$$

Clearly, (6) is zero for all h_j, $j = 0, 1, \ldots, k$, and if we stipulate that $B(h_k) = 0$, then $T_{k+1}(h_k) = 0$. Also, we want to force $T_{k+1}(h_{k+1}) = 0$, so it must be the case that

$$T_{k+1}(h_{k+1}) = AT_k(h_{k+1}) + B(h_{k+1})T_{k-1}(h_{k+1}) = 0. \tag{7}$$

We want T_{k+1} to be a polynomial of degree $[(k + 2)/2]$ in h, and linear in y, so that we can obtain

$$T_{k+1}(h) = P_{k+1}(h) + yQ_{k+1}(h),$$

in order to obtain a rational function. Upon reexamining (6), since T_k is a polynomial of degree $[(k + 1)/2]$ then if A is a constant, AT_k is still a polynomial of that degree. If B(h) is linear in h, since $T_{k-1}(h)$ is of degree $[k/2]$ in h, then $B(h)T_{k-1}(h)$ is of degree $[(k + 2)/2]$. There are two cases to examine, k being odd or even. If k is odd, and if $[k/2] = c$, then $[(k + 1)/2] = [(k + 2)/2] = c + 1$, and the degree of $B(h)T_{k-1}(h)$ is $c + 1$. If k is even, and if $[k/2] = c$, then $[(k+1)/2] = c$ and $[(k+2)/2] = c + 1 =$ degree $(B(h)T_{k-1}(h))$. If we let

$$A = \frac{(h - h_{k+1})T_{k-1}(h_{k+1})}{T_k(h_{k+1})}$$

and

$$B(h) = h - h_k,$$

which incidentally satisfies $B(h_k) = 0$, then clearly (7) is satisfied. Now, plugging these A and B(h) into (6) yields

$$T_{k+1}(h) = \frac{(h_k - h_{k+1})T_{k-1}(h_{k+1})T_k(h)}{T_k(h_{k+1})} + (h - h_k)T_{k-1}(h)$$

$$= \frac{(h_k - h_{k+1})T_{k-1}(h_{k+1})P_k(h)}{T_k(h_{k+1})} + (h - h_k)P_{k-1}(h)$$

$$+ y\left(\frac{(h_k - h_{k+1})T_{k-1}(h_{k+1})Q_k(h)}{T_k(h_{k+1})} + (h-h_k)Q_{k-1}(h)\right)$$

$$= P_{k+1}(h) + yQ_{k-1}(h).$$

Thus, we have a recurrence relation to calculate P_{k+1} and Q_{k+1} once we know P_k, Q_k, P_{k-1}, Q_{k-1}, y_{k+1}, h_{k+1}, and h_k. When the sequence (0,0), (0,1), (1,1), (1,2), . . . is used for (M,N), then the following scheme exists for rational function extrapolation.[18] This scheme was first formalized by Stoer in 1961.[19]

$$R^i_{-1} = 0$$

---

[18]Roland Bulirsch and Josef Stoer, "Fehlerabschätzungen und Extrapolation mit rationalen Funktionen bei Verfahren vom Richardson--Typus," Numerische Mathematik, VI (1964), 420-1.

[19]Josef Stoer, "Über Zwei Algorithmen zur Interpolation mit rationalen Funktionen," Numerische Mathematik, III (1961), 285-304.

$$R_0^i = T(h_i)$$

$$R_m^i = R_{m-1}^{i+1} + \frac{R_{m-1}^{i+1} - R_{m-1}^{i}}{\left(\frac{h_i}{h_{i+m}}\right)^2 \left(1 - \frac{R_{m-1}^{i+1} - R_{m-1}^{i}}{R_{m-1}^{i+1} - R_{m-2}^{i+1}}\right) - 1} \tag{8}$$

for $m \geq 1$. Letting

$$D_m^i = R_m^i - R_{m-1}^{i+1}$$

$$C_m^i = R_m^i - R_{m-1}^{i}$$

and

$$W_m^i = R_m^i - R_m^{i-1} ,$$

then (8) can be converted to

$$D_m^i = \frac{C_{m-1}^{i+1} \; W_{m-1}^{i+1}}{\left(\frac{h_i}{h_{i+m}}\right)^2 D_{m-1}^{i} - C_{m-1}^{i+1}} , \text{ for } m \geq 1,$$

$$C_m^i = \frac{\left(\frac{h_i}{h_{i+m}}\right)^2 D_{m-1}^{i} \; W_{m-1}^{i+1}}{\left(\frac{h_i}{h_{i+m}}\right)^2 D_{m-1}^{i} - C_{m-1}^{i+1}} , \text{ for } m \geq 1,$$

and $$W_m^i = C_m^i - D_m^{i-1} ,$$

with $$C_0^i = D_0^i = T(h_i)$$

and $$W_0^i = T(h_i) - T(h_{i-1}) . \tag{9}$$

The underlying discretization method used by Bulirsch and

Stoer is due to Gragg.[20] It is called the Modified Midpoint-Rule. It is listed here for one equation, but can be easily adapted to a system of equations.

$$y_0 = \text{initial condition}$$

$$z_0 = y_0 + hf(a,y_0)$$

$$y_{r+1} = y_r + hf(x_r + h/2, z_r), \text{ for } r = 0,1,\ldots,n-1$$

and

$$z_{r+1} = z_r + hf(x_r + h, y_{r+1}), \text{ for } r = 0, 1,\ldots,n-1$$

Then

$$U(h) = y_n$$

and

$$G(h) = z_n - \frac{hf(a,y_n)}{2},$$

and finally

$$T(h) = \frac{U(h) + G(h)}{2}.$$

U, T, and G all have asymptotic expansions in even powers of h, with coefficients independent of h, which is necessary for this h^2-type extrapolation process to be applied to any sequence of h_k, rather than just the sequence $(h,h/2,h/4,h/8,\ldots)$.[21]

In Chapter III, I will examine the program DIFSY1 in detail. This program does Bulirsch and Stoer rational function extrapolation, using Gragg's Modified Midpoint-Rule as the underlying discretization method.

---

[20] William B. Gragg, "On Extrapolation Algorithms for Ordinary Initial Value Problems," Siam Journal of Numerical Analysis, II (3,1965), 395-6.

[21] William B. Gragg, 396.

CHAPTER III

ANALYSIS OF DIFY1

In this chapter, the rational function extrapolation program DIFSY1 will be examined in detail. It is H. G. Hussel's Watfiv program.[22] The version examined here is an implementation which I received from W. H. Enright.[23] I made modifications to remove some unnecessary statements and to add output facilities. A double DO loop was added to zero out an array. This was necessary in order to implement the program with Youngstown State University's Watfiv interpreter. I also supplied a main program segment that sets the initial step size HINIT, the maximum step size, HMAX, and the error tolerance, EPS, and defines the initial x value, X, called a in (1), the final x value, XEND, called b in (1), the number of equations, N, and the initial values, y(1), y(2), ..., y(n), called c_1, c_2, ..., c_n in (1). It also sets a variable NOFNS, declared COMMON, equal to zero. NOFNS is incremented through the Hussels program each time a call is made to the EXTERNAL subroutine YF, which evaluates the functions f_1, f_2, ..., f_n, from (1), at specified points. Hussel's program is contained as a subroutine called METHOD. The main program makes the call

---

[22] H. G. Hussels.

[23] Wayne H. Enright, Personal Communication.

CALL METHOD (N,YF,X,Y,XEND,EPS,HMAX,HINIT) by which control is passed to METHOD, which then attempts to solve (1). METHOD calls YF by the call CALL YF (X,Y,DY), where DY is an array to hold the values of $f_1(x,y_1,\ldots,y_n)$, . . ., $f_n(x,y_1,\ldots,y_n)$. A step by step description of DIFSY1 is given in Appendix A in the form of comment cards in the program, so my intention here is to explain the three major areas of interest in the program. These are what order of extrapolation is used, when and how the step size h is changed, and when and how results are accepted as correct.

The program contains logical variables KONV, KL, and GR which are partially used as switches to control the order of extrapolation. KONV is always false when the order of extrapolation that is being attempted is less than or equal to three, which is half of the maximum allowable order six. It is also used for error testing, and is kept true if an error test criterion, which will be described shortly, is passed. KL is true only the first time through the extrapolation loop, when only one modified midpoint value is known. Extrapolation cannot be performed without two different modified midpoint values. GR is true only when the order of extrapolation is at least five and is used in connection with changing the step size. It, in effect, prohibits the program from altering a step changing parameter FS. The order of extrapolation is controlled as follows. First through

fourth-order extrapolations are carried out without testing for convergence, with possible modifications of the step size between each iteration. After fourth order, a test is made for convergence. If successful, the x and y values are printed, and the process continues with a first-order extrapolation at the new point, called X, or control is returned to the main program if the interval has been fully covered. If convergence was not reached, there may be a modification to h, which will be described shortly, and then fifth-order extrapolation is attempted. Convergence gives the same result. Failure yields a possible modification to h and sixth-order extrapolation is attempted. Again, convergence gives the obvious result. Failure means sixth-order extrapolation may be attempted three more times with a different step size each time. There is also a counter JTI that is incremented each time one of the step size parameters, FY is less than .7 times the previous FY, or is greater than .7. An increment to JTI passes control back to the point where first order extrapolation is begun again. This is a built in safety valve so that if the extrapolation loop does not yield convergence, then the program will not continue in an infinite loop. A value JTI greater than five means the program stops iterating and returns control to the main program segment. Convergence may always be reached at fourth-order extrapolation, so the method may not be of variable order. If convergence is not reached, however,

the program is free to vary its order.

The step size h is controlled as follows. First, the program uses the sequence

$$\left(h, \frac{h}{2}, \frac{h}{3}, \frac{h}{4}, \frac{h}{6}, \frac{h}{8}, \frac{h}{12}\right) \tag{10}$$

for rational function extrapolation rather than the sequence

$$\left(h, \frac{h}{2}, \frac{h}{4}, \frac{h}{8}, \frac{h}{16}, \frac{h}{32}, \frac{h}{64}\right),$$

because it yields equal accuracy with only half the number of operations.[24] Recall that (9) needs $(h_i/h_{i+m})^2$ computed. DIFSY1 stores the values in the D array, and depending on the order of extrapolation being used, assigns values when needed. For example, when $(h_3/h_6)^2$ is needed for an even order extrapolation, that is, when the order counter L is odd, the D(4) is set to 64/9, since

$$h_3 = h/3 \text{ and } h_6 = h/8,$$

and

$$\frac{h_3}{h_6} = \frac{h/3}{h/8} = \frac{8}{3}$$

and

$$\left(\frac{8}{3}\right)^2 = \frac{64}{9}.$$

So, in effect, $D(k) = \left(\frac{h_{i+1-k}}{h_i}\right)^2$. Also, if the present order of extrapolation is i, then

---

[24] Roland Bulirsch and Josef Stoer, "Numerical Treatment of Ordinary Differential Equations by Extrapolation Methods," 5.

$$D(i) = \left(\frac{h_{i+1-i}}{h_i}\right)^2 = \left(\frac{h_1}{h_i}\right)^2 = \left(\frac{h}{h/M}\right)^2 = M^2$$

where M stores the current denominator in (10). However, h is not kept constant throughout the program. Most often, h is changed by the formula h = h(FY), where

$$FY = \left(\frac{EP_k}{|T(h_{k+1}) - T(h_k)|}\right)^{\frac{1}{2k-1}},$$

where EP_k decreases for k = 1, 2, 3, 4, which occurs when the order of extrapolation is one, two, three, or four. FY will increase h if

$$|T(h_{k+1}) - T(h_k)| < EP_k$$

and decrease h if

$$|T(h_{k+1}) - T(h_k)| > EP_k.$$

If

$$|T(h_{k+1}) - T(h_k)| = EP_k,$$

then h will not change. If the program gets completely through the rational function extrapolation loop, that is, gets to a point where no convergence has been recently obtained, then h is halved, and another attempt is made from first-order extrapolation on. However, since JTI is also incremented, this cannot happen more than five times without success. There are also limits on h. If h becomes smaller than HMIN, then the program stops. If h becomes larger than HMAX, then HMAX is assigned to h.

Finally, achieving convergence means that the most recently extrapolated value minus the second most recently

extrapolated value in absolute value, divided by the step size h, is less than or equal to EPS, the user specified error tolerance. When this is the case, the point x and the values of $y_1(x)$, $y_2(x)$, ..., $y_n(x)$ are printed. Then, if the end of the interval has not been reached, the process begins at a new point, determined by the new value picked for h.

Next, I will explain how DIFSY1 was compared to a single equation Adams Predictor Corrector program, which I modified and adapted to handle (1).[25]

---

[25] Richard L. Burden, J. Douglas Faires, and Albert C. Reynolds, Numerical Analysis (Boston: Prindle, Weber and Schmidt, 1978), pp. 268-9.

CHAPTER IV

EXPLANATION OF COMPARISON CRITERIA

As mentioned in Chapter I, past research has shown that DIFSYl is one of the best programs to solve (1).[26] I did a series of tests over a set of eight problems which appear in the next chapter. Recall that variable-order Adams methods were also competitive.[27] One tested was DIFSUB, due to C. W. Gear.[28] DIFSUB also appears in Gear's text on initial value problems.[29] However, upon examining DIFSUB, I found it to be a very large, multipurpose program, capable of handling stiff systems, and requiring subroutines for matrix inversion and the computation of partial derivatives. It has been mentioned that such programs should be viewed as complete software packages, rather than as single methods.[30] However, I mention one small test of DIFSYl with DIFSUB on problem one in Chapter VI.

---

[26] Wayne H. Enright and T. E. Hull, 959.

[27] T. E. Hull, et al, 605.

[28] C. William Gear, "DIFSUB for Solution of Ordinary Differential Equations," Communications of the ACM, XIV (3, March 1971), 188-90.

[29] C. William Gear, Numerical Initial Value Problems in Ordinary Differential Equations (Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1971), pp. 158-66.

[30] L. F. Shampine, H. A. Watts, and S. M. Davenport, 376.

The program that I compare to DIFSY1 is a fourth-order Adams Predictor Corrector program which I adapted to handle a system of n equations. The program is called PC WATFIV. It appears in Appendix B. I gave it the exact output features that were built into DIFSY1, so that no time differential would be realized when printing. PC outputs when the maximum difference in absolute value between the predictor value and the corrector value is less than the error tolerance EPSIL, in all components. It changes step size relative to the formulas

$$q = \left(\frac{(EPSIL)h}{.2|PREDICTOR(I) - CORRECTOR(I)|} \right)^{.25}$$

$$h = qh,$$

if q is between .1 and one, and h = .1h if q is less than .1.

I chose .0001 as my error tolerance for two reasons. First, since PC is a fourth-order method, it would easily be able to accomodate this tolerance, since initially h is $1.5(EPSIL)^{.25}$. Second, I was curious to witness DIFSY1's performance at a relatively low error tolerance, since it is mentioned that DIFSY1 is not as good as the variable-order Adams, as well as Runge-Kutta methods, for low error tolerances.[31] This makes sense, since a program like DIFSUB starts with a first-order method and works up in order, and the higher the order, the more function evaluations,

---

[31] Wayne H. Enright and T. E. Hull, 959.

and consequently, the more time that would be needed. Since DIFSY1 does not even check for convergence until after fourth-order rational function extrapolation is performed in hopes of increasing the step, it seems logical that it would take slightly more time at a lower error tolerance.

I compared DIFSY1 and PC in four main areas: accuracy, number of function calls, computer time, and computer storage. Accuracy is probably the most important area to the casual user, who wishes only a solution to the problem. For six of the problems, the actual solution is evaluated at the right endpoint of the interval for each equation. It is then subtracted from the value which DIFSY1 or PC finds. The absolute value is taken, and this is referred to as the accuracy. For the other two problems, both of which are nonlinear systems, the actual solution at the right endpoint is taken to be the value obtained from a Runge-Kutta Four program, an $O(h^4)$ method, which integrates the equations with step size .001. Accuracy is then determined in the same manner. If the results for individual problems in the next chapter have more than one value listed under accuracy, it is because not all equations in the system yielded the same accuracy. Both programs were run in single and double precision, with single precision only to determine which method is more susceptible to roundoff error.

The reader familiar with the cost of computer time can see that considerations on computer costs must be made.

If a function is very complicated to evaluate, obviously one would like it to be evaluated as few times as possible. Each time a call is made to evaluate a function, both programs increment a counter to record the call. However, DIFSY1 evaluates all n components of a system on each call, while PC evaluates only one specified function, so a true comparison of number of calls can only be made if the number of PC's calls is divided by the number of equations in the system in question. I adapted PC to evaluate all n components at once, like DIFSY1. I found this to take slightly more computer time and storage, because dummy variables were needed to send all n values, so the figures quoted here are for a PC which evaluates components one at a time.

Computer time and storage are the final two ways in which DIFSY1 and PC were compared. Time has been divided into compile time and virtual CPU time. Compile time is the amount of time that it takes the computer to translate a program from Watfiv, a high level language, to a machine text, and virtual CPU time is the amount of time the compiled program actually spends solving the initial value problem in the central processor, not counting time for the particular system's paging apparatus to function. Finally, a comparison is made between the programs' storage for the actual DIFSY1 and PC object codes as well as the amount of array area that each require. Clearly, the program which takes up less of the computer's costly time and space would be more

desirable, providing it gave acceptable accuracy. The results for the test runs on the eight problems are given in the next chapter.

CHAPTER V

TEST RESULTS

DIFSY1 and PC were put through a series of computer tests on the Amdahl 470 V/5 computer at Youngstown State University. The eight problems tested were as follows. Number one is included because of an upcoming comparison with DIFSUB.

$$y_1' = -y_1 \ , \ y_1 = 1$$

$$a = 0 \ , \ b = 2$$

Solution: $y_1 = e^{-x}$

Number two is a logistic growth problem.

$$y_1' = \frac{y_1 \ (\ 1 - y_1 \ / \ 20 \ )}{4}$$

$$a = 0 \ , \ b = 5$$

Solution: $y_1 = \dfrac{20}{1 + 19e^{-x/4}}$

Number three is a conflicting species problem.

$$y_1' = .002 \ (\ y_1 - y_1 y_2 \ ) \ , \ y_1 = 30$$

$$y_2' = -.001 \ (\ y_2 - y_1 y_2 \ ) \ , \ y_2 = 30$$

$$a = 0 \ , \ b = 4$$

Solution: Given by Runge-Kutta Four.

Number four is a nonlinear chemical reaction problem.

$$y_1' = -y_1 \quad , y_1 = 1$$

$$y_2' = y_1 - y_2^2 \quad , y_2 = 0$$

$$y_3' = y_2^2 \quad , y_3 = 0$$

$$a = 0 \ , \ b = 1$$

Solution: Given by Runge-Kutta Four.

Number five is a linear chemical reaction problem.

$$y_1' = 5y_1 + 4y_3 \quad , y_1 = 2$$

$$y_2' = 5y_1 + 2y_2 + 5y_3 \quad , y_2 = 1$$

$$y_3' = -2y_1 - y_3 \quad , y_3 = 2$$

$$a = 0 \ , \ b = 2$$

Solution: $y_1 = -6e^x + 8e^{3x}$

$$y_2 = -19e^{2x} + 20e^{3x}$$

$$y_3 = 6e^x - 4e^{3x}$$

Number six is a non-autonomous problem.

$$y_1' = y_2 - 1 \quad , y_1 = 1$$

$$y_2' = y_1 - x^2 + 1 \quad , y_2 = 1$$

$$a = 0 \ , \ b = 6$$

Solution: $y_1 = x^2 + 1$

$$y_2 = 2x + 1$$

Number seven is a mildly stiff system.

$$y_1' = -y_1 + y_2 + 2x \quad , y_1 = 3$$

$$y_2' = y_1 - 2y_2 \quad , y_2 = 0$$

$$y_3' = y_2 - y_3 - 4x - 6 \ , \ y_3 = 1$$

$$a = 0 \ , \ b = 3$$

Solution: $y_1 = e^{-x}/2 + e^{-3x}/2 + x^2 = 1$

$$y_2 = -e^{-3x} + x^2 + 1$$

$$y_3 = -e^{-x}/2 + e^{-3x}/2 + x^2 - 6x + 1$$

Number eight is a mildly stiff system.

$$y_1' = -5y_1 - 5y_2 \ , \ y_1 = 2$$

$$y_2' = -6y_1 - 4y_2 + 38e^{-4x}/5 \ , \ y_2 = 0$$

$$a = 0 \ , \ b = 3$$

Solution: $y_1 = e^{-10x} + e^{-4x}$

$$y_2 = e^{-10x} - e^{-4x}/5$$

The test results for the individual problems are presented in Table 1. In the remainder of this chapter, the results are presented for all eight problems viewed as a group, so that any variance in one or more of the comparison parameters caused by a single problem will be averaged into the group. However, some additional individual results are presented in the next chapter.

In regard to number of function calls, DIFSY1 made an average of 323.75 calls to its function evaluation subroutine, while PC made an average of 380 if the number of calls for a system is divided by the number of equations, and 902.5 if not.

Both programs took an average of .24 seconds per problem to compile each source code. Average virtual CPU time

TABLE 1

TEST RESULTS FOR INDIVIDUAL PROBLEMS

Number		Calls	Compile Time[a]	CPU Time-[b] All Points	CPU Time-[b] Only	Accuracy[c]		Object Code[d]
						Double	Single	
	1	108	.22	.0376	.0335	10	6	5408
D	2	126	.24	.0486	.0406	11	4	5464
I	3	126	.24	.0683	.0598	6	F[e]	5568
F	4	75	.26	.0500	.0486	10,6,6	5	5584
S	5	1419	.24	.9154	.8514	9,8,9	F	5696
Y	6	168	.23	.0849	.0744	5	2	5512
1	7	195	.24	.1260	.1215	0	0	5720
	8	369	.25	.1890	.1792	1	1	5672
	1	68	.24	.0271	.0145	6	4	7048
	2	168	.23	.0646	.0351	7	F	7104
	3	264	.26	.0764	.0496	6	F	7248
P	4	132	.24	.0317	.0238	3	3	7288
C	5	3012	.24	.7036	.4943	4,3,4	F	7416
	6	408	.24	.1081	.0698	2	F	7200
	7	984	.24	.2340	.1765	0	F	7432
	8	2184	.23	.6222	.3995	1	F	7360

[a] Compile time is given in seconds.

[b] CPU time is given in seconds.

[c] Accuracy is given as $10^{-accuracy}$.

[d] Object code is given in bytes.

[e] An F signifies no output was received.

for DIFSY1 is .189983 seconds, and for PC is .233454 seconds, if the programs are allowed to output data wherever they wish, and .176119 seconds and .157882 seconds, respectively, if they are modified to output data only at the right endpoint of the interval.

DIFSY1 needed 528 bytes of array storage for each problem, while PC needed four hundred. For object codes, that is, the amount of storage needed for the compiled programs, DIFSY1 needed an average of 5578 bytes, while PC needed an average of 7262 bytes.

In terms of accuracy, recall, both programs were to consider convergence to be achieved if the two values being compared were accurate to within 10^{-4}. In double precision, DIFSY1 was accurate to within $10^{-6.75}$ on the average, while PC was accurate to within $10^{-3.6}$ on the average. These results are analyzed in the next chapter.

CHAPTER VI

CONCLUSIONS

In this chapter, the results of Chapter V are analyzed. The number of function evaluations and the computer time and storage are clearly influenced by the step size h, since this has a direct bearing on how many times the method must be called to solve (1) over [a,b]. By construction, PC restricts h more than DIFSY1, a fact that can be seen by viewing output, since PC prints out at far more points per problem than DIFSY1. For example, in problem five, PC produced 151 output points, while DIFSY1 produced only 53. Over all problems, PC produced about four and a half times as many output points as DIFSY1. Depending on the user's point of view, this could be good or bad. For example, if one requires the solution to (1) at only a few points in [a,b], then DIFSY1 appears more suited to their needs. The number of output points difference is clearly seen in the average virtual CPU time for output at all points, since on the average DIFSY1 is about .043 seconds faster per problem. However, if the user desires output at the right endpoint only, then average virtual CPU time for PC is about .018 seconds faster.

In regard to program and array storage, although DIFSY1 needs on the average 128 more bytes of storage for its arrays, it also need 1684 less bytes for its object code. So on the average, DIFSY1 saves the user 1556 bytes of space.

The main advantage of the rational function extrapolation program appears to be in the accuracy requirement. Although both programs specified that the output values be accurate to within 10^{-4}, DIFSY1 was on the average accurate to within $10^{-6.75}$, while PC was only accurate to within $10^{-3.6}$. This appears to be due to the fact that the Adams Predictor Corrector program is more susceptible to roundoff error. To confirm this, I ran all eight problems in single precision for both methods. DIFSY1 produced output for six out of eight of the problems, while PC could only produce output for two out of eight problems. Both DIFSY1 and PC were on the average accurate to within $10^{-3.25}$ in single precision. Errors were returned for attempting to divide by zero in all six failed PC runs, while the two DIFSY1 failures were due to page limit being exceeded in problem three, and job time being exceeded in problem five. This implies that PC builds up roundoff error at a faster pace than DIFSY1, that is, that rational function extrapolation is more stable than the Adams Predictor Corrector method.

When comparing DIFSY1 to Gear's DIFSUB on problem one, I found that DIFSY1 was more than twice as accurate. Although that was only one problem,it does give an indication that DIFSY1 is very powerful.

In conclusion, my test results support the previous results of Hull, Enright, Shampine, Watts, and Davenport by confirming the fact that rational function extrapolation is one of the best methods available today to solve nonstiff

initial value problems. When the problem of stiffness arises, as in problems seven and eight, the method is not very reliable, and neither is Adams'. One must employ another type of method to solve stiff systems, for example, the method due to Deuflhard and Bader.[32] However, for nonstiff problems, rational function extrapolation gives very good results, especially in regards to computer storage and solution accuracy, provided that function evaluations are not expensive.

---

[32]P. Deuflhard and G. Bader, "A Semi-Implicit Mid-Point Rule for Stiff Systems of Ordinary Differential Equations," Technische Universität München, (August 1978), 1-48.

APPENDIX A

DIFSY1 WATFIV Computer Program

DIFSY1 WATFIV COMPUTER PROGRAM

```
$JOB
C
C     THE PURPOSE OF THIS PROGRAM IS TO SOLVE THE INITIAL
C     VALUE PROBLEM IN ORDINARY DIFFERENTIAL EQUATIONS.
C     THIS IS THE MAIN PROGRAM SEGMENT. THE USER MUST
C     SPECIFY THE FOLLOWING PARAMETERS:
C
C     HINIT - THE INITIAL STEP SIZE
C     N     - THE NUMBER OF EQUATIONS
C     X     - THE LEFT ENDPOINT OF THE INTERVAL
C     Y(I)  - THE N INITIAL VALUES
C     XEND  - THE RIGHT ENDPOINT OF THE INTERVAL
C     EPS   - THE ERROR TOLERANCE
C     HMAX  - THE MAXIMUM ALLOWABLE STEP SIZE
C
      DOUBLE PRECISION HINIT,X,Y(4),XEND,EPS,HMAX
C
C     YF IS THE EXTERNAL SUBROUTINE THAT IS CALLED
C     BY METHOD TO EVALUATE THE FUNCTIONS.
C
      EXTERNAL YF
C
C     NOFNS COUNTS THE NUMBER OF FUNCTION EVALUATIONS.
C
      COMMON NOFNS
      NOFNS=0
      HINIT=2.0D-2
      N=1
      X=0.0D0
      Y(1)=1.D0
      XEND=5.D0
      EPS=.0001D0
      HMAX=1.0D0
C
C     METHOD IS THE SUBROUTINE THAT SOLVES THE INITIAL
C     VALUE PROBLEM. OUTPUT OF THE SOLUTION AT PROGRAM
C     SELECTED POINTS IS DONE BY METHOD.
C
      CALL METHOD(N,X,Y,XEND,EPS,HMAX,HINIT)
C
C     THE NUMBER OF FUNCTION EVALUATIONS IS OUTPUT.
C
      WRITE (6,998) NOFNS
  998 FORMAT (1H ,'NO. OF EVALUATIONS IS ',I6)
      STOP
      END
      SUBROUTINE YF(X,Y,DY)
C
C     THIS IS THE FUNCTION EVALUATION SUBROUTINE.
C     ARRAY DY RETURNS THE VALUES OF F(X,Y).
C     ALSO, THE NUMBER OF FUNCTION EVALUATIONS IS
```

```
C      INCREMENTED.
C
       DOUBLE PRECISION X,Y(4),DY(4)
       COMMON NOFNS
       NOFNS=NOFNS+1
       DY(1)=-Y(1)
       RETURN
       END
       SUBROUTINE METHOD(N,YF,X,Y,XEND,EPS,HMAX,HINIT)
C
C      THIS IS THE HUSSELS RATIONAL FUNCTION EXTRAPOLATION
C      SUBROUTINE.
C
       DOUBLE PRECISION Y(4),YA(4),YL(4),YM(4),DY(4),DZ(4),
      +DT(4,7),D(7),X,XN,H,G,B,B1,U,V,C,TA,W,XEND,EPS,HMAX,
      +HINIT,R(4),HMIN,ERREST
C
C      EPS IS USED TO CHANGE THE STEP SIZE H.
C
       REAL*4 EP(4)/0.4E-1,0.16E-2,0.64E-4,0.256E-5/
C
C      THE LOGICAL VARIABLES ARE:
C
C      KONV - KONV IS FALSE WHEN THE ORDER OF
C             EXTRAPOLATION IS THREE OR LESS, WHEN NO TEST
C             IS MADE FOR CONVERGENCE. IT IS TRUE IF
C             CONVERGENCE HAS BEEN ACHIEVED.
C      BO   - THIS IS A SWITCH TO DIFFERENTIATE BETWEEN
C             ODD AND EVEN ORDER EXTRAPOLATION.
C      KL   - KL IS TRUE ONLY WHEN FINDING THE FIRST
C             MODIFIED MIDPOINT VALUE.
C      GR   - GR IS TRUE WHEN THE ORDER OF EXTRAPOLATION
C             IS FIVE OR SIX. IT IS USED IN CONNECTION WITH
C             CHANGING THE STEP SIZE H.
C
       LOGICAL*1 KONV,BO,KL,GR
       EXTERNAL YF
       COMMON NOFNS
C
C      DT WILL HOLD EXTRAPOLATED VALUES. IT MUST BE ZEROED
C      OUT FOR THE FIRST ITERATION.
C
       DO 993 I=1,4
       DO 993 J=1,7
       DT(I,J)=0.0D0
  993  CONTINUE
C
C      HMIN IS THE MINIMUM ALLOWABLE STEP.
C
       HMIN=1.D-14
C
C      H IS THE CURRENT STEP SIZE BEING ATTEMPTED.
C
       H=HINIT
```

```
C
C     CONTROL IS PASSED TO STATEMENT 1000 WHEN A FRESH
C     START IS BEING MADE AT THE POINT X. ALSO, THE
C     SAFETY VALVE JTI IS SET EQUAL TO ZERO. JTI IS
C     INCREMENTED WHEN EXTRAPOLATION FAILS TO ACHIEVE
C     CONVERGENCE. THIS IS ALLOWED TO HAPPEN ONLY FIVE
C     TIMES, SO THAT THE PROGRAM DOES NOT CONTINUE
C     RUNNING INDEFINITELY.
C
 1000 JTI=0
C
C     FY IS THE PARAMETER THAT IS USED TO CHANGE STEP
C     SIZE H BY THE FORMULA H=H*FY. IT IS INITIALLY SET
C     TO ONE.
C
      FY=1.D0
C
C     YA SAVES THE VALUES OF Y(X).
C
      DO 100 I=1,N
  100 YA(I)=Y(I)
C
C     H CAN NOT EXCEED THE LENGTH OF THE INTERVAL.
C
      IF (H.GT.XEND-X) H=XEND-X
C
C     DZ HOLDS THE VALUES OF F(X,Y)
C
      CALL YF(X,Y,DZ)
C
C     CONTROL IS PASSED TO STATEMENT 10 WHEN A NEW STEP
C     SIZE H IS BEING ATTEMPTED.
C
   10 XN=X+H
      BO=.FALSE.
C
C     M,JR, AND JS ARE THE DENOMINATORS FOR THE STEP SIZE
C     SEQUENCE (H,H/2,H/3,H/4,H/6,H/8,H/12).
C
      M=1
      JR=2
      JS=3
C
C     THIS LOOP CONTROLS THE ORDER OF EXTRAPOLATION.
C
      DO 260 J=1,10
      IF(.NOT.BO) GO TO 200
C
C     THE D ARRAY HOLDS THE VALUES (H(I+1-K)/H(I))**2.
C     FOR ODD ORDER EXTRAPOLATION, D(2)=16/9,
C     D(4)=64/9, AND D(6)=256/9.
C
      D(2)=1.777777777777778D0
      D(4)=7.1111111111111111D0
```

```
      D(6)=2.844444444444444D1
      GO TO 201
C
C     FOR EVEN ORDER EXTRAPOLATION, D(2)=9/4,
C     D(4)=9, AND D(6)=36.
C
  200 D(2)=2.25D0
      D(4)=9.D0
      D(6)=3.6D1
C
C     THE MAXIMUM ORDER OF EXTRAPOLATION IS SIX.
C     L IS ONE MORE THAN THE CURRENT ORDER.
C
  201 IF (J.LE.7) GO TO 202
      L=7
      D(7)=6.4D1
      GO TO 203
  202 L=J
      D(L)=M*M
  203 KONV=L.GT.3
C
C     THIS SECTION PERFORMS THE MODIFIED MIDPOINT
C     RULE, DUE TO GRAGG.
C
      M=M+M
      G=H/DFLOAT(M)
      B=G+G
      DO 210 I=1,N
      YL(I)=YA(I)
  210 YM(I)=YA(I)+G*DZ(I)
      M=M-1
      DO 220 K=1,M
      CALL YF(X+DFLOAT(K)*G,YM,DY)
      DO 220 I=1,N
      U=YL(I)+B*DY(I)
      YL(I)=YM(I)
      YM(I)=U
  220 CONTINUE
      CALL YF(XN,YM,DY)
      KL=L.LT.2
      GR=L.GT.5
C
C     FS IS USED IN CHANGING THE STEP SIZE H.
C
      FS=0.D0
C
C     THIS LOOP PERFORMS THE RATIONAL FUNCTION
C     EXTRAPOLATION ALGORITHM OF BULIRSCH AND STOER.
C
      DO 233 I=1,N
C
C     V HOLDS THE VALUE OF THE MODIFIED MIDPOINT RULE
C     AT THE PREVIOUS STEP.
C
```

```
      V=DT(I,1)
C
C     C HOLDS THE CURRENT VALUE OF THE MODIFIED
C     MIDPOINT RULE.
C
      C=(YM(I)+YL(I)+G*DY(I))*0.5D0
      DT(I,1)=C
      TA=C
C
C     IF KL IS TRUE, THE PROGRAM IS ON THE FIRST
C     ITERATION, AND HENCE, EXTRAPOLATION CAN NOT YET
C     BE PERFORMED.
C
      IF(KL) GO TO 233
      DO 231 K=2,L
      B1=D(K)*V
      B=B1-C
      W=C-V
      U=V
      IF (B.EQ.0.D0) GO TO 230
      B=W/B
      U=C*B
      C=B1*B
  230 V=DT(I,K)
      DT(I,K)=U
C
C     TA HOLDS THE MOST RECENTLY EXTRAPOLATED VALUE.
C
  231 TA=U+TA
C
C     IF THE ORDER OF EXTRAPOLATION IS LESS THAN FOUR,
C     DO NOT TEST FOR CONVERGENCE.
C
      IF(.NOT.KONV) GO TO 232
C
C     R=ABS(LAST EXTRAPOLATION-THIS EXTRAPOLATION)/H.
C
      R(I)=DABS(Y(I)-TA)/H
C
C     IF R LESS THAN OR EQUAL TO THE ERROR TOLERANCE,
C     CONVERGENCE HAS BEEN ACHIEVED.
C
      IF(R(I).GT.EPS) KONV=.FALSE.
C
C     IF THE ORDER OF EXTRAPOLATION IS AT LEAST FIVE,
C     DO NOT ATTEMPT TO ALTER ANY OF THE STEP CHANGING
C     PARAMETERS.
C
  232 IF(GR) GO TO 233
C
C     FV=ABS(LAST MIDPOINT-PRESENT MIDPOINT)
C
      FV=DABS(W)
C
```

```
C     IF THERE IS SOME DIFFERENCE, FS IS INCREASED.
C
      IF(FS.LT.FV) FS=FV
C
C     Y NOW HOLDS THE MOST RECENTLY EXTRAPOLATED VALUE.
C
  233 Y(I)=TA
C
C     IF FS IS ZERO, DO NOT ATTEMPT TO ALTER FY.
C
      IF(FS.EQ.0.) GO TO 250
C
C     FA HOLDS THE LAST VALUE OF FY.
C
      FA=FY
      K=L-1
C
C     FY=(EP(K)/ABS(LAST MIDPNT-THIS MIDPNT))**(1/(2L-1)).
C     THIS IS WHAT H MAY BE ALTERED BY.
C
      FY=(EP(K)/FS)**(1./DFLOAT(L+K))
      IF (L.EQ.2) GO TO 240
C
C     IF FY GREATER THAN OR EQUAL TO OLD FY OR FY LESS THAN
C     .7, THEN SAFETY VALVE JTI IS INCREMENTED, H IS ALTERED,
C     AND A NEW ATTEMPT IS MADE TO CONTINUE WITH THIS NEW
C     STEP SIZE.
C
      IF(FY.LT.0.7*FA) GO TO 250
  240 IF(FY.GT.0.7) GO TO 250
      JTI=JTI+1
      IF (JTI.GT.5) GO TO 250
      H=H*FY
      IF (H.GT.HMAX) H=HMAX
      GO TO 10
C
C     IF CONVERGENCE HAS BEEN ACHIEVED, OUTPUT THE
C     RESULTS, AND PROCEED TO THE NEXT STEP. IF NOT,
C     HALVE THE STEP SIZE AND BEGIN AGAIN.
C
  250 IF (KONV) GO TO 20
      D(3)=4.D0
      D(5)=1.6D0
      BO=.NOT.BO
      M=JR
      JR=JS
  260 JS=M+M
      IF (JTI.GT.5) GO TO 30
      IF (DABS(H).LE.HMIN) GO TO 30
      H=H*0.5D0
      IF (DABS(H).GE.HMIN) GO TO 10
      H=DSIGN(HMIN,H)
      GO TO 10
C
```

```
C     OUTPUT THE RESULTS, CHANGE H, MOVE X UP TO XN,
C     AND BEGIN AT THIS NEW POINT.
C
   20 WRITE (6,551) XN,(Y(K),K=1,N)
  551 FORMAT(1H ,F20.14,4(2X,F20.14))
      X=XN
      H=H*FY
      IF(H.GT.HMAX) H=HMAX
      IF(DABS(XEND-X).LT.1.D-8) RETURN
      GO TO 1000
C
C     IF THERE HAS BEEN A FAILURE, THE INITIAL Y VALUES
C     ARE RESTORED, H IS SET EQUAL TO ZERO, AND THE
C     PROGRAM IS ABORTED.
C
   30 H=0.D0
      DO 300 I=1,N
  300 Y(I)=YA(I)
      RETURN
      END
$ENTRY
/*
```

APPENDIX B

PC WATFIV Computer Program

PC WATFIV COMPUTER PROGRAM

```
$JOB
C
C     THIS IS THE MAIN PROGRAM SECTION.
C     THE USER MUST SPECIFY:
C     N       -   THE NUMBER OF EQUATIONS IN THE SYSTEM
C     Y(I)    -   THE INITIAL VALUES
C     A       -   THE LEFT ENDPOINT
C     B       -   THE RIGHT ENDPOINT
C     EPSIL   -   THE ERROR TOLERANCE.
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION Y(4)
      EXTERNAL F
      N=2
      Y(1)=1.D0
      Y(2)=1.D0
      A=0.D0
      B=6.D0
      EPSIL=.0001D0
      H=1.5D0*EPSIL**.25
C
C     PC SOLVES THE INITIAL VALUE PROBLEM.
C
      CALL PC(Y,N,A,B,EPSIL,H)
      STOP
      END
      FUNCTION F(N,T,U,NUM)
C
C     F IS THE FUNCTION EVALUATION FUNCTION.
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION U(4)
      GO TO (1,2),NUM
    1 F=U(2)-1.D0
      RETURN
    2 F=U(1)-T*T+1.D0
      RETURN
      END
      SUBROUTINE PC(W1,N,A,B,EPSIL,H)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION T(5),W0(4),W1(4),W2(4),W3(4),W4(4),W5(4)
      NCALLS=0
C
C     IFLAG IS ONE IF THE PROGRAM IS AT POINT A.
C
      IFLAG=1
      IND=0
      T(1)=A
  551 FORMAT (1H ,F20.14,4(2X,F20.14))
  333 DO 8 I=2,4
```

```
    8 T(I)=A+(I-1)*H
C
C     THE FIRST THREE VALUES ARE RUNGE-KUTTA VALUES.
C
      CALL RK4(H,W1,W2,T(1),NCALLS,N)
      WRITE (6,551) T(2),(W2(K),K=1,N)
      CALL RK4(H,W2,W3,T(2),NCALLS,N)
      WRITE (6,551) T(3),(W3(K),K=1,N)
      CALL RK4(H,W3,W4,T(3),NCALLS,N)
      WRITE (6,551) T(4),(W4(K),K=1,N)
C
C     L IS ZERO IF VALUES ARE OBTAINED BY PREDICTOR
C     CORRECTOR LOOPS, AND ONE IF VALUES ARE OBTAINED
C     BY RUNGE-KUTTA FORMULAS.
C
      L=0
C
C     CONTROL IS TRANSFERRED HERE IS THE PREDICTOR
C     CORRECTOR LOOPS ARE TO BE USED.
C
    5 T(5)=T(4)+H
C
C     T(5) CAN NOT SURPASS THE INTERVAL.
C
      IF (T(5).LT.B) GO TO 6
      T(5)=B
      H=T(5)-T(4)
C
C     IND IS SET TO ONE IF THE PROGRAM HAS REACHED
C     THE END OF THE INTERVAL.
C
      IND=1
C
C     LOOP 34 COMPUTES A PREDICTOR VALUE.
C
    6 DO 34 NUM=1,N
      P=F(N,T(4),W4,NUM)
      Q=F(N,T(3),W3,NUM)
      R=F(N,T(2),W2,NUM)
      S=F(N,T(1),W1,NUM)
   34 W0(NUM)=W4(NUM)+H*(55.D0*P-59.D0*Q+37.D0*R-9.D0*S)/
     +24.D0
C
C     LOOP 35 COMPUTES A CORRECTOR VALUE.
C
      DO 35 NUM=1,N
      P=F(N,T(5),W0,NUM)
      Q=F(N,T(4),W4,NUM)
      R=F(N,T(3),W3,NUM)
      S=F(N,T(2),W2,NUM)
      NCALLS=NCALLS+8
   35 W5(NUM)=W4(NUM)+H*(9.D0*P+19.D0*Q-5.D0*R+S)/24.D0
C
C     SIGMA=ABS(PREDICTOR-CORRECTOR).
```

```
C
      SIGMA=DABS(W5(1)-W0(1))
C
C     IF N IS ONE, THERE IS ONLY ONE EQUATION.
C
      IF (N.EQ.1) GO TO 666
C
C     SS IS THE MAXIMAL DIFFERENCE OF ALL EQUATIONS
C     IN THE SYSTEM.
C
      DO 50 NUM=2,N
      SS=DABS(W5(NUM)-W0(NUM))
      IF (SS.GT.SIGMA) SIGMA=SS
   50 CONTINUE
  666 SIGMA=.1D0*SIGMA
C
C     IF .1*EPSIL LESS THAN OR EQUAL TO SIGMA, AND SIGMA
C     LESS THAN OR EQUAL TO EPSIL, PRINT THE VALUES. IF
C     NOT, CHANGE H AND COMPUTE NEW VALUES BY RUNGE-KUTTA.
C
      IF(.1D0*EPSIL.LE.SIGMA.AND.SIGMA.LE.EPSIL) GO TO 51
      GO TO 52
C
C     IFLAG IS ZERO IF THE PROGRAM IS NOT AT POINT A.
C
   51 IFLAG=0
      WRITE (6,551) T(5),(W5(K),K=1,N)
      DO 66 KK=1,N
      W1(KK)=W2(KK)
      W2(KK)=W3(KK)
      W3(KK)=W4(KK)
   66 W4(KK)=W5(KK)
      T(1)=T(2)
      T(2)=T(3)
      T(3)=T(4)
      T(4)=T(5)
      L=0
C
C     IF IND IS ONE, THE TASK HAS BEEN COMPLETED.
C
      IF (IND.EQ.1) GO TO 18
C
C     CONTINUE AT THE NEXT POINT.
C
      GO TO 5
C
C     Q IS USED TO CHANGE THE STEP SIZE H.
C
   52 Q=((EPSIL*H)/2.D0*SIGMA))**.25
C
C     IF SIGMA GREATER THAN EPSIL, CHANGE H.
C
      IF (SIGMA.GT.EPSIL) GO TO 17
      IF (Q.LE.1) H=Q*H
```

```
C
C     GET NEW VALUES BY RUNGE-KUTTA.
C
   14 L=1
      DO 77 KK=1,N
   77 W1(KK)=W5(KK)
      T(1)=T(5)
      T(2)=T(1)+H
      IF (T(2).LT.B) GO TO 56
      T(2)=B
      H=T(2)-T(1)
      IND=1
   56 CALL RK4(H,W1,W2,T(1),NCALLS,N)
      WRITE (6,551) T(2),(W2(K),K=1,N)
      IF (IND.EQ.1) GO TO 18
      T(3)=T(2)+H
      IF (T(3).LT.B) GO TO 57
      T(3)=B
      H=T(3)-T(2)
      IND=1
   57 CALL RK4(H,W2,W3,T(2),NCALLS,N)
      WRITE (6,551) T(3),(W3(K),K=1,N)
      IF (IND.EQ.1) GO TO 18
      T(4)=T(3)+H
      IF (T(4).LT.B) GO TO 64
      T(4)=B
      H=T(4)-T(3)
      IND=1
   64 CALL RK4(H,W3,W4,T(3),NCALLS,N)
      WRITE (6,551) T(4),(W4(K),K=1,N)
      IF (IND.EQ.1) GO TO 18
C
C     CONTINUE WITH THE PREDICTOR CORRECTOR LOOPS
C     WITH THE NEW INITIAL VALUES.
C
      GO TO 5
C
C     IF Q LESS THAN .1, H=.1*H, ELSE H=Q*H.
C
   17 IF (Q.LT..1) GO TO 58
      H=Q*H
      GO TO 59
   58 H=.1D0*H
C
C     GO TO 333 IF STILL AT POINT A.
C
   59 IF (IFLAG.EQ.1) GO TO 333
C
C     COMPUTE RUNGE-KUTTA VALUES IF L=1.
C
      IF (L.EQ.1) GO TO 14
C
C     PRINT THE NUMBER OF FUNCTION CALLS.
C
```

```
   18 WRITE (6,862) NCALLS
  862 FORMAT (1H ,'NUMBER OF FUNCTION CALLS IS ',I8)
      RETURN
      END
      SUBROUTINE RK4(H,A,B,T,NCALLS,N)
C
C     RK4 COMPUTES THE RUNGE-KUTTA VALUES.
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(4),HO(4),B(4),XK1(4),XK2(4),XK3(4),XK4(4)
      DO 22 NUM=1,N
   22 XK1(NUM)=H*F(N,T,A,NUM)
      DO 23 NUM=1,N
   23 HO(NUM)=A(NUM)+XK1(NUM)/2.D0
      DO 24 NUM=1,N
   24 XK2(NUM)=H*F(N,T+H/2.D0,HO,NUM)
      DO 25 NUM=1,N
   25 HO(NUM)=A(NUM)+XK2(NUM)/2.D0
      DO 26 NUM=1,N
   26 XK3(NUM)=H*F(N,T+H/2.D0,HO,NUM)
      DO 27 NUM=1,N
   27 HO(NUM)=A(NUM)+XK3(NUM)
      DO 28 NUM=1,N
   28 XK4(NUM)=H*F(N,T+H,HO,NUM)
      DO 29 NUM=1,N
   29 B(NUM)=A(NUM)+(XK1(NUM)+2.D0*XK2(NUM)+2.D0*XK3(NUM)+
     +XK4(NUM))/6.D0
      NCALLS=NCALLS+4*N
      RETURN
      END
$ENTRY
/*
```

BIBLIOGRAPHY

Books

Burden, Richard L., Faires, J. Douglas, and Reynolds, Albert C. Numerical Analysis. Boston: Prindle, Weber, and Schmidt, 1978.

Gear, C. William. Numerical Initial Value Problems in Ordinary Differential Equations. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1971.

Articles

Bulirsch, Roland and Stoer, Josef. "Fehlerabschätzungen und Extrapolation mit rationalen Funktionen bei verfahren vom Richardson-Typus." Numerische Mathematik, VI (1964), 413-27.

Bulirsch, Roland and Stoer, Josef. "Numerical Treatment of Ordinary Differential Equations by Extrapolation Methods." Numerische Mathematik, VIII (1966), 1-13.

Deuflhard, P. and Bader, G. "A Semi-Implicit Mid-Point Rule for Stiff Systems of Ordinary Differential Equations." Technische Universität München, (August 1978), 1-48.

Enright, Wayne H. Personal Communication.

Enright, Wayne H. and Hull, T. E. "Test Results on Initial Value Methods for Non-Stiff Ordinary Differential Equations." SIAM Journal of Numerical Analysis, XIII (6, December 1976), 944-61.

Fox, P. A. "DESUB: Integration of a First Order System of Ordinary Differential Equations." Mathematical Software. Edited by John Rice. New York: Academic Press, 1971.

Gear, C. William. "DIFSUB for Solution of Ordinary Differential Equations." Communications of the ACM, XIV (3, March 1971), 185-90.

Gragg, William B. "On Extrapolation Algorithms for Ordinary Initial Value Problems." SIAM Journal of Numerical Analysis, II (3, 1965), 384-403.

Hull, T. E. et al, "Comparing Numerical Methods for Ordinary Differential Equations." SIAM Journal of Numerical Analysis, IX (4, December 1972), 603-37.

Hussels, H. G. "Schrittweitensteuerung bei der Integration gewöhnlicher Differentialgleichungen mit Extrapolationverfahren." Unpublished M. Sc. Thesis, Universität Köln, 1973.

Joyce, D. C. "Survey of Extrapolation Processes in Numerical Analysis." SIAM Review, XIII (4, October 1971), 435-90.

Lambert, J. D. "The Initial Value Problem for Ordinary Differential Equations." The State of the Art in Numerical Analysis. Edited by D. A. H. Jacobs. London: Academic Press, 1977.

Shampine, L. F., Watts, H. A., and Davenport, S. M. "Solving Nonstiff Ordinary Differential Equations - The State of the Art." SIAM Review, XVIII (3, July 1976), 376-411.

Stoer, Josef. "Über zwei Algorithmen zur Interpolation mit rationalen Funktionen." Numerische Mathematik, III (1961), 285-304.