SVAZEK 13 (1968) APLIKACE MATEMATIKY ČÍSLO 1

PRACTICAL ASPECTS OF INTERVAL COMPUTATION

RAMON E. MOORE

0. INTRODUCTION

"Interval computation" or "interval analysis", [36], is concerned with the design and study of algorithms for the computer which produce, automatically, guaranteed upper and lower bounds to exact solutions of various types of mathematical problems. The aim is to exploit the high speed and accuracy of the computer by programming it to carry out all the necessary detailed analysis for each specific problem.

I wish to thank all those whose contributions are mentioned in the text following and in addition I am grateful for the programming assistance of Mr. M. Mc Clellan and Mr. D. Good. I am grateful for and honored by the invitation of Mr. I. Babuška and the Czechoslovak Academy of Sciences to present this paper to the Liblice conference.

My remarks may range from self-evident generalities to boring details but I hope that in between there may be something of interest to help stimulate further efforts. Much remains to be done.

1. COMPUTERS AND COMPUTATION

The electronic stored program digital computing machine – the "computer", for short – has made extensive, rapid numerical computation available to vast numbers of scientists and engineers and has revolutionized the applications of mathematics.

An electromechanical desk calculating machine requires about ten seconds to multiply a pair of ten decimal digit numbers. The computer can do this millions of times as fast at less than a thousand times the cost.

For computations of sufficient length, the computer is vastly less expensive and time consuming than the desk calculator. If you are only going to do one multiplication, or a few, then use the desk calculator, a slide rule, or pencil and paper. If you are going to calculate the orbit of Mars, then use the computer. Recently [23], a computer reproduced *in a few minutes* calculations on the orbit of Mars upon which the astronomer KEPLER spent four years.

The analytical preparation and programming of computations to be carried out by the computer can, unfortunately, be difficult, time-consuming, and costly. The early recognition of this fact motivated the development of a vast array of aids to the preparation of computations for the computer based on the use of the computer itself.

The earliest computers had only fixed point arithmetic wired-in and programs were coded in a "binary" representation of machine language. A considerable amount of time had to be spent "scaling" fixed decimal point computations before they could be programmed. And programming in machine language required a considerable amount of binary arithmetic by the programmer in order to assign addresses to the quantities in the program. An odd, perhaps unfortunate, side effect of this was the introduction of lessons in binary arithmetic at various elementary levels in the public schools.

Computer manufacturers very soon introduced built-in floating point arithmetic operations thereby eliminating the need for scaling preparation. Once computers appeared with large enough "memories" (high speed storage capacity of several thousand words), programmers invented various "languages" with notation much closer to "ordinary" mathematical notation and easier to use than machine language such as FORTRAN and ALGOL in all their versions, and many others as well and wrote compiler programs to translate statements written in these languages into machine code[1]); thereby greatly reducing the human time and effort needed to program a computation. The preparation of the computer program for Kepler's determination of the orbit of Mars was made with the help of FORTRAN in about three weeks.

[1]) The compiled machine code is, somewhat unfortunately, often a less efficient program than could be written by hand in machine language. A program written in FORTRAN might take twice as long to run on the computer as the hand written machine language program, but only a tenth of the time to writte and check out.

All sorts of programs, "routines", and "subroutines" have been written and rewritten and distributed for use in the approximate numerical solution of various types of problems.

Programs enabling the computer to perform algebraic manipulations and symbolic differentiation have begun to appear and should eventually have a strong impact on computational practices [39], [12]. Such programs seem to require fairly large memory capacity, perhaps 30 000 words or more, to be really effective.

One source of difficulty in connection with the publication, distribution and use of programs written for a specific type of computer or even in a specific programming language has been the proliferation and rapid obsolescence of "better and better" computers, programming languages, and operating systems. This situation has given rise to intense efforts to standardize programming languages. International groups have been formed to agree upon a universal programming language. As a result, we now have, in addition to all the other programming languages, several "universal" programming languages.

In practical applications of mathematics much of the analytical work that is done on a problem consists of performing transformations using previously obtained information in order to simplify further deductions about the properties of solutions – in particular, to make numerical computations go as "smoothly" as possible. "Preconditioning" transformations or a variety of other analytical devices can be programmed to be carried out by the computer before, during, or after a numerical solution.

As an aid to the analytical preparation of computations, programs for the approximate solution of various types of problems can be written which enable the computer itself to analyze and control the accuracy of the computation. A natural approach to the construction of such programs can be based upon computations with intervals, [36].

2. DECIMAL ARITHMETIC AND COMPUTING WITH INTERVALS

2.1. Decimal arithmetic. In most practical applications of mathematics, approximate results will suffice. Requirements of accuracy in numerical calculations vary, of course, but exact results are rarely needed.

Exact results will usually be impossible to compute anyhow, since most calculations begin with inexact data (i.e. with quantities known only to a certain degree of accuracy or with finite decimal approximation to real initial data).

Exact arithmetic calculation with rational numbers, even beginning with small integers, can lead to results with arbitrarily large integer numerators and denominators; and so can become prohibitively time consuming – even on electronic digital computers. Programs have been written [12] for exact arithmetic with arbitrarily$^{2)}$

$^{2)}$ Within the memory capacity of the computer.

large integers. For multiplication, the time increases as the square of the number of digits and each multiplication approximately doubles the number of digits.

For all these reasons, practical calculations are nearly always carried out only approximately using the more convenient decimal (or binary) fractions. Only part of the decimal digits in intermediate and final results are retained.

An assumption commonly made concerning the accuracy of such approximate calculations is that the number of correct decimal digits ("significant figures") in the result of a single arithmetic operation is the same as the smaller of the numbers of correct digits in the two operands (except in subtraction where it may be less on account of cancellation of leading digits).

This rule of thumb gives, of course, only a rough guide to an estimation of the accuracy of final results of a calculation.[3])

Examples of calculations occur in practice in which the number of correct decimal digits decreases rapidly or increases rapidly once an initial error has been made *no matter how many decimal digits (beyond a certain minimum) are carried in subsequent steps.*

Consider the following two illustrative computations (chosen from the excellent book of BABUŠKA, PRÁGER, and VITÁSEK, [6]) in which e is the base of natural logarithms, $e = 2{\cdot}71828\,18284\,59\ldots$

$$(1) \qquad I_n = (1/e) \int_0^1 x^n e^x \, dx .$$

From mathematical tables (or by long division) we find that

$$1/e = 0{\cdot}36787\,94411\,7\ldots$$

Thus

$$I_0 = 1 - (1/e) = {\cdot}63212\,05588\ldots$$

Table 1

	$d = 2$	$d = 3$	$d = 10$
I_0	·63	·632	·63212 05588
I_1	·37	·368	·36787 94412
I_2	·26	·264	·26424 11176
I_3	·22	·208	·20727 66472
I_4	·12	·168	·17089 34112
I_5	·40	·16	·14553 29440
I_6	−1·4	·04	·12680 23360
I_7	10·8	·72	·11238 36480

[3]) For a survey of some refinements of this rule, see [33].

Compute I_n from the recurrence formula (derived using "integration by parts") $I_n = 1 - nI_{n-1}$ for $n = 1, 2, 3, 4, 5, 6, 7$.

Starting with I_0 correct to d decimal digits and doing the subsequent arithmetic *exactly* we obtain the results given in Table 1.

The first 11 correct digits of I_7 are actually $I_7 = \cdot 11238\ 35040\ 6\ldots$

In order to obtain I_7 correct to 10 decimal digits it turns out to be necessary to start with 14 correct digits in I_0.

If we wish to continue the calculation of I_n for $n = 8, 9, 10, \ldots, 14$ then it turns out to be necessary to start with 16 correct decimal digits in I_0 in order to obtain $I_{14} = \cdot 0627\ldots$ to even 3 correct digits!

Thus it is in the nature of this example that a rapid loss of accuracy occurs due to growth of error in initial data.

On the other hand, consider the following example:

(2) $$I_7 = \cdot 11238\ 35040\ 6\ldots$$

Compute

$$I_{n-1} = \frac{1 - I_n}{n} \quad \text{for} \quad n = 7, 6, 5, 4, 3, 2, 1\ .$$

Starting with I_7 correct to d decimal digits and carrying only d decimal digits in subsequent calculations we obtain results which are also accurate to d places. Furthermore, starting with I_7 correct to only d digits and carrying 10 decimal digits in subsequent calculations we obtain the results given in Table 2.

In this example the calculations beginning with I_7 correct to only two decimal digits produce a result for I_0, for instance, which is correct to better than five digits.

Table 2

	$d = 2$	$d = 3$	$d = 10$
I_7	·11	·112	·11238 35040
I_6	·12714 28571	·12685 71428	·12680 23565
I_5	·14547 61904	·14552 38095	·14553 29405
I_4	·17090 47619	·17089 52381	·17089 34118
I_3	·20727 38095	·20727 61903	·20727 66470
I_2	·26424 20635	·26424 12699	·26424 11176
I_1	·36787 89682	·36787 93650	·36787 94411
I_0	·63212 10318	·63212 06350	·63212 05588

2.2. Computing with intervals. One means of keeping track of how many digits are correct during a calculation is to do the calculation with "interval numbers", [36].

The set (interval) of all real numbers x such that $1 \leqq x \leqq 2$ is denoted by [1, 2].

Similarly, for any pair of real numbers a, b such that $a \leqq b$, $[a, b]$ represents the interval of numbers between a and b, including end points.

We can think of the quantity $[a, b]$ as another kind of number – an *interval number*, made up of two real numbers – just as we regard the ratio of two integers m/n as another kind of number – a *rational number*, made up of two integers.

In order to set up a useful way of doing arithmetic with interval numbers we can use the following simple facts about inequalities.

Suppose a, b, r and x are real numbers and that p is a positive real number and n is a negative real number. The following statements are then true.

1. $p > 0$
2. $n < 0$
3. if $r \neq 0$, then either $r > 0$ or $r < 0$
4. if $a \leqq x \leqq b$, then $a + r \leqq x + r \leqq b + r$.
5. if $a \leqq x \leqq b$, then $pa \leqq px \leqq pb$
6. if $a \leqq x \leqq b$, then $nb \leqq nx \leqq na$

From these simple properties of inequalities the following additional properties can be derived:

7. if $a \leqq x \leqq b$ and $c \leqq y \leqq d$, then $a + c \leqq x + y \leqq b + d$
8. if $a \leqq x \leqq b$, then $-b \leqq -x \leqq -a$
9. if $0 < a \leqq x \leqq b$, then $1/b \leqq 1/x \leqq 1/a$
10. if $a \leqq x \leqq b$ and $c \leqq y \leqq d$, then
    $\min(ac, ad, bc, bd) \leqq xy \leqq \max(ac, ad, bc, bd)$

Based on these results we define arithmetic operations for interval numbers as follows:

addition	$[a, b] + [c, d] = [a + c, b + d]$
negative	$-[a, b] = [-b, -a]$
subtraction	$[a, b] - [c, d] = [a, b] + (-[c, d])$
multiplication	$[a, b]\,[c, d] = [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)]$
reciprocal	$1/[a, b] = [1/b, 1/a]$ (if $a > 0$)
division	$[a, b]/[c, d] = [a, b]\,(1/[c, d])$

We can mix arithmetic with interval numbers and real numbers by treating $[a, a]$ and a as the same thing.

If we perform a calculation with interval numbers using the arithmetic operations just defined for such numbers, we will obtain interval numbers as results. These will *contain* the exact real number results of the same sequence of arithmetic operations for any set of choices of real numbers from the intervals involved. If we start with intervals containing the initial data and compute the end points of the intervals using approximate decimal arithmetic, and if we take care to "round" in such a way that the resulting approximate intervals contain the correct interval results then we will have computed *intervals which contain the exact real result.*

In short, if we calculate with interval numbers we can keep track of how many digits are correct.

To illustrate, we repeat now some of the calculations of the examples already discussed – but this time using *interval arithmetic*.

Consider again the computational problem beginning with $I_0 = 1 - (1/e)$, or $I_0 = \cdot 63212\,05588\ldots$ to compute $I_n = 1 - nI_{n-1}$ for $n = 1, 2, \ldots, 7$.

Starting with intervals of width one in the dth decimal place and containing the exact value of I_0 and doing the subsequent interval arithmetic to 10 decimal places we obtain the results given in Table 3. To illustrate how these results were obtained we will first carry out explicitly the interval calculation of I_1, I_2, I_3 for $d = 2$.

With $d = 2$, we put

$$I_0 = [\cdot 63, \cdot 64]\,;$$

then

$$I_1 = 1 - I_0 = 1 + [-\cdot 64, -\cdot 63]\,,$$

and so

$$I_1 = [\cdot 36, \cdot 37]\,.$$

Similarly,

$$I_2 = 1 - 2I_1 = 1 - 2[\cdot 36, \cdot 37] = 1 - [\cdot 72, \cdot 74] = 1 + [-\cdot 74, -\cdot 72]\,;$$

then

$$I_2 = [\cdot 26, \cdot 28]\,.$$

Next,

$$I_3 = 1 - 3I_2 = 1 - 3[\cdot 26, \cdot 28] = 1 - [\cdot 78, \cdot 84] = 1 + [-\cdot 84, -\cdot 78] = [\cdot 16, \cdot 22].$$

The rest of the results for $d = 2, 3, 10$ are given in Table 3.

Table 3

	$d = 2$	$d = 3$	$d = 10$
I_0	[·63, ·64]	[·632, ·633]	[·63212 05588, ·63212 05589]
I_1	[·36, ·37]	[·367, ·368]	[·36787 94411, ·36787 94412]
I_2	[·26, ·28]	[·264, ·266]	[·26424 11176, ·26424 11178]
I_3	[·16, ·22]	[·202, ·208]	[·20727 66466, ·20727 66472]
I_4	[·12, ·36]	[·168, ·192]	[·17089 34112, ·17089 34136]
I_5	[−·8, ·4]	[·04, ·16]	[·14553 29320, ·14553 29440]
I_6	[−1·4, 5·8]	[·04, ·76]	[·12680 23360, ·12680 24080]
I_7	[−39·6, 10·8]	[−4·32, ·72]	[·11238 31440, ·11238 36480]

Each interval number entry in Table 3 contains the exact value of I_n because of the properties of inequalities discussed earlier, especially properties 7., 8., 9., and 10. Each entry of Table 3 also contains the corresponding entry of Table 1. It is in this way that the calculations which produced Table 3 kept track of the number of correct digits in the calculations which produced Table 1.

We turn now to the second example, connected with Table 2,

$$I_7 = \cdot 11238\ 35040\ 6\ldots$$

$$I_{n-1} = \frac{1 - I_n}{n} \quad \text{for} \quad n = 7, 6, 5, 4, 3, 2, 1\ .$$

We will carry out explicitly the interval calculation corresponding to the first column of results in Table 2.

The exact value of I_7 is contained in the interval $[\cdot 11, \cdot 12]$; so we put $I_7 = [\cdot 11, \cdot 12]$ and calculate (using 10 decimal digit arithmetic with appropriate rounding for computation of end points of intervals).

$$\begin{aligned}
I_6 &= (1 - [\cdot 11, \cdot 12])/7 = [\cdot 88, \cdot 89]/7 = [\cdot 12571\ 42857, \cdot 12714\ 28572] \\
I_5 &= (1 - I_6)/6 = [\cdot 87285\ 71428, \cdot 87428\ 57143]/6 = [\cdot 14547\ 61904, \cdot 14571\ 42858] \\
I_4 &= (1 - I_5)/5 = [\cdot 17085\ 71428, \cdot 17090\ 47620] \\
I_3 &= [\cdot 20727\ 38095, \cdot 20728\ 57143] \\
I_2 &= [\cdot 26423\ 80285, \cdot 26424\ 20635] \\
I_1 &= [\cdot 36787\ 89682, \cdot 36788\ 09858] \\
I_0 &= [\cdot 63211\ 90142, \cdot 63212\ 10318]
\end{aligned}$$

Again, these intervals contain both the exact results and the results of Table 2.

Notice that in this example the calculated intervals decreased in width from $\cdot 01$ for I_7 to $\cdot 000002$ for I_0 and indicated a gain in accuracy as the computation proceeded whereas the intervals calculated for the previous example, (see Table 3), grew in width and indicated a loss of accuracy as that computation proceeded.

Calculations with intervals can easily be programmed for the computer [36], [40], [22], [8], [12].

In fact, using the CDC 1604 computer at the University of Wisconsin, the following table of intervals containing I_n was computed using rounded interval arithmetic[4]) for the first example discussed above, namely

$$I_0 = 1 - 1/e\ ; \quad I_n = 1 - nI_{n-1}\ , \quad n = 1, 2, \ldots, 14\ .$$

The results are given in the form $aEe = a\ .\ 10^e$.

[4]) In our program, [40] we did not bother to account for the error in converting the final machine results from binary to decimal before printing. This can be corrected by pessimistically adding one to the eleventh decimal digit of all the upper bounds given here and subtracting one from the eleventh digit of the lower bounds.

n	Lower bound	Upper bound
0	6·32120 558806E-01	6·32120 558911E-01
1	3·67879 441072E-01	3·67879 441177E-01
2	2·64241 117629E-01	2·64241 117850E-01
3	2·07276 646432E-01	2·07276 647090E-01
4	1·70893 411615E-01	1·70893 414249E-01
5	1·45532 928736E-01	1·45532 941908E-01
6	1·26802 348529E-01	1·26802 427575E-01
7	1·12383 006953E-01	1·12383 560275E-01
8	1·00931 517777E-01	1·00935 944353E-01
9	9·15765 007958E-02	9·16163 399816E-02
10	8·38366 001844E-02	8·42349 920422E-02
11	7·34150 875360E-02	7·77973 979712E-02
12	6·64312 243462E-02	1·19018 949565E-01
13	−5·47246 344376E-01	1·36394 083497E-01
14	−9·09517 168975E-01	8·66144 882143E-00

The second example discussed above was also programmed and (cf. table 2.5 in [6]) was recomputed using interval arithmetic beginning with

$$I_j = [0, 1/(j+1)] \quad \text{for} \quad j = 9, 19, 29, 39, 59$$

and computing

$$I_{n-1} = (1 - I_n)/n\,, \quad n = j\,,\ j-1, \ldots, 2, 1\,.$$

We will reproduce some of the results here for $j = 9$, 59.

$J = 9$

$n+1$	Lower bound	Upper bound
10	0	1·00000 000000E-01
9	9·99999 999953E-02	1·11111 111109E-01
8	1·11111 111107E-01	1·12500 000000E-01
7	1·26785 714278E-01	1·26984 126985E-01
6	1·45502 645496E-01	1·45535 714287E-01
5	1·70892 857134E-01	1·70899 470901E-01
4	2·07275 132264E-01	2·07276 785711E-01
3	2·64241 071412E-01	2·64241 622575E-01
2	3·67879 188701E-01	3·67879 464285E-01
1	6·32120 535697E-01	6·32120 811287E-01

$J = 59$

$n+1$	Lower bound	Upper bound
60	0	1·66666 666669E-02
59	1·66666 666657E-02	1·69491 525325E-02
58	1·69491 525416E-02	1·69540 229885E-02
57	1·72464 206483E-02	1·72465 060963E-02
56	1·75491 695336E-02	1·75491 710601E-02
55	1·78627 423436E-02	1·78627 423727E-02
54	1·81877 269922E-02	1·81877 269942E-02

$J = 59$

$n+1$	Lower bound	Upper bound
53	1·85247 598667*E*-02	1·85247 598682*E*-02
52	1·88745 238484*E*-02	1·88745 238492*E*-02
51	1·92377 544340*E*-02	1·92377 544349*E*-02
50	1·96152 449102*E*-02	1·96152 449114*E*-02
40	2·44044 317823*E*-02	2·44044 317835*E*-02
30	3·22906 775342*E*-02	3·22906 775359*E*-02
20	4·77227 557939*E*-02	4·77227 557974*E*-02
10	9·16122 929845*E*-02	9·16122 929915*E*-02
9	1·00931 967440*E*-01	1·00931 967446*E*-01
8	1·12383 504066*E*-01	1·12383 504069*E*-01
7	1·26802 356556*E*-01	1·26802 356562*E*-01
6	1·45532 940567*E*-01	1·45532 940576*E*-01
5	1·70893 411877*E*-01	1·70893 411888*E*-01
4	2·07276 647020*E*-01	2·07276 647026*E*-01
3	2·64241 117646*E*-01	2·64241 117658*E*-01
2	3·67879 441165*E*-01	3·67879 441171*E*-01
1	6·32120 558806*E*-01	6·32120 558829*E*-01

The following computations, also chosen from [6], were also run on the computer using properly rounded interval arithmetic.

Put $z_1 = 1$; compute

$$y_n = z_n/n\,, \quad z_{n+1} = ny_n\,, \quad n = 1, 2, \ldots, 10000\,.$$

The computer obtained the result

$$y_{10000} \times 10^4 \in [\cdot 99999\ 9621534,\ 1{\cdot}00000\ 024744]\,.$$

Putting $y_1 = 1$,

$$y_{n+1} = y_n v_n\,, \quad v_n = n/(n+1)\,, \quad n = 1, 2, \ldots, 10000$$

the computer obtained

$$y_{10000} \times 10^4 \in [\cdot 99999\ 9622745,\ 1{\cdot}00000\ 018076]\,.$$

3. INTERVAL METHODS FOR COMPUTERS

3.1. Introduction. The evaluation of any rational expression, using interval arithmetic, for given intervals of values of the real arguments can be carried out by the computer to obtain upper and lower bounds on the range of values of the real rational function defined by the expression.

Furthermore, it has been shown [36] that arbitrarily sharp upper and lower bounds on the range of values of a real rational function can be obtained by subdividing each of the intervals of values of arguments, evaluating the rational expression using interval arithmetic over each subdivision and forming the finite union, (and, of course, carrying enough digits in the computation).

An "extended" ("erweiterte") interval arithmetic based on arbitrary unions of intervals has recently been developed [2], [3], [4].

KRÜCKEBERG [31] considers "inner" and "outer" intervals and develops an arithmetic for both kinds of intervals on the extended real line. W. KAHAN (personal communication) has suggested a generalization of interval arithmetic to "interior" and "exterior" intervals.

A very frequently made suggestion is that of developing a *probabilistic* version of interval arithmetic. The argument goes like this: in a large number of practical applications of mathematics, guaranteed bounds on errors in numerical solutions of equations have no real meaning, since the mathematical equations themselves are only approximate descriptions of the real physical process involved. Therefore (continues the argument) it would make better sense to estimate intervals which "contain" values of exact solutions with high *probability.*

An interesting attempt along these lines is the current work of A. CHAI, a doctoral candidate at the University of Wisconsin, based on computing with *approximating* normal distributions represented by the number pairs: (mean, variance). Numerical experiments on the computer have yielded encouraging results. A very great difficulty encountered in such work, however, is that of establishing a precise *interpretation* of the numerical results, once an algorithm has been chosen; that is, in what *sense* precisely is an exact result contained with a certain probability in a computed "probabilistic interval" and what value should be assigned to that probability. See also [24], [29], [41].

I would like to cite an occurence that took place in a research laboratory some years ago. A physicist who was puzzled by some numerical results he had obtained from the computer came to me one day. He told me he didn't believe the results and that he suspected "round-off error" as the source of the trouble. The computation was quite involved and it would have been difficult and time consuming (but not impossible, of course) to analyse the round-off error accumulation by pencil and paper methods. Instead, we put the same computation on the computer a second time but using rounded interval arithmetic. The bounds on round-off error computed in this way did not exceed one in the sixth decimal digit of any result and round-off error was *ruled out* as the source of the trouble. It turned out later that the mathematical description of the physical process had been inadequate.

The degree of sharpness of bounds obtained using rounded interval arithmetic on the computer depends upon, among other things, the number of occurrences of a given variable, [36]. In computations involving a large number of occurrences of each of many independent variables and involving lots of *subtractions,* the intervals may grow very wide and give very pessimistic bounds. Direct evaluation of a rational expression in interval numbers will give the same result as if the interval numbers each represented the range of values of a distinct real variable. Thus cancellation of error by subtraction for example, is lost. An example of such a computation is the inversion of large matrices by use of Gaussian elimination. For such computations the interval bounds on accumulated round-off error seem from experiments to be about as pessimistic as the a priori bounds of von Neumann and Goldstine [24] who summarized their results as follows: "Matrices of order 15, 50, 150 can usually be inverted with a (relative) precision of 8, 10, 12 decimal digits less respectively than the number of digits carried throughout". Actual errors are often much smaller because of cancellations. Hansen [26] has combined interval arithmetic with Neumann series expansion about an approximate inverse for much sharper guaranteed bounds in numerical matrix inversion. The propagation of initial error is also taken into account in this method.

The sharpness of round-off error bounds obtained using interval arithmetic will depend upon the specific *form* of a rational expression, (i.e. the particular factorization and order of operations used in its evaluation). The actual error also depends on the specific form used, [36]. Interval arithmetic can be useful as a tool in the search for a particular way of evaluating a rational function on a computer which minimizes the error.

An important property of interval arithmetic is *inclusion monotonicity.* If $F(X_1, X_2, \ldots, X_n)$ is a rational expression in the interval variables $X_1, X_2, \ldots, X_n$ then $X_i' \subset X_i$, $i = 1, 2, \ldots, n$ implies $F(X_1', X_2', \ldots, X_n') \subset F(X_1, X_2, \ldots, X_n)$.

In particular, if $F(X)$ is a rational expression in X and if $F(X_0) \subset X_0$, *which can be tested by a computer*, then it follows that the sequence of intervals defined by

$$X^{(0)} = X_0\,, \quad X^{(k+1)} = F(X^{(k)})\,, \quad k = 0, 1, 2, \ldots$$

is a nested sequence $X^{(0)} \supset X^{(1)} \supset X^{(2)} \supset \ldots$ and hence *converges* to some interval X^* with $X^* \subset X^{(k)}$ for all k.

Of course inclusion monotonicity supposes exact (i.e., infinite precision decimal) arithmetic in the computation of end points of intervals. On the computer, using fixed (or limited) precision arithmetic properly rounded for the computation of end points of intervals it may happen that

$$X^{(1)} = F(X^{(0)})$$

is properly contained in $X^{(0)}$ but that for some k, $X^{(k+1)} \not\subset X^{(k)}$. We can program the iteration procedure to evaluate

$$X^{(k+1)} = X^{(k)} \cap F(X^{(k)})$$

and to stop when $X^{(k+1)} = X^{(k)}$; then the computer will have obtained the smallest interval it can find for the fixed precision used in the rounded interval arithmetic evaluation of F.

Furthermore, if $F(X) \cap X$ is empty, then X clearly contains no fixed points of F; i.e. there is no $x \in X$ such that $F(x) = x$.

A computer program for finding intervals guaranteed to contain a zero (or no zeros) of a real rational function has been written [14] using the above mentioned properties and an interval version of Newton's method, [36].

3.2 Algebraic systems. An interval version of Newton's method for solving systems of algebraic equations with guaranteed error bounding has been described [36]. Improvements have recently been made by Hansen [27], [28]. We now describe briefly the original version and the improvements. Suppose the system of algebraic equations is, in vector form, $f(x) = 0$, with components $f_i(x_1, x_2, \ldots, x_n) = 0$, $i = 1, 2, \ldots, n$, and that $x_0 = (x_{10}, x_{20}, \ldots, x_{n0})$. If $\varphi(s)$ is a continuously differentiable path from x_0 to x, with $\varphi(0) = x_0$, $\varphi(1) = x$, and $g_i(s) = f_i(\varphi(s))$ then by the mean value theorem

$$g_i(1) = f_i(x) = 0 = f_i(x_0) + g_i'(s_i)$$

for some $s_i \in [0, 1]$. Now

$$g_i'(s_i) = \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j} (\varphi(s_i))\, \varphi_j'(s_i)$$

and if the path is chosen as the "diagonal" line segment connecting x_0 and x, namely $\varphi(s) = (1 - s)\, x_0 + sx$ then $\varphi_j'(s_i) = x_j - x_{j0}$. Therefore, if X_i, $i = 1, 2, \ldots, n$ are intervals which contain *both* the corresponding solution components x_i and the components of the approximate solution x_{i0} we can conclude that $\varphi_i([0, 1]) \subset X_i$, $i = 1, 2, \ldots, n$, and the solution $\bar{X}_1, \bar{X}_2, \ldots, \bar{X}_n$ of the linear algebraic system with interval coefficients

$$(3.2\text{-}1) \quad f_i(x_0) + \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j} (X_1, X_2, \ldots, X_n)\, (x_j - x_{j0}) = 0, \quad i = 1, 2, \ldots, n$$

will also contain the solution point x. If V is a matrix of intervals which each contain all the values of the corresponding elements of the inverse of any real matrix with coefficients in the intervals $(\partial f_i / \partial x_j)$ then $x \in \bar{X} \subset x_0 - Vf(x_0)$.

Hansen [28] observes that the evaluation of the Jacobian matrix $\partial f_i / \partial x_j$ over the entire n-dimensional rectangle (interval vector) $X_1, X_2, \ldots, X_n$ is unnecessary and that sharper results (faster convergence, narrower resulting interval bounds) can be obtained by using a path going along edges of an n-dimensional rectangle from x_0 to x in n steps applying the mean value theorem in one coordinate direction at a time. In this way many of the arguments of the matrix elements $\partial f_i / \partial x_j$ are real numbers instead of intervals. Furthermore, instead of computing the interval matrix V containing the inverses of the Jacobian matrices with elements ranging over intervals, the linear system (3.2-1) can be solved directly for $\bar{X}$, [27] again improving both speed and accuracy (sharpness of bounding intervals).

We will illustrate these remarks with a simple example.

For the algebraic system

$$(3.2\text{-}2) \qquad f_1(x_1, x_2) = x_1^2 - x_2^2 + 1 = 0\,, \quad f_2(x_1, x_2) = x_1 x_2 - 1 = 0\,,$$

we have the Jacobian matrix

$$\left(\frac{\partial f_i}{\partial x_j}\right) = \begin{pmatrix} 2x_1 & -2x_2 \\ x_2 & x_1 \end{pmatrix}.$$

Suppose that x_{10}, $x_1 \in X_1$ and x_{20}, $x_2 \in X_2$ and call $f_{10} = x_{10}^2 - x_{20}^2 + 1$, $f_{20} = x_{10}x_{20} - 1$, then the two forms of the mean value theorem discussed lead to the linear systems.

(3.2-3)
$$\begin{aligned} f_{10} + 2X_1(\bar{x}_1 - x_{10}) - 2X_2(\bar{x}_2 - x_{20}) &= 0 \\ f_{20} + X_2(\bar{x}_1 - x_{10}) + X_1(\bar{x}_2 - x_{20}) &= 0 \end{aligned}$$
(for the "diagonal" path)

(3.2-4)
$$\begin{aligned} f_{10} + 2X_1(\bar{x}_1 - x_{10}) - 2X_2(\bar{x}_2 - x_{20}) &= 0 \\ f_{20} + X_2(\bar{x}_1 - x_{10}) + x_{10}(\bar{x}_2 - x_{20}) &= 0 \end{aligned}$$
(using the edges of (X_1, X_2))

Now take $x_{10} = {\cdot}9$, $x_{20} = 1{\cdot}1$, $X_1 = [0, {\cdot}9]$ $X_2 = [1{\cdot}1, 2]$ then $f_{10} = {\cdot}6$, $f_{20} = -{\cdot}01$ and (3.2-3) has the form

$$\begin{aligned} {\cdot}6 + [0, 1{\cdot}8](\bar{x}_1 - {\cdot}9) - [2{\cdot}2, 4](\bar{x}_2 - 1{\cdot}1) &= 0 \\ -{\cdot}01 + [1{\cdot}1, 2](\bar{x}_1 - {\cdot}9) + [0, {\cdot}9](\bar{x}_2 - 1{\cdot}1) &= 0 \end{aligned}$$

That is to say we want to find intervals $\bar{X}_1$, $\bar{X}_2$ such that $\bar{x}_1 \in \bar{X}_1$ and $\bar{x}_2 \in \bar{X}_2$ for all solutions $\bar{x}_1$, $\bar{x}_2$ of systems

$$\begin{aligned} a(\bar{x}_1 - {\cdot}9) - b(\bar{x}_2 - 1{\cdot}1) &= -{\cdot}6 \\ c(\bar{x}_1 - {\cdot}9) + d(\bar{x}_2 - 1{\cdot}1) &= {\cdot}01 \end{aligned}$$

for which $a \in [0, 1{\cdot}8]$, $b \in [2{\cdot}2, 4]$, $c \in [1{\cdot}1, 2]$, $d \in [0, {\cdot}9]$. Using an interval version of Gaussian elimination and "pivoting" on c we find that

$$\begin{aligned} \bar{x}_2 - 1{\cdot}1 &= ({\cdot}6c + {\cdot}01a)/(bc + ad) \\ \bar{x}_1 - {\cdot}9 &= ({\cdot}01 - d(x_2 - 1{\cdot}1))/c \end{aligned}$$

and

$$bc + ad \in ([2{\cdot}2, 4][1{\cdot}1, 2] + [0, 1{\cdot}8][0, {\cdot}9]) = [2{\cdot}42, 9{\cdot}62]$$

and

$${\cdot}6c + {\cdot}01a \in ({\cdot}6[1{\cdot}1, 2] + {\cdot}01[0, 1{\cdot}8]) = [{\cdot}66, 1{\cdot}218]$$

so

$$\bar{x}_2 - 1{\cdot}1 \in [{\cdot}66, 1{\cdot}218]/[2{\cdot}42, 9{\cdot}62] = [{\cdot}068\ldots, {\cdot}50\ldots]$$

and

$$\bar{x}_2 \in [1{\cdot}168\ldots, 1{\cdot}60\ldots].$$

Similarly

$$\bar{x}_1 - {\cdot}9 \in ({\cdot}01 - [0, {\cdot}9][{\cdot}068, {\cdot}51])/[1{\cdot}1, 2]$$

and so

$$\bar{x}_1 \in [{\cdot}491\ldots, {\cdot}909\ldots].$$

Therefore (3.2-3) leads to the conclusion that a solution of (3.2-2) which is in the rectangle

$$(x_1, x_2) \in ([0, \cdot 9], [1{\cdot}1, 2])$$

is also in the smaller rectangle

$$(x_1, x_2) \in ([\cdot 491, \cdot 91], [1{\cdot}168, 1{\cdot}61]) .$$

Now repeating the calculations using (3.2-4) instead of (3.2-3), (which just amounts in this example to replacing the statement $d \in [0, \cdot 9]$ by $d = \cdot 9$) we obtain the slightly improved result

$$(x_1, x_2) \in ([\cdot 491, \cdot 875], [1{\cdot}168, 1{\cdot}61]) .$$

For other (and especially higher dimensional) examples the improvement can be more marked.

The computations in this example amount to one iteration of the interval version of Newton's method. The whole process can be iterated and will converge rapidly to a bounding rectangle of arbitrarily small width, limited only by the number of digits carried in the interval arithmetic operations.

Applications of the interval version of Newton's method for algebraic systems have been made to: the determination by the computer of arbitrarily small rectangles in the complex plane guaranteed to contain complex roots of polynomials, and machine computed upper and lower bounds on eigenvalues and eigenvectors of matrices, [28]. Interval versions of Wielandt's "inverse iteration" and methods of Wilkinson are also discussed by HANSEN [28].

3.3. Quadrature. Interval quadrature methods have been developed based on Taylor series with remainder [36]. Specifically, if $F^{(r)}$, $r = 0, 1, 2, \ldots, k$ are rational interval extensions of the real rational function $f = f^{(0)}$ and its first k derivatives and if $I = \int_{[a,b]} f(x)\,\mathrm{d}x$ and

$$I_{n,k} = \sum_{i=1}^{n} \sum_{r=0}^{k-1} \frac{F^{(r)}(x_{i-1})}{(r+1)!} (x_i - x_{i-1})^{r+1} + \frac{1}{(k+1)!} \sum_{i=1}^{n} F^{(k)}([x_{i-1}, x_i])\,(x_i - x_{i-1})^{k+1} \tag{3.3-1}$$

with $a = x_0 < x_1 < \ldots < x_n = b$ then for all $n, k \geqq 1$ we have $I \in I_{n,k}$ and the *width* of the interval $I_{n,k}$ satisfies

$$w(I_{n,k}) \leqq L_k \max_{i=1,2,\ldots,n} (x_i - x_{i-1})^{k+1} .$$

A sharper result pointed out by E. Hansen (private communication) is

$$I \in I^*_{n,2k} = \sum_{i=1}^{n} \sum_{r=0}^{k-1} \frac{2}{(2r+1)!} F^{(2r)}\left(\frac{x_i + x_{i-1}}{2}\right)\left(\frac{x_i - x_{i-1}}{2}\right)^{r+1} + \frac{2}{(2k+1)!}\left(\frac{x_i - x_{i-1}}{2}\right)^{2k+1} F^{(2k)}([x_{i-1}, x_i]) . \tag{3.3-2}$$

An interval version of Gaussian quadrature formulas with remainder has been developed also [34].

An interval "cubature" method for multidimensional iterated integrals based on Taylor series or Gaussian quadrature should not present any insurmountable problems in programming since computer programs for generating partial derivatives of any order are available, [39].

The computer generated subroutines for derivatives can be executed in interval arithmetic, so that the computer can produce intervals containing such terms in the formulas we have been discussing as $\partial f_i/\partial x_j\ (X_1, X_2, \ldots, X_n)$ and $F^{(2k)}([x_{i-1}, x_i])$, [36], [38], [39].

An alternative to the direct evaluation of remainder terms in higher derivative form using computation with intervals is the use of Cauchy's inequality. Recall that if f is analytic in the open disc $|z - z_0| < D$, then

$$\left|\frac{f^{(r)}(z_0)}{r!}\right| \leqq \frac{\max |f(z_0 + \varrho e^{i\theta})| \quad \text{for} \quad \theta \in [0, 2\pi]}{\varrho^r} \tag{3.3-3}$$

for any $0 < \varrho < D$.

By treating real and imaginary parts separately, complex functions can be bounded in rectangles in the complex plane using interval computation [8], [28].

An optimization problem arises in connection with formulas such as (3.3-2). For a fixed choice of $\{x_i\}$, depending only on n, both the *computation time* for evaluating $I^*_{n,2k}$ and the *width* of the interval $I^*_{n,2k}$ depend on the two approximation parameters n and k. The question is for a given width of $I^*_{n,2k}$ what choice of n and k minimizes computation time? A discussion of this question for formula (3.3-1) is given in [36]. We will take up similar questions in connection with methods for differential equations later in this paper. The question of the "best" choice of $\{x_i\}$ will not be considered in this paper, except for the following remark. A popular opinion, based on experimental evidence in practical computation is that an efficient scheme for variable step size in numerical integration (quadrature as well as numerical solution of ordinary differential equations) is one in which some estimate of the local truncation error is kept approximately constant; for formula (3.3-2), this would mean choosing $\{x_i\}$ such that

$$\left(\frac{x_i - x_{i-1}}{2}\right)^{2k+1} \left\{\frac{w(F^{(2k)}([x_{i-1}, x_i]))}{(2k+1)!}\right\}$$

is roughly independent of i. For this, $w(F^{(2k)}([x_{i-1}, x_i]))$ could be estimated by $|F^{(2k+1)}(x_{i-1})|\,(x_i - x_{i-1})$. Given x_0, x_1 we could then put

$$x_i = x_{i-1} + \left|\frac{F^{(2k+2)}(x_0)}{F^{(2k+1)}(x_{i-1})}\right|^{1/(2k+2)}.$$

We will now illustrate some of the methods and considerations just mentioned by detailed examination of a specific example.

We will study the application of (3.3-2) to the integral

$$I = \int_1^b (1/x)\,\mathrm{d}x = \ln b\,. \tag{3.3-4}$$

We put $f(x) = 1/x$, then $xf = 1$ and by Leibniz' formula (which is used by the computer programs for generating derivative subroutines, [36]) we have

$$(xf)^{(r)} = \sum_{j=0}^{r} \binom{r}{j} x^{(j)} f^{(r-j)} = 0\,, \quad r \geqq 1\,.$$

Solving for $f^{(r)}/r!$ we obtain (as the computer does), making use of the additional information $x^{(1)} = 1$, $x^{(r)} = 0$, $r \geqq 2$, the result

$$\frac{f^{(r)}(x)}{r!} = -\left(\frac{f^{(r-1)}(x)}{(r-1)!}\right)\Big/x\,, \quad r = 1, 2, \ldots \tag{3.3-5}$$

The programs referred to seek recursion relations (such as (3.3-5)) defining higher derivatives in terms of lower order ones. The interval extensions of $f^{(r)}$, $r = 0, 1, 2, \ldots$ are then simply defined by interpreting the operations in (3.3-5) and $f(x) = 1/x$ as interval arithmetic operations. Thus, for $F^{(2k)}([x_{i-1}, x_i])$ in (3.3-2) we have in this example, from (3.3-5),

$$\frac{F^{(2k)}([x_{i-1}, x_i])}{(2k)!} = -\left(\frac{F^{(2k-1)}([x_{i-1}, x_i])}{(2k-1)!}\right)\Big/[x_{i-1}, x_i] \tag{3.3-6}$$

$$= -[1/x_i, 1/x_{i-1}]\frac{F^{(2k-1)}([x_{i-1}, x_i])}{(2k-1)!} = (-1)^{2k}[1/x_i, 1/x_{i-1}]^{2k+1}\,.$$

Note that this means, in particular,

$$\frac{f^{(2k)}(\xi)}{2k!} \in (-1)^{2k}[1/x_i, 1/x_{i-1}]^{2k+1} \quad \text{for any} \quad \xi \in [x_{i-1}, x_i]\,. \tag{3.3-7}$$

Let us now compare this part of our numerical illustration with an alternative approach using Cauchy's inequality.

For this, we put $f(z) = 1/z$ and representing complex numbers by real pairs we have for $z = (x, y)$ and $\theta \in [0, 2\pi]$ that $z + \varrho e^{i\theta} = (u, v)$ is contained in a rectangle in the complex plane defined by $(u, v) \in z + ([-\varrho, \varrho], [-\varrho, \varrho]) = (x + [-\varrho, \varrho], y + [-\varrho, \varrho])$. In order to bound $f^{(2k)}(\xi)/2k!$ for any $\xi \in [x_{i-1}, x_i]$ using Cauchy's inequality (3.3-3) we need to bound $\max |f(z + \varrho e^{i\theta})|$ for $\theta \in [0, 2\pi]$ and for z in the segment $[x_{i-1}, x_i]$. Then $z + \varrho e^{i\theta} = (u, v)$ will lie in $([x_{i-1}, x_i], 0) + ([-\varrho, \varrho], [-\varrho, \varrho]) = ([x_{i-1} - \varrho, x_i + \varrho], [-\varrho, \varrho])$ and

$$f(z + \varrho e^{i\theta}) = 1/(z + \varrho e^{i\theta}) = 1/(u + iv) = \left(\frac{u}{u^2 + v^2}, \frac{-v}{u^2 + v^2}\right)$$

will lie in

$$\left(\frac{[x_{i-1} - \varrho, x_i + \varrho]}{[x_{i-1} - \varrho, x_i + \varrho]^2 + [0, \varrho^2]}, \frac{[-\varrho, \varrho]}{[x_{i-1} - \varrho, x_i + \varrho]^2 + [0, \varrho^2]}\right)$$
$$= \left(\left[\frac{x_{i-1} - \varrho}{(x_i + \varrho)^2 + \varrho^2}, \frac{x_i + \varrho}{(x_{i-1} - \varrho)^2}\right], \left[\frac{-\varrho}{(x_{i-1} - \varrho)^2}, \frac{\varrho}{(x_{i-1} - \varrho)^2}\right]\right)$$

for any $0 < \varrho < x_{i-1}$. Therefore

(3.3-8) $$\left|\frac{f^{(2k)}(\xi)}{2k!}\right| \leq \frac{((x_i + \varrho)^2 + \varrho^2)^{1/2}}{\varrho^{2k}(x_{i-1} - \varrho)^2}$$

for any $\xi \in [x_{i-1}, x_i]$ and any $0 < \varrho < x_{i-1}$.

We could program the minimization of the right hand side of (3.3-8) with respect to admissible choices of ϱ for particular numerical values of x_{i-1} and x_i and k. Doing this we would obtain $\varrho = \theta x_{i-1}$ for some $0 < \theta < 1$ and (3.3-8) would become

(3.3-9) $$\left|\frac{f^{(2k)}(\xi)}{2k!}\right| \leq \frac{((x_i + \theta x_{i-1})^2 + \theta^2 x_{i-1}^2)^{1/2}}{(1 - \theta)^2\, \theta^{2k} x_{i-1}^{2k+2}}.$$

The bound given by (3.3-9) will not be as sharp as that given by (3.3-7).

Using (3.3-6) we obtain from (3.3-2) for the width of $I^*_{n,2k}$, (ignoring round-off, for the moment)

$$w(I^*_{n,2k}) = \frac{2}{2k+1} \sum_{i=1}^{n} \left(\frac{x_i - x_{i-1}}{2}\right)^{2k+1} \left(\frac{1}{x_{i-1}^{2k+1}} - \frac{1}{x_i^{2k+1}}\right) \leqq 4 \sum_{i=1}^{n} \left(\frac{x_i - x_{i-1}}{2x_{i-1}}\right)^{2k+2}.$$

According to a previous remark on schemes for variable step size we set

$$\frac{x_i - x_{i-1}}{2x_{i-1}} = h.$$

Then $x_i = x_{i-1} + 2hx_{i-1} = (1 + 2h)\, x_{i-1}$ and $x_i = (1 + 2h)^i\, x_0$. For our problem

$$I = \int_1^b (1/x)\, \mathrm{d}x$$

so we put $x_0 = 1, x_n = b$ and obtain $(1 + 2h)^n = b$ or $h = \frac{1}{2}(b^{1/n} - 1)$ and $x_i = b^{i/n}$. With this choice we have

(3.3-10) $$w(I^*_{n,2k}) < 4n\left(\frac{b^{1/n} - 1}{2}\right)^{2k+2}.$$

We will now consider briefly the problem of minimizing for $\varepsilon > 0$ the computation time required to evaluate (3.3-2) among choices of n and k for which $4n\,(\frac{1}{2}(b^{1/n} - 1))^{2k+2} \leqq \varepsilon$.

The next remark to be made is this. A common practice in predicting computing time is to count arithmetic operations in a formula to be evaluated. This is not too bad a procedure, but it should always be realized that actual numerical evaluation of a formula by a computer involves many other operations as well: storing partial results, modifying addresses of instructions, etc. In a "typical" situation the computation time may be *proportional* to the number of arithmetic operations but the total time might be five times the time for executing the arithmetic operations alone.

At this point we will suppose that all arithmetic operations are to be carried out by the computer in rounded interval arithmetic so that round-off error will be accounted for in the final result.

From (3.3-5), (3.3-6) and (3.3-2) we determine that the evaluation of $I^*_{n,2k}$ requires $n(2k-1)$ divisions to get $F^{(2r)}(\frac{1}{2}(x_i + x_{i-1}))$ for $r = 0, 1, 2, \dots, k-1$, $i = 1, 2, \dots, n$; another $n(2k+1)$ divisions to get $F^{(2k)}([x_{i-1}, x_i])$, $i = 1, 2, \dots, n$; plus an additional $n(2k+2)$ multiplications and additions (using the "nested" form of evaluating the polynomial) and nk divisions to complete the evaluation of $I^*_{n,2k}$. Thus, the computation time can be assumed to be proportional, for this example, to nk.

In order to see which choices of n, k minimize nk subject to

$$4n\left(\frac{b^{1/n}-1}{2}\right)^{2k+2} \leqq \varepsilon, \tag{3.3-11}$$

for some small value of ε, say $\varepsilon = 10^{-10}$ and for various values of b, we notice first that we certainly must have $b^{1/n} < 3$, otherwise no choice of k will work.

A systematic way of searching for n and k might be to start with the smallest $n \geqq \ln b/\ln 3$, find the smallest k satisfying (3.3-11) then add one to n and find the smallest k again, proceeding in this way until the product nk has passed through a minimum value.

As Babuška and Sobolev [5] have pointed out one should really take into account, in such deliberations, the time required to *obtain* the optimal choice of parameters. Even for such a simple example as the one we have been discussing, the determination of the optimal choice of parameters can be disturbingly complicated.

A reasonably good choice for most practical purposes is to set the number of terms carried in the Taylor expansion equal to the number of decimal digits of accuracy sought, or, say, the number of decimal digits carried in the arithmetic used, [36].

In this example, if the computations are carried out using about 10 decimal digit rounded interval arithmetic, then 10^{-10} is certainly as small as is reasonable for ε and we could put $2k = 10$ or $k = 5$ and solve for the smallest n such that

$$4n\left(\frac{b^{1/n}-1}{2}\right)^{12} \leqq 10^{-10}.$$

By rough slide rule calculation, I predict for example, that for $b = 2$ this gives about $n = 3$ and for $b = 100$, about $n = 25$.

A rough sketch of the dependence of nk on n, k and b from (3.3-11) with $\varepsilon = 10^{-10}$ is given in the following tables. The figures given are only approximations.

$b = 2$

n	k	nk
1	17	17
2	6	12
3	6	18
850	1	850

$b = 100$

n	k	nk
5	45	225
6	19	114
10	10	100
25	5	125
28 000	1	28 000

A useful thing about an interval method such as formula (3.3-2) is that *whatever choice* of n and k we (or the computer) make, the computed interval will contain the exact result for certain and if the interval result is too wide for our purpose we can get a narrower one by repeating the calculation with more steps (larger n) and by using higher precision arithmetic (more digits) if necessary.

An application of interval computation by the computer which should not be overlooked is in the preparation or checking of published tables of numerical values of functions. I have not undertaken this in any extensive way. However, I have checked some interval results against tables and have noticed occasional errors in the last digit or two of a published result.

3.4. Integral equations and two point boundary value problems. In this section we will use the following type of notation for the interval extension of real rational functions $f(x, y) = x + x/y \rightarrow F(X, Y) = X + X/Y$. From inclusion monotonicity and the mean value theorem, it follows that

$$\int_{[a,b]} f(x')\,\mathrm{d}x' \in \sum_{i=1}^{n} F(X_i^{(n)}) \left(\frac{b-a}{n}\right) \tag{3.4-1}$$

where

$$X_i^{(n)} = a + [i-1, i]\left(\frac{b-a}{n}\right).$$

If F is an interval function whose real restriction, $F([x, x]) = F(x)$, is still interval-valued, we define

$$\int_{[a,b]} F(x)\,\mathrm{d}x = \bigcap_{n=1}^{\infty} \sum_{i=1}^{n} F(X_i^{(n)}) \left(\frac{b-a}{n}\right). \tag{3.4-2}$$

Consider an integral equation of the form

(3.4-3) $$y(x) = h(x) + \int_{[a,b]} g(x, x', y(x'))\,\mathrm{d}x' \quad \text{for} \quad a \leqq x \leqq b\,.$$

Define a sequence of interval-valued functions

$$Y_0(x) \equiv [c, d]$$

(3.4-4) $$Y_{m+1}(x) = h(x) + \int_{[a,b]} G(x, x', Y_m(x'))\,\mathrm{d}x'\,, \quad m = 1, 2, \ldots$$

If $Y_1(x) \subset [c, d]$ for all $x \in [a, b]$, then for each $x \in [a, b]$, we have

(3.4-5) $$y(x) \in Y_{m+p}(x) \subset Y_m(x)\,, \quad (m, p = 0, 1, 2, \ldots)\,.$$

Let n be a positive integer; for $i = 1, 2, \ldots, n$; $m = 0, 1, \ldots,$ define

$$Y_{0,i}^{(n)} \equiv [c, d]$$

(3.4-6) $$Y_{m+1,i}^{(n)} = H(X_i^{(n)}) + \sum_{j=1}^{i} W_{i,j}^{(n)}\, G(X_i^{(n)}, X_j^{(n)}, Y_{m,j}^{(n)})$$

where

$$W_{i,j}^{(n)} = \begin{cases} (b - a)/n, & \text{if} \quad i \neq j\,, \\ [0, (b - a)/n], & \text{if} \quad i = j\,. \end{cases}$$

If H and G are rational interval functions, with real restrictions h and g, then a computer program can be written for the evaluation of $Y_{m,i}^{(n)}$ using rounded interval arithmetic.

If $Y_{1,1}^{(1)} \subset [c, d]$, (which can be tested by the computer), then

$$y(x) \in Y_m(x) \subset Y_{m,i}^{(n)}$$

for all $x \in X_i^{(n)}$; $m, n = 1, 2, \ldots$

At "mesh points" $x_i^{(n)} = X_i^{(n)} \cap X_{i+1}^{(n)}$, we have the sharper result, [36],

(3.4-7) $$y(x_i^{(n)}) \in Y_{m,i}^{(n)} \cap Y_{m,i+1}^{(n)}\,.$$

If the number of places in the rounded arithmetic used is increased with n and m, then for rational $(x - a)/(b - a) = M/N$, we have convergence to real values:

(3.4-8) $$y(x) = \bigcap_{\substack{n \geqq N \\ m \geqq 0}} Y_{m,n((x - a)/(b - a))}^{(n)}\,.$$

The two point boundary value problem

(3.4-9) $$y'' = f(x, y)\,, \quad y(a) = y_0\,, \quad y(b) = y_1\,,$$

which can be put into the form of the above integral equation, can also be treated in this way; if f is rational, then the corresponding g is piecewise rational.

Consider the boundary value problem

$$y'' = t^2 + y^2\,, \quad y(0) = y(1) = 0\,. \tag{3.4-10}$$

It is sometimes useful to note that (3.4-2) gives for the *interval-valued* function $F(t) = [f_1(t), f_2(t)]$, the integral

$$\int F(t)\,\mathrm{d}t = \left[\int f_1(t)\,\mathrm{d}t,\ \int f_2(t)\,\mathrm{d}t\right]. \tag{3.4-11}$$

We can define (and program) the *square* of an interval by

$$[a, b]^2 = \begin{cases} [a^2, b^2]\,, & a > 0\,, \\ [b^2, a^2]\,, & b < 0\,, \\ [0, \max(a^2, b^2)]\,, & a < 0 < b\,. \end{cases}$$

Now write the boundary value problem as an integral equation

$$y(t) = \int_0^1 K(t, t')\,(t'^2 + y^2(t'))\,\mathrm{d}t' \tag{3.4-12}$$

with

$$K(t, t') = \begin{cases} (t - 1)\,t'\,, & 0 \leqq t' \leqq t\,, \\ (t' - 1)\,t\,, & t' < t < 1\,. \end{cases}$$

We define a sequence of interval valued functions as follows.

$$Y_0(t) \equiv [-1, 0]$$

$$Y_{k+1}(t) = \int_0^1 K(t, t')\,(t'^2 + Y_k^2(t'))\,\mathrm{d}t' \tag{3.4-13}$$

For this example, we will carry out directly some of the integration instead of using the discretized form given by (3.4-6).

By *interval computation*, using the above definitions, we find, remembering that $0 \leqq t \leqq 1$,

$$\begin{aligned} Y_1(t) &= \int_0^t (t - 1)\,t'(t'^2 + [0, 1])\,\mathrm{d}t' + \int_t^1 (t' - 1)\,t(t'^2 + [0, 1])\,\mathrm{d}t' \\ &= (t - 1)\left[\int_0^t t'^3\,\mathrm{d}t',\ \int_0^t (t'^3 + t')\,\mathrm{d}t'\right] \\ &\quad + t\left[\int_t^1 ((t' - 1)\,t'^2 + t' - 1)\,\mathrm{d}t',\ \int_t^1 (t' - 1)\,t'^2\,\mathrm{d}t'\right] \end{aligned} \tag{3.4-14}$$

$$= (1 - t)\left[-\frac{t^4}{4} - \frac{t^2}{2},\ -\frac{t^4}{4}\right]$$

$$+ t\left[-\frac{7}{12} + t - \frac{t^2}{2} + \frac{t^3}{3} - \frac{t^4}{4},\ -\frac{1}{12} + \frac{t^3}{3} - \frac{t^4}{4}\right]$$

$$= \left[-\frac{7t}{12} + \frac{t^2}{2} + \frac{t^4}{12},\ -\frac{t}{12} + \frac{t^4}{12}\right].$$

Now it happens that $Y_1(t) \subset Y_0(t) = [-1, 0]$ for all t in $[0, 1]$. From inclusion monotonicity it follows that for *all* $k = 1, 2, \ldots$, the interval valued functions defined by (3.4-13) satisfy $Y_{k+1}(t) \subset Y_k(t)$ for all t in $[0, 1]$ and furthermore if $Y_k(t) = [v_k(t), u_k(t)]$, then $u_k(t) - v_k(t)$ converges to zero for all t as k increases and the solution to the boundary value problem, $y(t)$, satisfies $y(t) \in Y_k(t)$ for *all* k and *all* t in $[0, 1]$.

Carrying out *part* of the calculation of $Y_2(t)$ we find that the part which is linear in t is given by

(3.4-15) $$Y_2(t) = \left[-\frac{1}{12}\left(1 + \frac{49}{144}\right),\ -\frac{1}{12}\left(1 + \frac{1}{144}\right)\right] t + \ldots$$

For $Y_1(t)$ we had

$$Y_1(t) = \left[-\frac{7}{12},\ -\frac{1}{12}\right] t + \ldots$$

This means that we have determined the following intervals containing the *initial slope* y_0' of the solution:

from $Y_1(t)$:

$$-\frac{7}{12} < y_0' < -\frac{1}{12},$$

from $Y_2(t)$:

$$-\frac{1}{12}\left(1 + \frac{49}{144}\right) < y_0' < -\frac{1}{12}\left(1 + \frac{1}{144}\right).$$

The *width* of the interval derived from $Y_1(t)$ is $\frac{1}{2}$ while that derived from $Y_2(t)$ has the smaller width $\frac{1}{36}$.

If $Y_k(t)$ gives $S_1^{(k)} < y_0' < S_2^{(k)}$, then a calculation such as that indicated above yields the following information:

The interval function $Y_{k+1}(t)$ will have the linear part

(3.4-16) $$Y_{k+1}(t) = \left[-\frac{1}{12}(1 + (S_1^{(k)})^2),\ -\frac{1}{12}(1 + S_2^{(k)})^2)\right] t + \ldots$$

so that $Y_{k+1}(t)$ yields the following bounds on y_0',

(3.4-17)
$$S_1^{(k+1)} < y_0' < S_2^{(k+1)}$$

where

$$S_1^{(k+1)} = -\frac{1}{12}\left(1 + (S_1^{(k)})^2\right), \quad S_2^{(k+1)} = -\frac{1}{12}\left(1 + (S_2^{(k)})^2\right).$$

Define $w_k = S_2^{(k)} - S_1^{(k)}$, then $w_{k+1} = -\frac{1}{12}(S_2^{(k)} + S_1^{(k)})\, w_k$. From $Y_{k+1}(t) \subset Y_k(t)$, it follows that

$$S_1^{(k)} < S_1^{(k+1)} < S_2^{(k+1)} < S_2^{(k)}$$

so for $k \geqq 2$, $S_2^{(k)} + S_1^{(k)}$ is very close to $-\frac{1}{6}$ and we have, approximately,

(3.4-18)
$$w_{k+1} = \frac{1}{72} w_k .$$

Thus $Y_4(t)$ should give upper and lower bounds to y_0' differing by less than 10^{-5}.

We point out this application of interval methods for two-point boundary problems since oftentimes it is of practical importance to compute bounds on the initial slope of the solution.

A more rapidly convergent sequence of interval valued functions containing the solution to an integral equation or a two point boundary value problem can be obtained using Newton's method and a combination of techniques from interval analysis and functional analysis. But we will defer a discussion of that until the last section of this paper.

3.5. The initial value problem in ordinary differential equations. There are several "standard" types of problems for which a large number of competitive computational algorithms have been derived and studied (and continue to be derived and studied).

Included among such types of problems are: systems of linear algebraic equations, algebraic (polynomial) equations, and the initial value problem in ordinary differential equations.

A specific algorithm for problems of a certain type often has some advantage over other algorithms for a certain subclass of the class of problems of the type in question.

In practice it is usually difficult to tell at a glance whether a specific problem falls in the subclass for which a given algorithm is "best" in some sense. For one thing, a precise and simple description of the subclass is rarely available. So the *choice* of which of a number of algorithms to use in each specific case is usually considered to be a matter for the judgment of an expert – a person experienced in the "art of computation".

To quote Babuška, Práger, and Vitásek:

"Given a mathematical problem and an automatic computer, it is required to select the best numerical method of solution," [6].

If we choose a precise enough and narrow enough definition of "best", then a selection process can be programmed for the computer as part of the computation [36]. If this is done, however, the time required to obtain the optimal choice should also be considered (as pointed out by Babuška and Sobolev [5]).

Sometimes, it doesn't make very much difference which of several comparable algorithms are used.

There have been a number of comparative numerical studies made in which several popular methods are tried on the same problem. In such studies, however, there is always some question about the generality of the conclusions drawn. It has often been conjectured that given two distinct numerical methods there are problems, perhaps artificially contrived, for which either method will give better results than the other. Still, it is of practical significance to compare actual running times on a computer required to obtain results of the same accuracy on some nontrivial problems of real interest using various available methods.

An interesting study of this kind was reported recently [21]. The differential equations used as the test problem were those for the restricted three body problem.

$$y_1'' = y_1 + 2y_2' - \frac{(1-\mu)(y_1+\mu)}{r_1^3} - \frac{\mu(y_1 - 1 + \mu)}{r_2^3} \tag{3.5-1}$$

$$y_2'' = y_2 - 2y_1' - \frac{(1-\mu)\,y_2}{r_1^3} - \frac{\mu y_2}{r_2^3}$$

with

$$r_1 = ((y_1+\mu)^2 + y_2^2)^{1/2}, \quad r_2 = ((y_1 - 1 + \mu)^2 + y_2^2)^{1/2}.$$

The methods compared were each used with some flexibility, varying step size and, when possible, even the *order* of the method in order to obtain something like an experimentally determined optimal use of the formula. A variety of initial conditions were used and a variety of "error tolerances" were used to choose step size. The computations were carried out using double precision arithmetic (about 22 decimals) on the Burroughs B-5500 computer. Another set of runs was made using 30 decimal place arithmetic on the IBM 7094 computer with the Runge-Kutta-Fehlberg method to obtain more accurate results for comparison. These are "believed" to be accurate to 24 significant figures. We will quote one of the tables of comparative results [21] corresponding to the initial conditions:

$$y_1 = 1{\cdot}2 \qquad y_1' = 0$$
$$y_2 = 0 \qquad y_2' = -1{\cdot}04935750983031990726$$
$$\mu = 0{\cdot}012128562765312310491 2068$$

"This table summarizes results of runs of comparable accuracy ($\sim 10^{-12}$) for the various methods." The solutions were carried out over an interval slightly larger than [0, 6] in the independent variable.

Number of steps	Average number of function evaluations per step	Processor time in seconds	Maximum error units of 10^{-12}	Order	Method
693	21	293	0·08	7—8	Runge-Kutta-Fehlberg
236	5	173	0·03	11	Runge-Kutta-Fehlberg
1912	3·1	296	0·2	12—13	Adams-Bashforth-Moulton
1959	3·1	330	0·01	13—16	Stoermer-Cowell
1893	3·2	133	1	12	Cowell, constant *N*th order difference
1126	3·1	177	0·8	11	Stetter-Gragg-Butcher

The figures given in the study for computer processing time do not appear to vary enormously – from a little over two minutes to five and a half minutes. On this basis the methods compared could be judged to be, more or less, equally good.

E. FEHLBERG has derived recently some very efficient algorithms, [16], [17], [18], [19]. One of these, referred to by Fehlberg as the "Runge-Kutta-Transformation" method, which combines Taylor series expansions and Runge-Kutta type formulas, is the algorithm referred to in the above table as the "Runge-Kutta-Fehlberg" method. GALLAHER and PERLIN, [21], conclude from their numerical studies that in several senses "the Runge-Kutta-Fehlberg method is probably superior" to the other methods they tried.

Fehlberg [16] has also run some timing studies on the same numerical example as used in the study quoted in the above table (involving equations (3.5-1) and the same initial conditions). We quote now some of the results of his study obtained using double precision arithmetic (about 16 decimals) on the IBM 7090 computer. Again the methods were programmed with "automatic step size control".

Number of steps	Processor time in seconds	Error estimates*)	Order	Method
17 750	592	$\cdot 28 \, . \, 10^{-10}$	4	Runge-Kutta-Nyström
726	106	$\cdot 25 \, . \, 10^{-10}$	8	Taylor series expansions
520	57	$\cdot 11 \, . \, 10^{-10}$	8	Runge-Kutta-Transformation
280	79	$\cdot 25 \, . \, 10^{-10}$	12	Taylor series expansions
176	42	$\cdot 18 \, . \, 10^{-10}$	12	Runge-Kutta-Transformation

*) Change in value of Jacobi integral.

Notice here the rather significant decrease in the computing time from the fourth order to the 12th order methods.

Suppose we consider a much larger class of methods for this same problem, including say Newton's method [30], Lie series [25], etc. Suppose we allow all sorts of transformations of the variables in the problem itself. And suppose we are to define "best" here in terms of processor time to obtain a solution of at least 12 decimal place accuracy. Clearly, if we carry out enough analysis and hand computation on the problem before we begin the machine computation we could, in principle, even *finish* the computation by hand and have the machine simply read in and print out the answers! Therefore a reasonable criterion for "best" when allowing preparatory analytical work must include some measure of the time and labor involved in that work. On the other hand, if it were possible for the *machine* to do practically all the "analytical" work: setting up transformations, choosing step size and order, etc., then processor time alone would be a reasonable criterion for "best".

It is my belief that the "art of computation" can eventually, to some extent be made into a "science of computation" by a formal algorithmic description of various analytical processes, transformation techniques and selection criteria; putting these descriptions into the form of computer programs; and letting the computer carry out the symbolic manipulations and computations leading to the selection of the "best (or, at least, a good) method of solution" for given a mathematical problem.

Sometimes we want the algorithm chosen to have special properties. This limits the selection to the available algorithms with those properties.

Suppose, for example, that we want an algorithm for calculating approximate solutions along with guaranteed error bounds with all the analysis and computation to be carried out by the computer.

For the initial value problem in ordinary differential equations a *family* of such algorithms has been derived and programmed for the computer based on repeated expansions in Taylor series truncated at the Kth term with the remainder in "mean-value form" to be bounded by interval computation, [36]. Recurrence formulas for the Taylor coefficients are derived by the computer in the form of subroutines. This is done once, for a given problem, during "compilation time", requires little computer time and *no* work or time on the part of the user. During the "execution" of the computation for a given problem the *time* $T(K)$ required by the computer to obtain values for the first K Taylor coefficients is proportional to K^2 for differential equations which are non-linear after reduction to autonomous systems of first order equations. The values $t_0, t_1, t_2, \dots$ of the independent variables at which successive Taylor expansions are to be carried out and the number of terms $K_0, K_1, K_2, \dots,$ to be carried respectively in each expansion are the parameters in the "family" of algorithms.

Alternatively we can think of this as a single algorithm if we add some procedure for choosing $t_0, t_1, t_2, \dots$ and $K_0, K_1, K_2, \dots$

The time required by the computer to obtain the approximation and error bounds at each t_{i+1} from its expansion of order K_i at t_i is roughly proportional to K_i^2, [36].

One criterion for "optimization" of the algorithm is the following: choose the values of t_i, K_i, $i = 0, 1, 2, \dots$ so that for maximum accuracy obtainable using a fixed precision machine arithmetic the total computation time is minimum.

It does not seem possible to mechanize this precise choice of t_i, K_i by any procedure which would involve an amount of computation which could be ignored in comparison to that required by the resulting optimal algorithm itself.

An approximation to this choice of t_i, K_i is evidently to set $K_0 = K_1 = \dots = K_i \approx \approx (1{\cdot}15\dots)\, d$ where d is the number of decimal places carried in the fixed precision machine arithmetic used; and to choose t_{i+1} so that, ([36] p. 102), the local truncation error is kept roughly constant relative to the change in solution values from step to step.

For the algorithm under discussion this can be programmed for the computer as:

$$t_{i+1} - t_i = \left(\frac{10^{-d}|y_i|}{|F^{(K)}(y_i)/K!|}\right)^{1/(K+1)}, \tag{3.5-2}$$

if this is less than

$$\frac{|y_i|}{|F^{(0)}(y_i)|};$$

otherwise,

$$t_{i+1} - t_i = \left(\frac{10^{-d}|F^{(0)}(y_i)|}{|F^{(K)}(y_i)/K!|}\right)^{1/K}.$$

In the expressions given for $t_{i+1} - t_i$ by (3.5-2), the quantity $|y_i|$ is supposed to be the maximum absolute value of any component of the approximate solution vector y at t_i and $F^{(K)}(y_i)$ is the Kth derivative with respect to t of the right hand side of the autonomous system of differential equations: in vector form, $\mathrm{d}y/\mathrm{d}t = F(y)$.

Numerical experiments with a number of systems of differential equations were run on the computer to *compare* actual running time using the choice of t_i, K_i indicated with the time required for other choices of t_i, K_i. In all cases tried the choices given here gave results of comparable known accuracy (i.e. comparable machine computed error bounds) in computing times which were close to the minimum times found.

A suggestion sometimes heard is that in the presence of a nearby singularity the order K_i should be varied from step to step for more efficient computation. However, I do not know of any very definite results to that effect.

Over a fairly short range of values of the independent variable quite sharp intervals (containing exact solution values) are obtained by the computer based on repeated Taylor series expansions with interval computation of the remainder terms. Interval widths of a few units in the last place carried in the fixed precision arithmetic used have been obtained in this way for initial value problems involving a variety of systems of differential equations including the restricted – three body equations (3.5-1), [36]. This required about 5 seconds per step for the restricted three body problem on the IBM 7094 computer *but no preparatory analysis by hand.*

This could be compared with about ·1 seconds per step using truncated Taylor series expansions of the same order at each step but without all the error bounding interval computations.

There is a price to be paid for automatic guaranteed error bounding by the computer. It may still be infinitesimal compared to the cost of guaranteed error bounding by the analyst, (see section 1 of this paper).

For *long range* numerical solutions with automatic error bounding by the computer using this approach there is a source of excessive growth in the widths of the bounding intervals. It is that the family of solution points $\{y(t) \mid y(t_i) \in R(t_i)\}$ emanating from an n-dimensional (for an nth order system) rectangle $R(t_i)$ fill out a region $S(t)$ which at some t_{i+1} must be bounded by another n-dimensional rectangle $R(t_{i+1})$ *with sides parallel to the coordinate axes.* If the region $S(t)$ is a rigidly rotating rectangle, for example, then $R(t_{i+1})$ can grow to arbitrarily large width, [36].

Methods for reducing the effect of this source of growth of error bounds have been studied, [35], [36]. These include a procedure based on local coordinate transformations $y = y^* + C^*z$ using an approximation $C^*(t)$ to the "connection matrix" $C(t)$ of the vector field (given by $y' = f(y)$) along an approximate solution $y^*(t)$, defined by

$$\text{(3.5-3)} \qquad \frac{\mathrm{d}C(t)}{\mathrm{d}t} = J(t)\,C(t)\,, \quad C(0) = I$$

where I is the identity matrix and $J(t)$ is the Jacobian matrix $(\partial f_i/\partial y_j)\mid_{y^*(t)}$.

A computer program incorporating this procedure did not, at first, have the expected beneficial effect. The bounds grew faster, in fact.

The cause of the trouble was the loss of cancellation of error in interval subtraction because of the loss of the identity of variables after substitution of interval values. The cancellations must be done symbolically, [36]. Computer programs and programming languages providing this capability in convenient form may be available soon. Meanwhile, the formulas required for the application of the transformation technique can be *derived* by hand for each specific problem and programmed for use by the computer in connection with an interval solution of that specific problem.

Results obtained in this way were reported [36] for the initial value problem

$$\frac{dy_1}{dx} = y_2\,, \quad \frac{dy_2}{dx} = -y_1\,, \qquad (3.5\text{-}4)$$

$$y_1(0) = 0\,, \quad y_2(0) = 1{\cdot}0\,.$$

The computations were performed on the CDC 1604 computer using the interval version of repeated Taylor series expansion with $K = 12$ (twelve terms in the series plus remainder in interval form) and using local coordinate transformations based on approximate solution of (3.5-3). We refer to [36] for details. We quote part of the results of the computation.

SOLUTION AT $X = 3{\cdot}0$

$Y1 =\ \ {\cdot}14112000805$ ERROR BOUND $= 5{\cdot}6\,.\,10^{-10}$

$Y2 = -{\cdot}98999249665$ ERROR BOUND $= 6{\cdot}2\,.\,10^{-10}$

SOLUTION AT $X = 6{\cdot}0$

$Y1 = -{\cdot}27941549819$ ERROR BOUND $= 2{\cdot}3\,.\,10^{-9}$

$Y2 =\ \ {\cdot}96017028692$ ERROR BOUND $= 2{\cdot}4\,.\,10^{-9}$

SOLUTION AT $X = 9{\cdot}0$

$Y1 =\ \ {\cdot}41211848524$ ERROR BOUND $= 9{\cdot}3\,.\,10^{-9}$

$Y2 = -{\cdot}91113026212$ ERROR BOUND $= 9{\cdot}4\,.\,10^{-9}$

The error bounds are still growing, but not as fast as without the use of the transformation technique. Using the transformation technique the bounds in this example grew (from $x = 3{\cdot}0$ to $x = 9{\cdot}0$) by the factor

$$\frac{9{\cdot}4\,.\,10^{-9}}{6{\cdot}2\,.\,10^{-10}} \approx 15$$

whereas *without* the transformation technique the factor of increase was 420.

In principle, the transformation technique can reduce the factor of increase to $1 + \varepsilon$ for any $\varepsilon > 0$ in this example. The bounds can also be decreased *without* the transformation technique by going to higher precision arithmetic, (and correspondingly more terms in the Taylor series).

Which of these approaches is more efficient has not yet been determined.

An alternative approach based on quadratic forms describing ellipsoidal bounding regions has been suggested by W. Kahan of the University of Toronto, (see p. 335 of the proceedings referred to in [13]).

We have chosen another example from [6] to further illustrate the sharpness of bounds obtainable using the transformation technique.

On p. 101 of [6] is given a table of results obtained using the Runge-Kutta method to solve approximately the problem

$$y' = -y \quad \text{for} \quad y(0) = 0{\cdot}9\,. \tag{3.5-5}$$

The Runge-Kutta method was modified for the computations to include addition and subtraction of bounds on the local error (taking round-off error into account) at each step and the results are reported in the form of a "lower" and an "upper" solution.

We repeat the example, for comparison, using our interval-transformation method. In order to obtain a reasonably fair comparison with the results of [6], we will reduce the number of terms to be carried in our Taylor series expansions to five so that our local error will be $O(h^5)$ like the Runge-Kutta method. Also we carry out our successive Taylor expansions at the constant step $h = 0{\cdot}1$ used in [6].

For the approximate solution $y^*(t)$ in (3.5-3) we use the discrete approximation $y_p^* \approx y({\cdot}1p)$ given by $y_0^* = {\cdot}9$

$$y_p^* = \left(\sum_{j=0}^{4} \frac{(-1)^j}{j!} ({\cdot}1)^j\right) y_{p-1}^*\,, \quad p = 1, 2, \ldots \tag{3.5-6}$$

For the approximate connection "matrix" $C^*(t)$, (which in this simple example is a scalar function) we use $C_0^* = 1{\cdot}0$. From (3.5-5), $J(t) \equiv -1{\cdot}0$; we put $C_p^* = (1 + hJ)\, C_{p-1}^*$, or

$$C_p^* = {\cdot}9 C_{p-1}^*\,, \quad p = 1, 2, \ldots \tag{3.5-7}$$

at "mesh points" $t_p = {\cdot}1p$ and define $C^*(t)$ for intermediate points by linear interpolation.

From $y' = -y$ we obtain, for z defined implicitly by $y = y^* + C^*z$, the derived differential equation in $t_p \leqq t < t_p + {\cdot}1$

$$\begin{aligned} z' &= (C^*)^{-1}(-y^* - y^{*\prime} - C^*z - C^{*\prime}z) \\ &= \frac{1}{(1 - (t - t_p))\, C_p^*}\left(\frac{-(t - t_p)^4}{4!}\, y_p^* + (t - t_p)\, C_p^* z\right). \end{aligned} \tag{3.5-8}$$

Notice the cancellations we have performed in (3.5-8).

If $z(t_p) \in z_p$, then for $t_p \leqq t < t_p + {\cdot}1$ we have $z(t) \in z_p + (t - t_p)\, z'([t_p, t_p + {\cdot}1])$ by the mean value theorem, so

$$z(t) \in z_p + B_p\,, \tag{3.5-9}$$

where B_p is an interval such that (using (3.5-8))

$$z_p + \frac{[0, {\cdot}1]}{(1 - [0, {\cdot}1])\, C_p^*}\left(-\frac{[0, {\cdot}1]^4}{24}\, y_p^* + [0, {\cdot}1]\, C_p^*(z_p + B_p)\right) \subset z_p + B_p\,. \tag{3.5-10}$$

By continuity, we also have $z(t_p + \cdot 1) \in z_p + B_p$. Thus if we choose z_0 such that $y_0 \in y_0^* + C_0^* z_0$ and put $z_{p+1} = z_p + B_p$ then it will follow that for $t_p \leqq t \leqq t_{p+1}$

$$y(t) \in y^*(t) + C^*(t)(z_p + B_p)\,. \tag{3.5-11}$$

We must now choose B_p satisfying (3.5-10). We try B_p in the form

$$B_p = \alpha z_p + R_p\,, \tag{3.5-12}$$

and try to satisfy

$$-\frac{[0, 10^{-4}]}{216}\frac{y_p^*}{C_p^*} + \left[0, \frac{\cdot 1}{9}\right](z_p + \alpha z_p + R_p) \subset \alpha z_p + R_p\,. \tag{3.5-13}$$

If $0 \in z_0$ and $0 \in R_p$, then $0 \in z_p + R_p = z_{p+1}$ by induction for all p and

$$\left[0, \frac{\cdot 1}{9}\right](z_p + \alpha z_p) \subset \alpha z_p \quad \text{provided} \quad \alpha \geqq \frac{1}{89}\,. \tag{3.5-14}$$

Put $R_p = [-r_p, r_p]$ and choose r_p so that

$$\frac{-[0, 10^{-4}]}{216}\frac{y_p^*}{C_p^*} \subset \left(1 - \frac{\cdot 1}{9}\right)[-r_p, r_p] \tag{3.5-15}$$

then (3.5-13) and (3.5-10) will be satisfied. We can satisfy (3.5-15) by choosing

$$r_p = 4{\cdot}7\,.\,10^{-7}\left|\frac{y_p^*}{C_p^*}\right|. \tag{3.5-16}$$

This gives (along with (3.5-14))

$$B_p = \frac{1}{89}z_p + [-1, 1]\left(4{\cdot}7\,.\,10^{-7}\left|\frac{y_p^*}{C_p^*}\right|\right). \tag{3.5-17}$$

From (3.5-17) and $z_{p+1} = z_p + B_p$ and (3.5-6) and (3.5-7) and (3.5-11) we have, finally that

$$C_{p+1}^* z_{p+1} \in (\cdot 9102)^{p+1}(1{\cdot}06\,.\,10^{-4}[-1, 1] + z_0) \tag{3.5-18}$$

and the exact solution to (3.5-5) at t_{p+1} satisfies

$$y(t_{p+1}) \in y_{p+1}^* + C_p^* z_{p+1} \tag{3.5-19}$$

with $C_{p+1}^* z_{p+1}$ bounded by (3.5-18) and $y_{p+1}^* = (\cdot 9048375)^{p+1}(\cdot 9)$ from (3.5-6).

For $p = 100$, this gives $t_{100} = 10{\cdot}0$, $y(10{\cdot}0) \in [\cdot 000040844, \cdot 000040876]$. The exact solution is $y(10{\cdot}0) = \cdot 000040860\ldots$

By comparison the upper and lower solutions given in [6] determine the following bounds at $t_{100} = 10{\cdot}0$:

$$y(10{\cdot}0) = [{\cdot}000040314, {\cdot}000041403].$$

The relative error in our y^*_{p+1} can be bounded using (3.5-18) and (3.5-19) by

$$\left|\frac{y(t_{p+1}) - y^*_{p+1}}{y^*_{p+1}}\right| < (1{\cdot}006)^{p+1}\,(2{\cdot}12)\,.\,10^{-4}\,. \tag{3.5-20}$$

The right hand side of (3.5-20) will be less than $1{\cdot}0$ for $p < 1424$; i.e. the bounds will be sharper than one in the leading non-zero digit for $t < 142{\cdot}4$. This point is reached by the method of [6] by $t \doteq 19{\cdot}0$.

The *desirability* of programming a *general* procedure enabling the *computer* to carry out all the derivations and computations involved in applying the transformation technique to specific examples should be obvious from the details of the very simple example just discussed.

I am convinced it can be done.

In addition to providing a means for the automatic determination of sharp rigorous error bounds in the initial value problem, such a computer program would be a valuable aid in the computation of "reachable sets" in control theory problems.

4. FUNCTIONAL ANALYSIS FOR COMPUTERS

It is beyond the scope of this paper to attempt a survey of the unifying concepts and very general techniques of functional analysis which are enjoying such vigorous application to computational problems. Important examples of work in this area can be found in [1], [5], [7], [9], [10], [11], [13], [20], [30], [38].

In this final section we will merely discuss a few of the computational and error bounding techniques of functional analysis which can, in principle, be carried out by the computer with the help of interval computations. KRÜCKEBERG [32] has reported some work on partial differential equations using interval computation.

In this section, we will discuss an application of the contraction mapping principle to the initial value problem and an application of Newton's method to the two point boundary value problem.

We will use the following form of the contraction mapping principle, given by RALL [38]:

If F is an operator in a Banach space X which is a contraction mapping of $\overline{U}(x^{(0)}, r)$ for

$$r \geqq \frac{1}{1-\theta}\,\|x^{(0)} - F(x^{(0)})\| = r_0 \tag{4-1}$$

where $0 \leqq \theta < 1$ and

$$\|F(\bar{x}) - F(y)\| \leqq \theta\|\bar{x} - y\| \tag{4-2}$$

for all

$$\bar{x}, y \in \bar{U}(x^{(0)}, r) = \{\bar{x} \mid \|\bar{x} - x^{(0)}\| \leqq r\}$$

then:

1. F has a fixed point x^* in $\bar{U}(x^{(0)}, r_0)$.
2. x^* is the unique fixed point of F in $\bar{U}(x^{(0)}, r)$.
3. The sequence of successive approximations defined by

$$x^{(m+1)} = F(x^{(m)}), \quad m = 0, 1, 2, \ldots$$

converges to x^* with

(4-3) $$\|x^{(m)} - x^*\| \leqq \theta^m r_0 .$$

An interesting application of this principle is the following.

Suppose we are given an initial value problem

(4-4) $$x' = f(t, x), \quad x(t_0) = x_0$$

(with f continuously differentiable in a suitable region) and an approximate numerical solution at the points $t_0 < t_1 < t_2 < \ldots < t_n$, say $x_i \approx x(t_i)$; $i = 1, 2, \ldots, n$ where $x(t)$ is the exact (unknown) solution to (4-4). The contraction mapping principle can be used to compute bounds on the errors $x_i - x(t_i)$ *without knowing how the numbers x_i were computed.*

We choose for X, the Banach space of continuous functions on $[t_0, t_n]$ with the norm

$$\|x\| = \max_{t \in [t_0, t_n]} |x(t)| \quad \text{for} \quad x \in X .$$

We write the initial value problem (4-4) as an integral equation

(4-5) $$x(t) = x_0 + \int_{t_0}^{t} f(s, x(s))\, \mathrm{d}s .$$

The equation (4-5) has the form

(4-6) $$x = F(x)$$

where the operator F is defined for $y \in X$ by

(4-7) $$F(y)(t) = x_0 + \int_{t_0}^{t} f(s, y(s))\, \mathrm{d}s .$$

Suppose we interpolate the approximate numerical solution by some *continuous* function $x^{(0)}$ (for example a polynomial) such that $x^{(0)}(t_i) = x_i$, $i = 0, 1, 2, \ldots, n$.

If the conditions (4-1), (4-2) are satisfied, then we will have, from (4-3), $\|x^{(0)} - x\| \leqq$ $\leqq r_0$; in particular, we have a uniform bound on the errors of the approximate solution

$$|x_i - x(t_i)| \leqq \|x^{(0)} - x\| \leqq r_0 . \tag{4-8}$$

We illustrate the details of the application with an example.

Consider the initial value problem

$$x' = x^2 , \quad x(0) = 1{\cdot}0 \tag{4-9}$$

and the single approximate solution value $x_1 = 1{\cdot}143 \approx x({\cdot}125)$. The problem can be rewritten as the integral equation

$$x(t) = 1{\cdot}0 + \int_0^t x^2(s)\, \mathrm{d}s . \tag{4-10}$$

We define F for the space X of continuous functions on $[0, {\cdot}125]$ by

$$F(y)(t) = 1{\cdot}0 + \int_0^t y^2(s)\, \mathrm{d}s . \tag{4-11}$$

Then

$$\begin{aligned} \|F(\bar{x}) - F(y)\| &= \max_{t\in[0,{\cdot}125]} \left| \int_0^t (\bar{x}^2(s) - y^2(s))\, \mathrm{d}s \right| \\ &\leqq {\cdot}125\|\bar{x} + y\|\ \|\bar{x} - y\| . \end{aligned} \tag{4-12}$$

For $x^{(0)}$, we choose an interpolating quadratic polynomial $x^{(0)}(t) = 1 + x'(0)\, t + at^2$. We find from (4-9) that $x'(0) = 1$ and determine the coefficient, a, from $x^{(0)}({\cdot}125) = 1{\cdot}143$ that is, $1 + {\cdot}125 + ({\cdot}125)^2\, a = 1{\cdot}143$ or $a = 1{\cdot}152$ and so

$$x^{(0)}(t) = 1 + t + 1{\cdot}152t^2 . \tag{4-13}$$

Now, for $\bar{x}$, $y \in U(x^{(0)}, r)$ and since

$$\max_{t\in[0,{\cdot}125]} |x^{(0)}(t)| = 1{\cdot}143 , \tag{4-14}$$

we have

$$\|\bar{x} + y\| \leqq 2(\|x^{(0)}\| + r) = 2(1{\cdot}143 + r) . \tag{4-15}$$

Thus for θ in (4-2) we can put

$$\theta = {\cdot}125(2{\cdot}286 + 2r) = {\cdot}28575 + {\cdot}25r . \tag{4-16}$$

We check that $0 \leqq \theta < 1$ provided that $r < 2{\cdot}86\ldots$

We next compute an upper bound on r_0 from (4-1). For this, we need an upper bound on $\|x^{(0)} - F(x^{(0)})\|$. From (4-13) and (4-11) we have

(4-17) $$x^{(0)} - F(x^{(0)}) = \int_0^t ((x^{(0)}(s) - 1)' - (x^{(0)}(s))^2)\,ds$$
$$= \int_0^t (\cdot 304s - 3\cdot 304s^2 - 2\cdot 304s^3 - 1\cdot 327104s^4)\,ds\,.$$

Put

(4-18) $$p(t) = \cdot 152t^2 - (1\cdot 101\ldots)\,t^3 - \cdot 578t^4 - (\cdot 265\ldots)\,t^5\,.$$

Then

(4-19) $$\|x^{(0)} - F(x^{(0)})\| \leqq \max_{t\in[0,\cdot 125]} |p(t)|\,.$$

We have for $t \in [0, \cdot 125]$

(4-20) $$p(t) \in \cdot 152[0, \cdot 152]^2 - [1\cdot 10, 1\cdot 11]\,[0, \cdot 125]^3 - \cdot 578[0, \cdot 125]^4$$
$$- [\cdot 26, \cdot 27]\,[0, \cdot 125]^5\,.$$

Therefore

$$\max_{t\in[0,\cdot 125]} |p(t)| < \cdot 00238$$

and

(4-21) $$\|x^{(0)} - F(x^{(0)})\| \leqq \cdot 00238\,.$$

The smallest r_0 we are entitled to use for (4-8) is, from (4-1) and (4-16) and (4-21), the smallest positive root of

(4-22) $$r_0 = \left(\frac{1}{1 - \cdot 28575 - \cdot 25r_0}\right)(\cdot 00238)\,.$$

Call

$$g(r_0) = \frac{\cdot 0033\ldots}{1 - (\cdot 35\ldots)\,r_0}$$

then

$$g([\cdot 003, \cdot 004]) = \frac{\cdot 0033\ldots}{[\cdot 9986\ldots, \cdot 9989\ldots]} \subset [\cdot 0032, \cdot 0034]$$

so

(4-23) $$r_0 < \cdot 0034$$

and we have finally arrived at the result that

(4-24) $$|x(\cdot 125) - 1\cdot 143| < \cdot 0034\,.$$

Actually the exact value of $x(\cdot 125)$ for (4-9) is $x(\cdot 125) = \frac{8}{7} = 1{\cdot}14285\ldots$ and so the actual error is $x(\cdot 125) - 1{\cdot}143 = -{\cdot}00015\ldots$

We will comment upon the possibility of programming the various steps in applying the technique from (4-12) through (4-24).

The factorization we did in (4-12) could be carried out, at least for rational functions f in (4-4) by "polynomial manipulation programs".

The computation of coefficients for an interpolating function (perhaps a polynomial) leading to (4-13) can certainly be programmed.

The computation of an upper bound for θ in (4-16) as a function of r and testing for $\theta < 1$ could be programmed for a given form (say polynomial) of the majorizing function of r (at least for rational f) using interval techniques, for example.

The computation of (4-17), for sharp results, should use some algebraic manipulation for symbolic cancellations and the upper bound in (4-19) can be programmed using interval methods, for example, such as we in fact used in (4-20) to get (4-21).

Again we used interval methods, which could be programmed, to get (4-23) from (4-22).

Some of the steps could be refined to give sharper bounds. For example, the bound (4-21) can be sharpened (among other ways, [36]) by subsividing the interval $[0, \cdot 125]$ as $[0, \cdot 125] = \bigcup_{j=1}^{N} [a_j, b_j]$ and computing, in place of (4-20),

$$p(t) \in \bigcup_{j=1}^{N} p([a_j, b_j]) .$$

We turn now to an application of Newton's method using a generalization to Banach spaces by KANTOROVIČ, [30].

We will outline briefly some recent work of Mr. T. TALBOT, a doctoral candidate at the University of Wisconsin. A more complete description of his work will appear elsewhere.

The form of the Kantorovič theorem [30] used by Talbot is:

Let P be an operator in a Banach space B. Let $y_0 \in B$ and let $(P'(y_0))^{-1}$ exist. If

$$y_1 = y_0 - (P'(y_0))^{-1} P(y_0) \tag{4-25}$$

and

$$\|(P'(y_0))^{-1}\| \leqq \beta_0 \tag{4-26}$$

and

$$\|y_1 - y_0\| \leqq \eta_0 \tag{4-27}$$

and

$$\|P''(y)\| \leqq K \quad \text{for} \quad \|y - y_0\| \leqq 2\eta_0 \tag{4-28}$$

and

$$(4\text{-}29) \qquad \beta_0\eta_0 K \leqq \tfrac{1}{2},$$

then there exists $y^* \in B$ such that $P(y^*) = 0$ and

$$(4\text{-}30) \qquad \|y_1 - y^*\| \leqq 2\beta_0 K\eta_0^2 .$$

Talbot considers the application of this theorem to the computer solution, *with automatic guaranteed error bounds,* of the two point boundary value problem

$$(4\text{-}31) \qquad y''(x) = f(x, y)\,; \quad y(a) = y(b) = 0 .$$

(Non-homogeneous boundary conditions can, of course, be reduced to (4-31) by addition of a linear function of x to y.)

Talbot has written a computer program for the CDC 1604 computer which, given a problem of the form (4-31), tries to construct the numbers β_0, η_0, and K for (4-26), (4-27), and (4-28) using interval analysis [36]. The program first tries to generate a y_0 by iterating (via Newton's method) a discrete solution several times starting with the constant function 0. This is then interpolated by cubic splines. Then an interval function Y_1 containing y_1 is found, if possible, such that y_1 satisfies (4-25). The numbers β_0, η_0, and K are obtained using interval techniques for obtaining upper bounds on the norms. If these numbers satisfy (4-29) then the bound (4-30) is computed. All the derivations, (e.g. for $P'(y_0)$), and computations are carried out automatically by the computer. If some part of the process cannot be carried out for a given problem, (which may not satisfy part of the hypotheses), then the computation stops and a "discouraging" message is printed out.

The program was tried on the following example

$$(4\text{-}32) \qquad y''(x) = 2(y(x) - {\cdot}8x + 1{\cdot}8)^3, \quad y(1) = y(1{\cdot}25) = 0$$

(with known solution $y(x) = 1/x + {\cdot}8x - 1{\cdot}8$).

Three Newton iterations were required by the program to obtain y_0 as tabulated here. The known solution is listed for comparison.

x	$y_0(x)$	$y(x)$	$y_0(x) - y(x)$
1·0	0·0	0·0	0·0
1·05	−0·007556487	−0·007619048	0·000062560
1·10	−0·010861949	−0·010909091	0·000047142
1·15	−0·010403285	−0·010434783	0·000031498
1·20	−0·006650829	−0·006666667	0·000015838
1·25	0·0	0·0	0·0

The program then constructed an interval function containing $y_1(x)$ and then constructed the following bounds to be used with (4-30)

$$\beta_0 = \cdot 068\,, \quad \eta_0 = \cdot 000076\,, \quad K = 12{\cdot}0\,.$$

The final output of the program was given as the first three columns in the table below. The actual errors in $y_1(x)$ are listed for comparison.

x	$y_1(x)$	Guaranteed bound for $\lvert y_1(x) - y(x)\rvert$	Actual error $y_1(x) - y(x)$
1·00	0·00000000000	0·00000000000	0·00000000000
1·05	—·00761904746	0·00000000956	0·00000000016
1·10	—·01090909072	0·00000000959	0·00000000019
1·15	—·01043478245	0·00000000963	0·00000000016
1·20	—·00666666657	0·00000000968	0·00000000009
1·25	0·00000000000	0·00000000000	0·00000000000

References

[1] *Anselone, P. M.,* "Convergence and error bounds for approximate solutions of integral and operator equations," Error in Digital Computation, Vol. *II,* Ed. by L. B. Rall, Wiley, New York, 1965, pp. 219—230.

[2] *Apostolatos, N.* and *Kulisch, U.,* "Grundlagen einer Maschinenintervallarithmetik", Technische Hochschule Karlsruhe, Karlsruhe, September, 1966.

[3] *Apostolatos, N.* and *Kulisch, U.,* "Approximation der erweiterten Intervallarithmetik durch die einfache Maschinenintervallarithmetik", THK, Karlsruhe, November, 1966.

[4] *Apostolatos, N., Kulisch, U.,* and *Nickel, K.,* "Ein Einschliessungsverfahren für Nullstellen", THK, Karlsruhe, December, 1966.

[5] *Babuška, I.* and *Sobolev, S. L.,* "Оптимизация численных методов", Aplikace matematiky, svazek *10* (1965), pp. 96—129.

[6] *Babuška, I., Práger, M.,* and *Vitásek, E.,* Numerical Processes in Differential Equations, Interscience, London, 1966.

[7] *Banach, S.,* Opérations Linéaires, Monografje Matematyczne, Warsaw, 1932.

[8] *Boche, R.,* "Complex interval arithmetic with some applications", Lockheed Missiles and Space Company report # 4—22—61—1, 1966.

[9] *Brown, R. W.,* "Upper and lower bounds for solutions of integral equations", Error in Digital Computation, Vol. *II,* Ed. by L. B. Rall, Wiley, New York, 1965, pp. 231—252.

[10] *Collatz, L.,* Numerical Treatment of Differential Equations (English Ed.), Berlin, 1959.

[11] *Collatz, L.,* "Applications of functional analysis to error estimation", Error in Digital Computation, Vol. *II,* Ed. by L. B. Rall, Wiley, New York, 1965, pp. 253—269.

[12] *Collins, G.,* "PM, a system for polynomial manipulation", Comm. A.C.M., Vol. *9,* No. 8, August, 1966, pp. 578—589. (See also *other* papers in this same issue.)

[13] *Dahlquist, G.,* "On rigorous error bounds in the numerical solutions of ordinary differential equations", Numerical Solutions of Nonlinear Differential Equations, Ed. by D. Greenspan, Wiley, 1966, pp. 89—96.

[14] *Dargel, R. H., Loscalzo, F. R.,* and *Witt, T. H.,* "Automatic Error Bounds on Real Zeros of Rational Functions", Communications of the ACM, Vol. *9,* No. 11, 1966, pp. 806—809.

[15] *Fehlberg, E.,* "Eine Methode zur Fehlerverkleinerung beim Runge-Kutta-Verfahren", Z. Angew. Math. Mech., *38* (1958), pp. 421—426.

[16] *Fehlberg, E.,* "Runge-Kutta type formulas of high order accuracy and their application to the numerical integration of the restricted problem of three bodies." Proc. Int. Symp. on Analogue and Digital Techniques Applied to Aeronautics, Liege, Belgium, September, 1963.

[17] *Fehlberg, E.,* "Zur numerische Integration von Differential-gleichungen durch Potenzreihen-Ansätze...", Z. Angew. Math. Mech., *44* (1964), pp. 83—88.

[18] *Fehlberg, E.,* "New High-Order Runge-Kutta Formulas with Step-size Control for Systems of First-and Second-Order Differential Equations", Z. Angew. Math. Mech., *44* (1964), Sonderheft, pp. T17—T29.

[19] *Fehlberg, E.,* "New High-order Runge Kutta Formulas with an Arbitrarily Small Truncation Error," (personal communication), 1965.

[20] *Fréchet M.,* Les Espaces Abstraits, Gauthier-Villars, Paris, 1928.

[21] *Gallaher, L. J.* and *Perlin, I. E.,* "A comparison of several methods of numerical integration of nonlinear differential equations", RECC, Georgia Institute of Technology, Atlanta, Georgia, 1966.

[22] *Gibb, A.,* "ALGOL 60 Procedures for Range Arithmetic," Tech. Rep. No. 10, Appl. Math. Stat. Lab., Stanford University, 1961.

[23] *Gingerich, O.,* "The Computer Versus Kepler", American Scientist, *52,* 1964, pp. 218—226.

[24] *Goldstine, H. H.* and *Von Neumann, J.,* "Numerical inverting of matrices of high order, II", Proc. Amer. Math. Soc., Vol. *2,* 1951, pp. 199—202. Also: Bull. Amer. Math. Soc. *53* (1947) pp. 1021—1099.

[25] *Gröbner, W.,* Die Lie Reihen und Ihre Anwendungen, Deutscher Verlag der Wissenschaften, Berlin, 1960.

[26] *Hansen, E.,* "Interval arithmetic in matrix computations", J.S.I.A.M., series B, Numerical Analysis, Part 1, *2* (1965), pp. 308—320.

[27] *Hansen, E.* and *Smith, R.,* "Interval arithmetic in matrix computations, Part 2", J.S.I.A.M., series B, Numerical Analysis, *4:* 1 (1967), pp. 1—9.

[28] *Hansen, E.,* "On solving systems of equations using interval arithmetic", (to appear).

[29] *Henrici, P.,* Discrete Variable Methods in Ordinary Differential Equations, Wiley, New York, 1962.

[30] *Kantorovič, L. V.,* "On Newton's method for functional equations", Dokl. Akad. Nauk SSSR (N.S.), *59* (1948), pp. 1237—1240.

[31] *Krückeberg, F.,* "Zur numerische Intervallrechnung", Rheinisch-Westfälisches Institute für Instrumentelle Mathematik, Bonn, June, 1966.

[32] *Krückeberg, F.,* "Defekterfassung bei gewöhnlichen und partiellen Differentialgleichungen, Bonn, June, 1966.

[33] *Meinguet, J.,* "La contrôle des erreurs en calcul automatique", M.B.L.E. Laboratoire des Recherches, Brussels, Rapport R 29, avril, 1965.

[34] *Moore, R. E.,* "The automatic analysis and control of error in digital computation based on the use of interval numbers", Error in Digital Computation, Vol. *I,* Ed. by L. B. Rall, Wiley, 1965, pp. 61—130.

[35] *Moore, R. E.,* "Automatic local coordinate transformations to reduce the growth of error bounds in interval computation of solutions of ordinary differential equations", Error in Digital Computation, Vol. *II,* Ed. by L. B. Rall, Wiley, New York, 1965, pp. 103—140.

[36] *Moore, R. E.,* Interval Analysis, Prentice-Hall, 1966.

[37] *von Neumann, J.* and *Goldstine, H.,* "Numerical Inversion of Matrices of High Order", Bulletin of Amer. Math. Soc. *53* (1947).

[38] *Rall, L. B.,* Computational Methods for Nonlinear Operator Equations, (to appear).

[39] *Reiter, A.,* "Compiler of differentiable expressions", MRC Computer Program # 1., Mathematics Research Center, University of Wisconsin.

[40] *Reiter, A.,* "Interval arithmetic package", MRC Computer Program # 2, Math. Res. Center, University of Wisconsin.

[41] *Turing A. M.,* "Rounding-off errors in matrix processes", Quart. J. Mech. Appl. Math., Vol. *1*, 1948, pp. 287—308.

Ramon E. Moore, Computer Sciences Department, University of Wisconsin, 1210, W. Dayton, Madison, Wisconsin, U. S. A.