Eldon Hansen

TECHNICAL REPORT

AUTOMATIC ERROR ANALYSIS IN DIGITAL COMPUTATION

LMSD-48421 28 JANUARY 1959

Lockheed

MISSILES and SPACE DIVISION

LOCKHEED AIRCRAFT CORPORATION · SUNNYVALE, CALIF.

TECHNICAL REPORT

AUTOMATIC ERROR ANALYSIS IN DIGITAL COMPUTATION

BY RAMON E. MOORE

LMSD-48421 28 JANUARY 1959

WORK CARRIED OUT UNDER
LOCKHEED GENERAL RESEARCH PROGRAM

Lockheed

MISSILES and SPACE DIVISION

LOCKHEED AIRCRAFT CORPORATION • SUNNYVALE, CALIF.

FOREWORD

This technical report is presented on research conducted during 1958 on Numerical Analysis for Digital Computers. Work on this study was carried out under the Lockheed General Research Program.

SUMMARY

This technical report introduces a systematic approach to the general subject of digital computation. The objective of the study is to mechanize a complete error analysis for any digital computation. A method for such mechanization is described, by which the digital computer is to deliver error bounds along with every approximation.

CONTENTS

Section 1
INTRODUCTION

We present a formal description of a stored-program, digital computer. The automatic or sequential operation of such a machine forms a basis for the present study. We investigate finite or terminating computations by such a computer and their relation to both terminating and non-terminating computations with real numbers. In particular, we are concerned with automatically obtaining error bounds along with approximations in digital computation.

We exhibit an interval arithmetic which is the basis for an automatic analysis of total error in any direct digital computation.

Finally, we consider, briefly, a particular class of indirect computations: iterative computations.

Previous work on automatic error analysis is discussed in S. Gorn, "The automatic analysis and control of computing errors," J. Soc. Indust. Appl. Math. Vol 2, No. 2, June 1954, and in S. Gorn and R. Moore, "Automatic error control — the initial value problem in ordinary differential equations," Aberdeen Proving Ground, BRL Report No. 893, March 1953.

We may represent the integers $0, 1, \dots, k-1$ by the words $00\dots0$, $00\dots1$, ..., $11\dots1$ respectively, hence we will consider the address cell to contain at any moment the address of some cell in $M_{s,m}$ by means of the indicated representation.

By a <u>state</u> of the computer we mean the ordered set of words contained in its cells: $\{(C),\ (0),\ (1),\ \dots,\ (m-1)\}$.

The <u>computer</u> consists of the cell C , the memory $M_{s,m}$, and a transformation called machine language, or ML , which we shall sketch here and return to in Section 5.

Let Σ be the set of states defined above, we will define a transformation ML which is a function whose domain is Σ and whose range is contained in Σ . If σ is a state, then $ML(\sigma)$ is called the successor of σ and we use the notation $\sigma' = ML(\sigma)$. For any state σ , the ordered pair of states (σ, σ') is called an <u>operational cycle.</u> The act of changing a state σ into its successor is assumed to involve an interval of time during which the following takes place:

- The address cell C is examined in state σ ; the word (C) is found in C ; (C) represents the address of some cell in $M_{s,m}$.
- The cell (C) in $M_{s,m}$ is examined in state σ ; the word $((C))$ is found in (C) ; $((C))$ is some word in W_s which is interpreted in the context of machine language as representing an instruction to carry out a particular operation on the words in the cells of the computer, i.e. to change the state. The "interpretation" involves an examination of the word $((C))$ in state σ in order to discover how the successor state σ' is to be computed from the state σ .
- The interpreted instruction is "executed," changing σ to σ' .

A <u>computation</u> by the computer is a sequence of states beginning with some initial state σ_0 called a <u>program</u> and proceeding in successive operational cycles to the states σ_0' $(\sigma_0')'$, ...

A state σ is called a terminal state if $\sigma' = \sigma$. We assume for at least one word w in W_s that if $((C)) = w$ in state σ, then σ is a terminal state.

Omits cases of cycles of order > 1 ([illegible]

A computation by the computer is completely determined by its initial state or program. Let $\sigma_o{}^*$ stand for the ordered set $\{ \sigma_o, \sigma_o', (\sigma_o')', \dots \}$ of states computed from σ_o by successive applications of the mapping ML. We say that $\sigma_o{}^*$ is <u>finite</u> if it contains a terminal state. A program σ_o produces a <u>finite computation</u> if $\sigma_o{}^*$ is finite. If we call the states $\sigma_o' = \sigma_1$, $\sigma_1' = \sigma_2$, etc., then the <u>length</u> of a finite computation is the integer i where σ_i is the first terminal state in the computation.

The number of distinct words in W_s is 2^s; the number of distinct words in W_k is m; the number of distinct states is therefore $m2^{ms}$. The number of distinct terminal states is at least $2^{(m-1)s}$. Any computation by the computer will therefore be either a finite computation of length not exceeding $m2^{ms}$, or it will be a <u>cyclic</u> computation. By a cyclic computation, we mean a sequence of states $\{ \sigma_o, \sigma_1, \dots \}$ containing a state σ_p such that $\sigma_{p+1} = \sigma_i$ for some $i < p$.

Since any state may be taken as a program for a computation, there are $m2^{ms}$ distinct programs. At least $2^{(m-1)s}$ produce finite computations since we may take a terminal state for a program.

If we have, for example, a computer with a 4000-cell memory whose cells store words of length 36 (binary digits), then there are more than $10^{43,200}$ distinct programs which produce finite computations.

Section 3
DIGITAL REPRESENTATIONS

3.1 INTRODUCTION

Before we can discuss digital computations, we first consider some means of representing numbers and other things by computer words.

Clearly there are exactly 2^s distinct words in W_s and therefore we can represent at most 2^s distinct things by any relation which associates a unique thing with each word in W_s .

3.2 THE FUNCTION R_1

We can represent the integers $0, 1, 2, \ldots, 2^{s-1}$ by the words $00\ldots0, 00\ldots1, \ldots, 11\ldots1$ respectively. We define the function R_1 with domain W_s such that if $w = b_0 b_1 \ldots b_{s-1}$, then $R_1(w) = \sum_{j=0}^{s-1} b_{s-1-j} \, 2^j$.

We say that the positive integer $\sum_{j=0}^{s-1} b_{s-1-j} \, 2^j$ is <u>represented in</u> $\underline{R_1}$ by the word $b_0 b_1, \ldots b_{s-1}$.

3.3 THE FUNCTION R_2

We define the function R_2 with domain W_s by the formula

$$R_2(b_0 b_1 \ldots b_{s-1}) = (-1)^{b_0} \sum_{j=1}^{s-1} b_{s-j} \, 2^{j-1}$$

In R_2 we have a representation of the set of integers n such that $-2^{s-1} < n < 2^{s-1}$ by W_s. The correspondence is one-one except for the double representation of the integer 0 by the words 000...0 and 100...0 .

3.4 THE FUNCTION R_3

Another set we wish to represent is the set of "fixed point fractional numbers." We define R_3 on W_s such that

$$R_3(b_0 b_1 \ldots b_s) = (-1)^{b_0} \sum_{j=1}^{s-1} b_j \; 2^{-j} .$$

In this instance each signed (s-1)-place binary fraction in the interval (-1, +1) is represented by some word in W_s . Again the number 0 is represented twice. This particular representation enables us to approximate any real number in the interval (-1, 1) by some fractional number representable by a word in W_s with an absolute error of no more than $2^{-(s-1)}$. More precisely, for any real number x in the interval (-1, +1) there is a fractional number, $\bar{x}$, representable in R_3 by some w in W_s , i.e. a number of the form $\bar{x} = (-1)^{b_0} \sum_{j=1}^{s-1} b_j \left(\frac{1}{2}\right)^j$, with $w = b_0 b_1 \ldots b_{s-1}$ such that $|x - \bar{x}| \leq \left(\frac{1}{2}\right)^{s-1}$.

We can show this from the fact that any real number x in (0, 1) can be written $x = \sum_{j=1}^{\infty} C_j \left(\frac{1}{2}\right)^j$ for some choice of C_j from the set $\{0, 1\}$ for $j = 1, 2, \ldots$. Let the notation sgn x represent the number 1 , if $x \geq 0$ and -1 if $x < 0$. We may then write any real number x in (-1, +1) as $x = (\text{sgn } x) \sum_{j=1}^{\infty} C_j \left(\frac{1}{2}\right)^j$ for some $C_j \in \{0, 1\}$.

If we now choose b_0 such that $(-1)^{b_0} = \text{sgn } x$, and $b_j = C_j$, (j=1, 2, ..., s-1), then $x - \bar{x} = (\text{sgn } x) \sum_{j=s}^{\infty} C_j \left(\frac{1}{2}\right)^j$.

Since

$$\sum_{j=s}^{\infty} C_j \left(\frac{1}{2}\right)^j \leq \sum_{j=s}^{\infty} \left(\frac{1}{2}\right)^j = \left(\frac{1}{2}\right)^{s-1}$$

we have $|x - \bar{x}| \leq \left(\frac{1}{2}\right)^{s-1}$.

3.5 THE FUNCTION R_4

An interesting representation is that of "floating point" numbers. If n and c are fixed positive integers such that $n + c + 2 = s - 1$, then we define R_4 such that word $b_o b_1 \dots b_n b_{n+1} \dots b_{n+c+2}$ represents, in R_4 , the (floating point) number $x(2^y)$ where

$$x = (-1)^{b_o} \sum_{j=1}^{n} b_j \left(\frac{1}{2}\right)^j$$

$$y = (-1)^{b_{n+1}} \sum_{j=o}^{c} b_{n+2+j} \; 2^{c-j} .$$

In this way we can approximate any real number v whose magnitude $|v|$ is in the interval $\left(2^{-(2^{c+1}-1)},\ 2^{(2^{c+1}-1)}\right)$ by some floating point number representable by a word in W_s with a <u>relative error</u> of no more than $2^{-(n-1)}$.

Suppose v is a real number such that $2^{k-1} \leq |v| \leq 2^k$ for some integer $-(2^{c+1}-1) \leq k \leq 2^{c+1}-1$.

We may write

$$v = \left\{ (\text{sgn } v) \sum_{j=1}^{\infty} c_j \left(\frac{1}{2}\right)^j \right\} \cdot 2^k$$

for some $\{c_j\}$ such that $c_1 = 1$.

Suppose we choose b_o such that

$$(-1)^{b_o} = \text{sgn } v,$$

$$b_j = c_j \text{ for } j = 1, 2, \ldots, n,$$

and

$$b_j \text{ for } j = n+1, \ldots, n+c+2$$

such that

$$k = (-1)^{b_{n+1}} \sum_{j=o}^{c} b_{n+2+j} \; 2^{c-j}$$

Let

$$\hat{v} = \left\{ (-1)^{b_o} \sum_{j=1}^{n} b_j \left(\frac{1}{2}\right)^j \right\} \cdot 2^k .$$

[We call v a <u>normalized</u> floating point number since $b_1 = 1$]

We have

$$2^{-k} (v - \hat{v}) = (\text{sgn } v) \sum_{n+1}^{\infty} c_j \left(\frac{1}{2}\right)^j$$

and

$$2^{-k} \left| v - \hat{v} \right| \leq \left(\frac{1}{2}\right)^n$$

Since $b_1 = 1$, then $|\hat{v}| \geq 2^{k-1}$ and $\left| \dfrac{v - \hat{v}}{\hat{v}} \right| \leq \left(\frac{1}{2}\right)^{n-1}$

So far, we have described four ways in which we may represent numbers by words. In each of these representations a number was represented by a single word. We call these <u>single word representations</u>. We might also consider representing several numbers with a single word — this we call <u>fractional word representation</u>. Still another possibility is that of employing several words to represent a single number — this we call <u>multiple word representation</u>.

3.6 THE FUNCTION R_5

If $s = 2s'$ we define R_5 such that the word $b_o b_1, \ldots b_{s'-1}\; b_{s'} \ldots b_{s-1}$ represents in R_5 at once the pair of integers

$$\left(\sum_{j=o}^{s'-1} b_j\, 2^{s'-1-j} \;,\; \sum_{j=o}^{s'-1} b_{s'+j}\, 2^{s'-1-j} \right)$$

3.7 THE FUNCTION R_6

As an example of double word representation, we define R_6 such that the words $w = b_o b_1 \ldots b_{s-1}$ and $w' = b'_o b'_1 \ldots b'_{s-1}$ represent together in R_6 the number

$$\bar{x} = (-1)^{b_o} \left\{ \sum_{j=1}^{s-1} b_j \left(\tfrac{1}{2}\right)^j + \sum_{j=o}^{s-1} b'_j \left(\tfrac{1}{2}\right)^{s+j} \right\} .$$

This representation enables us to approximate any real number x in $(-1, 1)$ by some fractional number representable by an ordered pair of words in W_s with an absolute error of no more than $2^{-(2s-1)}$.

3.8 COMPLEX NUMBERS

We can represent complex numbers with pairs of words. We can take the pair (w, w') of words $w = b_o b_1 \ldots b_{s-1}$, $w' = b'_o b'_1 \ldots b'_{s-1}$ to represent a complex number $Z = (x, y)$ whose real and imaginary parts x and y are fractional numbers, representable by w and w', respectively, in R_3.

3.9 INTERVALS

We can represent an interval on the real line by a pair of words.

Let the pair (w, w'') represent the interval $[a, b]$ where a and b are fractional numbers represented by the words w and w' respectively, in R_3.

3.10 OTHER EXPRESSIONS

We can also, of course, represent things other than numbers by words in W_s .

Let the letters a, b, c, ..., z of the English alphabet be represented by words of length 5, say 00000 for a, 00001 for b, ..., 11001 for z .

Suppose $s = 30$, then we can represent by a word $w = b_o b_1 \ldots b_{29}$ in W_s any string of six letters taking $b_o b_1 b_2 b_3 b_4$ to represent the first letter in the string, etc. If we represent the symbols: "(", ")", "+", "-", "/", "." by the words 11010, 11011, 11100, 11101, 11110, 11111 respectively, then we may set up a digital representation for an expression such as . (a + b) / (c) + d e (f + g/ (h - i / (j))) . which is a string of the symbols mentioned above and which begins and ends with the symbol, "." . The above expression is a string of length 29 and we may represent it by an ordered set of five words $(w_1, w_2, w_3, w_4, w_5)$. We let w_1 represent . (a + b) and represent in groups of six the rest of the expression by $w_2, \ldots w_5$. We may fill out the last six bits of w_5 with another "." . The digital representation of the expression above will then be $(w_1, w_2, w_3, w_4, w_5)$ where

$$w_1 = 11111\ 11010\ 00000\ 11100\ 00001\ 11011$$

$$w_2 = 11110\ 11010\ 00010\ 11011\ 11100\ 00011$$

$$w_3 = 00100\ 11010\ 00101\ 11100\ 00110\ 11110$$

$$w_4 = 11010\ 00111\ 11101\ 01000\ 11110\ 11010$$

$$w_5 = 01001\ 11011\ 11011\ 11011\ 11111\ 11111$$

The periods at the beginning and end of the expression are a convenience to enable us to determine where one such expression ends and another begins. We can, of course, establish representation for longer lists of symbols in a similar way.

Section 4
DIGITAL ARITHMETIC

4.1 INTRODUCTION

In this section we discuss arithmetic with the various types of numbers representable by digital words. It is of particular interest to define arithmetic operations on pairs of numbers in such a way that the results of the operations are again numbers representable by words in the same representation as the operands.

4.2 EXACT ARITHMETIC WITH INTEGERS

The first digital representation we described in Section 3, R_1, was a representation of the first 2^s non-negative integers by W_s, the set of words of length s.

Let p and q be integers with $p \leq q$; and let $I[p, q]$ be the set consisting of all integers i such that $p \leq i \leq q$.

The range of R_1 is $I[0, 2^s - 1]$. If i_1 and i_2 are in $I[0, 2^s - 1]$, then what can we say about $i_1 + i_2$, $i_1 - i_2$, $i_1 \times i_2$, and $i_1 \div i_2$? We can say only that $i_1 + i_2$ will be in $I[0, 2^{s+1} - 2]$; $i_1 - i_2$ will be in $I[-(2^s - 1), (2^s - 1)]$; $i_1 \times i_2$ will be in $I[0, (2^s - 1)^2]$; and if $i_2 \neq 0$, $i_1 \div i_2$ will be a rational number expressible as the sum of an integer in $I[0, 2^s - 1]$ and a proper fraction of the form i_3/i_2 for some i_3 in $I[0, i_2 - 1]$. The set $I[0, 2^s - 1]$ of the first 2^s non-negative integers is not closed with respect to any of the ordinary arithmetic operations.

If we wish the numbers $i_1 + i_2, i_1 - i_2, i_1 \times i_2, i_1 \div i_2$ to each be representable by a single word, we must suitably restrict the domain of each of the arithmetic operations involved.

If i_1 and i_2 are in $I[0, 2^s - 1]$ (and hence representable in R_1), then we may represent in R_1 the results of the four arithmetic operations on i_1 and i_2 by restricting the domain of "+" to integers i_1 and i_2 such that $i_1 + i_2 \leq 2^s - 1$, by restricting the domain of "-" to integers i_1 and i_2 such that $i_1 \geq i_2$, by restricting the domain of "x" to integers such that $i_1 \times i_2 \leq 2^s - 1$ and by restricting the domain of "$\div$" to integers i_1 and i_2 such that $i_1 \div i_2$ is an integer.

With these restrictions we can perform exact arithmetic with non-negative integers, the result of each arithmetic operation again being representable by a single word of length s.

4.3 FINITE PRECISION ARITHMETIC - SINGLE PRECISION, FIXED POINT

Another type of digital arithmetic we wish to consider is <u>finite precision</u> or "rounded" arithmetic. In this case approximations to the exact results of the arithmetic operations are computed which involve saving a limited number of the leading significant digits of the results.

The third representation described in Section 3, R_3, was a representation of "fractional" numbers or "single-precision fixed-point digital numbers." We recall that if $w = b_o b_1 \ldots b_{s-1}$, then

$$R_3(w) = (-1)^{b_o} \sum_{j=1}^{s-1} b_j \, 2^{-j}$$

The range of R_3 is the set $S_3 = \left\{ R_3(w) \mid w \in W_s \right\}$*.

If r is a real number, then let $[r]$ stand for "the greatest integer less than or equal to r".

* We use the notation $\{p \mid q\}$ to mean the set of p such that q is true, e.g. $\{R_3(w) \mid w \in W_s\}$ is the set of $R_3(w)$ such that w is in W_s.

We define single-precision fixed-point arithmetic operations, $\oplus$, $\ominus$, $\otimes$, $\oplus\!\!\!\!\div$ as follows:

For a, b such that $a \in S_3$, $b \in S_3$,

$$a \otimes b = (\text{sgn } ab) \left[|ab| \; 2^{s-1} \right] 2^{-s+1}$$

For a, b such that $b \neq 0$, $b \in S_3$, $|a| < |b|$

$$a \,\textcircled{\div}\, b = (\text{sgn } ab) \left[|a/b| \; 2^{s-1} \right] 2^{-s+1}$$

For a, b such that $b \in S_3$, $a \in S_3$, $|a + b| < 1$,

$$a \oplus b = a + b$$

For a, b such that $b \in S_3$, $a \in S_3$, $|a - b| < 1$,

$$a \ominus b = a - b$$

If r is a real number with the signed binary expansion

$$r = (\pm)\, b_{-n}\, b_{-n+1} \cdots b_o \,.\, b_1\, b_2 \cdots$$

i.e.

$$r = (\pm) \sum_{i=-n}^{\infty} b_i \cdot 2^{-i}$$

then the operation $(\text{sgn } r)\left[|r|\, 2^{s-1} \right] 2^{-s+1}$ truncates the series above at the term $i = s - 1$ i.e.

$$(\text{sgn } r)\left[|r|\, 2^{s-1} \right] 2^{-s+1} = (\pm)\, b_{-n} \ldots b_o \,.\, b_1 \ldots b_{s-1} = (\pm) \sum_{i=-n}^{s-1} b_i \, 2^{-i}$$

For example, if $s = 7$ and $r = -1/5$, then

$$\begin{aligned}(\text{sgn } r)\left[|r|\, 2^{s-1} \right] 2^{-s+1} &= (-1)\left[|-1/5|\, 2^{6} \right] 2^{-6} \\ &= (-1)\left[12\;4/5 \right] 2^{-6} \\ &= -\,3/16\end{aligned}$$

or in the binary expansion

$$(\text{sgn } r)\left[|r|\ 2^{s-1}\right] = (-1)\left[(.00110011\ldots)\ 2^6\right] 2^{-6}$$
$$= -\ .001100$$
$$= -\ .0011$$

Clearly, we have $\left| r - (\text{sgn } r)\left[|r|\ 2^{s-1}\right] 2^{-s+1}\right| \leq 2^{-s+1}$.

The operations $\oplus$, $\ominus$, $\otimes$, $\oplus\!\!\!\!\div$ may be regarded as functions mapping certain subsets (as described in their definitions) of $S_3 \times S_3$ into S_3. In other words, if (a, b) is in the domain of the operation $\oplus\!\!\!\!\div$, then $a \oplus\!\!\!\!\div b$ is representable by R_3 , etc.

If a and b are in S_3 , then $a \otimes b$ is in S_3 and we will have $|a \otimes b - ab| < 2^{-s+1}$;
and if $|a| < |b|$ also holds, then $a \oplus\!\!\!\!\div b$ is in S_3 and we will have
$|a \oplus\!\!\!\!\div b - a/b| \leq 2^{-s+1}$, for $b \neq 0$.

The first inequality in the above statement is easily seen, for example, from the identity
$|a \otimes b - ab|\ 2^{s-1} = |ab|\ 2^{s-1} - \left[|ab|\ 2^{s-1}\right]$.

The relations $a \oplus b = b \oplus a$, $(a \oplus b) \oplus c = a \oplus (b \oplus c)$, $a \otimes b = b \otimes a$ are obviously true for a, b, c in the domains of the respective operations while the relations $a \otimes (b \otimes c) = (a \otimes b) \otimes c$ and $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$ are <u>not</u> satisfied for every proper triplet a, b, c.

For example, let $s = 4$, $a = .111$. $b = .101$, $c = .010$,

then $a \otimes b = .111 \otimes .101 = .100$

$(a \otimes b) \otimes c = .100 \otimes .010 = .001$

but $b \otimes c = .101 \otimes .010 = .001$

$a \otimes (b \otimes c) = .111 \otimes .001 = .000$

$\neq (a \otimes b) \otimes c$;

also $b \oplus c = .101 \oplus .010 = .111$

$$a \otimes (b \oplus c) = .111 \otimes .111 = .110$$

but

$$a \otimes b = .111 \otimes .101 = .100$$

$$a \otimes c = .111 \otimes .010 = .001$$

$$(a \otimes b) \oplus (a \otimes c) = .101$$

$$\neq a \otimes (b \oplus c) .$$

Single precision arithmetic operations do not, then, enjoy the same relations as ordinary arithmetic operations.

The particular definitions given above for $\otimes$, $\oplus$ are "unrounded" versions of single precision arithmetic. We could have chosen instead to give the "rounded" versions which would be

$$a \otimes_r b = (\text{sgn } ab) \left[|ab| \, 2^{s-1} + 1/2 \right] 2^{-s+1}$$

and

$$a \oplus_r b = (\text{sgn } ab) \left[|a/b| \, 2^{s-1} + 1/2 \right] 2^{-s+1}$$

If we recompute for $s = 4$, $a = .111$, $b = .101$, $c = .010$ as in the above example but with the rounded operation $\otimes_r$, we will find that

$$a \otimes_r b = .111 \otimes_r .101 = .100$$

$$(a \otimes_r b) \otimes_r c = .100 \otimes_r .010 = .001$$

and

$$b \otimes_r c = .101 \otimes_r .010 = .001$$

$$a \otimes_r (b \otimes_r c) = .111 \otimes_r .001 = .001$$

$$= (a \otimes_r b) \otimes_r c ;$$

also

$$b \oplus c = .101 \oplus .010 = .111$$

$$a \otimes_r (b \oplus c) = .111 \otimes_r .111 = .110$$

and

$$a \otimes_r b = .111 \otimes_r .101 = .100$$

$$a \otimes_r c = .111 \otimes_r .010 = .010$$

$$(a \otimes_r b) \oplus (a \otimes_r c) = .110$$

$$= a \otimes_r (b \oplus c)$$

However, if we take $s = 4$, $a = .010$, $b = .110$, $c = .110$, then

$$a \otimes_r b = .010 \otimes_r .110 = .010$$

$$(a \otimes_r b) \otimes_r c = .010 \otimes_r .110 = .010$$

but

$$b \otimes_r c = .110 \otimes_r .110 = .101$$

$$a \otimes_r (b \otimes_r c) = .010 \otimes_r .101 = .001$$

$$\neq (a \otimes_r b) \otimes_r c \,;$$

Rounding, then, does not restore associativity or distributivity.

Without ambiguity we may also use the notation $w_1 \oplus w_2$, $w_1 \ominus w_2$, $w_1 \otimes w_2$, $w_1 \oslash w_2$ to stand for words w_3, w_4, w_5, w_6 in W_s such that $R_3(w_3) = R_3(w_1) \oplus R_3(w_2)$, $R_3(w_4) = R_3(w_1) \ominus R_3(w_2)$, $R_3(w_5) = R_3(w_1) \otimes R_3(w_2)$, $R_3(w_6) = R_3(w_1) \oslash R_3(w_2)$, whenever $(R_3(w_1), R_3(w_2))$ is in the domain of the operation in question.

4.4 FINITE PRECISION ARITHMETIC - DOUBLE PRECISION, FIXED POINT

The sixth representation described in section 3, R_6, was a representation of double precision fractional numbers. If $w = b_o b_1 \ldots b_{s-1}$ and $w' = b'_o b'_1 \ldots b'_{s-1}$, then

$$R_6(w, w') = (-1)^{b_o} \left\{ \sum_{j=1}^{s-1} b_j 2^{-j} + \sum_{j=0}^{s-1} b'_j 2^{-s-j} \right\} .$$

The range of R_6 is the set

$$S_6 = \left\{ R_6(w, w') \mid (w, w') \in W_s \times W_s \right\} .$$

We will use the notation $w = 0$ when $b_o = b_1 = \ldots = b_{s-1} = 0$ if no ambiguity results.

We define double-precision fixed point arithmetic operations, $\oplus_2, \ominus_2, \otimes_2, \oslash_2$ as follows:

for a, b such that $a \epsilon S_6$, $b \epsilon S_6$:

1) $a \otimes_2 b = (\text{sgn } ab) \left[|ab| \; 2^{2s-1} \right] 2^{-2s+1}$,

2) if $b \neq 0$, $|b| > |a|$,

$a \oslash_2 b = (\text{sgn } ab) \left[|a/b| \; 2^{2s-1} \right] 2^{-2s+1}$,

3) if $|a + b| < 1$,

$a \oplus_2 b = a + b$,

and 4) if $|a - b| < 1$

$a \ominus_2 b = a - b$.

The operations $\oplus_2$, $\ominus_2$, $\otimes_2$, $\oslash_2$ may be regarded as functions mapping certain subsets (described in their definitions) of $S_6 \times S_6$ into S_6 .

We use the notation $(w_1, w_1') \oplus_2 (w_2, w_2')$, etc. to stand for pairs of words (w_3, w_3') such that $R_6(w_3, w_3') = R_6(w_1, w_1') \oplus_2 R_6(w_2, w_2')$, etc.

For pairs of words of the form $(w, 0)$, the operations $\oplus_2$, $\ominus_2$, $\otimes_2$, $\oslash_2$, are related to the operations $\oplus$, $\ominus$, $\otimes$, $\oslash$, in the following way:

$$(w_1, 0) \oplus_2 (w_2, 0) = (w_1 \oplus w_2, 0)$$
$$(w_1, 0) \ominus_2 (w_2, 0) = (w_1 \ominus w_2, 0)$$
$$(w_1, 0) \otimes_2 (w_2, 0) = (w_1 \otimes w_2, w_3)$$
$$(w_1, 0) \oslash_2 (w_2, 0) = (w_1 \oslash w_2, w_4)$$

where w_3 and w_4 are words representing the "less significant halves" of the results.

To illustrate,

let $s = 4$ and let $w_1 = 0111$, $w_2 = 0101$,

then $R_6(w_1, 0) \otimes_2 R_6(w_2, 0) = .1110000 \otimes_2 .1010000 = .1000110$

or $(w_1, 0) \otimes_2 (w_2, 0) = (0100, 0110)$

The word 0100 is, in fact, $w_1 \otimes w_2$; i.e.,

$$R_3(0111) \otimes R_3(0101) = .111 \otimes .101 = .100$$
$$= R_3(0100) .$$

We observe that $R_6(w, 0) = R_3(w)$ for any word w in W_s ; conversely $R_3(w) \in S_6$ for any w in W_s since if we choose $w' = 0$, then $R_3(w) = R_6(w, 0)$ and $R_6(w, 0) \in \left\{R_6(w, w') \mid (w, w') \in W \times W\right\}$.

If $a = R_3(w)$ and $b = R_3(w')$,

then $|a + b| < 1$ implies $a \oplus_2 b = a \oplus b = a + b$,

and $|a - b| < 1$ implies $a \oplus_2 b = a \oplus b = a + b$,

and $a \otimes b = (\text{sgn } ab) \left[|a \otimes_2 b| \ 2^{s-1} \right] 2^{-s+1}$

and if $b \neq 0$, $|b| > |a|$, then

$$a \oplus b = (\text{sgn } ab) \left[|a \oplus_2 b| \ 2^{s-1} \right] 2^{-s+1}$$

4.5 EXACT ARITHMETIC WITH INTEGERS WITH FINITE PRECISION ARITHMETIC

From the arithmetic operations we have already defined we can construct restricted exact arithmetic operations with integers.

The second representation described in section 3, R_2, was a representation of the integers in $I\left[-(2^{s-1}-1)\ ,\ +\ (2^{s-1}-1)\right]$.

If $w = b_o b_1 \ldots b_{s-1}$, then

$$R_2(w) = (-1)^{b_o} \sum_{j=1}^{s-1} b_{s-j} 2^{j-1}$$

We observe that $R_2(w) = 2^{s-1} R_3(w)$, i.e.,

$$(-1)^{b_o} \sum_{j=1}^{s-1} b_{s-j} 2^{j-1} = 2^{s-1} \left\{(-1)^{b_o} \sum_{j=1}^{s-1} b_j 2^{-j}\right\}$$

Therefore if w and w' are in W_s, we have $R_3(w) = 2^{1-s} R_2(w)$, $R_3(w') = 2^{1-s} R_2(w')$ and $R_3(w) \otimes R_3(w') = 2^{1-s} R_2(w) \otimes 2^{1-s} R_2(w')$. But $R_3(w) = R_6(w, 0)$ and $R_3(w') = R_6(w', 0)$; and $R_2(w)$ and $R_2(w')$ are integers, say p_1 and p_2 respectively. Then $2^{1-s} p_1 \otimes_2 2^{1-s} p_2 = (\text{sgn } p_1 p_2)\left[\left|p_1 p_2\right| 2^1\right] 2^{-2s+1}$

$$= p_1 p_2 \ 2^{-2s+2}$$

Furthermore, if $\left|p_1 p_2\right| < 2^{s-1}$, then

$$\left|2^{1-s} p_1 \otimes_2 2^{1-s} p_2\right| = \left|p_1 p_2 \ 2^{-2s+2}\right| < 2^{-s+1}$$

and $\left\{2^{1-s} p_1 \otimes_2 2^{1-s} p_2\right\} \oplus_2 2^{-s+1}$

$$= (\text{sgn } p_1 p_2)\left[\frac{p_1 p_2 2^{-2s+2}}{2^{-s+1}} \ 2^{2s-1}\right] 2^{-2s+1} = p_1 p_2 2^{-s+1} .$$

We have shown that if w and w' are words in W_s such that $\left|R_2(w) R_2(w')\right| < 2^{s-1}$, then $\left\{R_6(w, 0) \otimes_2 R_6(w', 0)\right\} \oplus_2 2^{-s+1}$

$$= R_6(w'', 0) \ \text{ such that } \ R_2(w'') = R_2(w) R_2(w') .$$

The number 2^{-s+1} may be represented in R_6 by the pair $(00\ldots01, 0)$. Beginning with the words w and w' representing in R_2 integers p_1 and p_2 respectively such that $|p_1p_2| < 2^{s-1}$, we may compute the exact product p_1p_2 and represent it in R_2 by the word w'' such that $R_6(w'', 0) = \left\{R_6(w, 0) \otimes_2 R_6(w', 0)\right\} \oplus_2 2^{-s+1}$.

For the addition of integers, we note that if w and w' are words in W_s representing in R_2, integers p_1 and p_2 such that $R_2(w) = p_1$, $R_2(w') = p_2$, and $|p_1 + p_2| < 2^{s-1}$, then $R_3(w) = 2^{1-s} p_1$, $R_3(w') = 2^{1-s} p_2$, and $|R_3(w) + R_3(w')| < 1$, hence

$$\begin{aligned} R_3(w) \oplus R_3(w') &= R_3(w) + R_3(w') \\ &= (p_1 + p_2)\, 2^{-s+1} \\ &= R_3(w'') \text{ for } w'' \\ \text{such that } R_2(w'') &= R_2(w) + R_2(w') . \end{aligned}$$

For subtraction if $|p_1 - p_2| < 2^{s-1}$, then

$$\begin{aligned} R_3(w) \ominus R_3(w') &= (p_1 - p_2)\, 2^{-s+1} \\ &= R_3(w'') \text{ for } w'' \\ \text{such that } R_2(w'') &= R_2(w) - R_2(w') . \end{aligned}$$

For division, if $p_2 \neq 0$ and $p_1 \equiv 0 \pmod{p_2}$, then $|R_6(w, 0)\, 2^{-s+1}| < |R_6(w', 0)|$ and

$$\begin{aligned} R_6(w, 0)\, 2^{-s+1} \oplus_2 R_6(w', 0) &= (\operatorname{sgn} p_1p_2) \left[\frac{p_1 2^{-s+1}}{p_2} \quad 2^{2s-1} \right] 2^{-2s+1} \\ &= p_1/p_2\, 2^{-s+1} \\ &= R_3(w'') \text{ for } w'' \\ \text{such that } R_2(w'') &= R_2(w)/R_2(w') . \end{aligned}$$

In other words, if w and w' are words in W_s representing in R_2 integers p_1 and p_2 respectively such that $p_1 \equiv 0 \pmod{p_2}$, then we may compute the exact quotient p_1/p_2 and represent it in R_2 by a word w'' such that

$$R_6(w'', 0) = \left(R_6(w, 0) \otimes_2 2^{-s+1}\right) \oplus_2 R_6(w', 0) .$$

4.6 FINITE PRECISION ARITHMETIC – SINGLE PRECISION FLOATING POINT

The fourth representation given in section 3, R_4 was that of floating point numbers.

We recall that n and c are fixed positive integers such that $n+c+2 = s-1$, and the word $w = b_o b_1 \ldots b_n \; b_{n+1} \ldots b_{n+c+2}$ represents in R_4 the number

$$R_4(w) = m(w)\, 2^{c(w)}$$

$$\text{where } m(w) = (-1)^{bo} \sum_{j=0}^{c} b_j 2^{-j}$$

$$\text{and } c(w) = (-1)^{b_{n+1}} \sum_{j=0}^{c} b_{n+2+j} \; 2^{c-j}$$

If $b_1 = 1$, then we call $R_4(w)$ a "normalized" floating point number. Call $\hat{W}_s$ the set of words $w = b_o b_1 --- b_{s-1}$ in W_s such that $b_1 = 1$; call $S_4 = \left\{R_4(w) \mid w \in W_s\right\}$; and call $\hat{S}_4 = \left\{R_4(w) \mid w \in \hat{W}_s\right\}$.

If a and b are in S_4, then $a = x_1 2^{y_1}$ and $b = x_2 2^{y_2}$ for some x_1, x_2, y_1, y_2, $1/2 \leq |x_1| < 1$, $1/2 \leq |x_2| < 1$, $0 \leq |y_1| \leq 2^{c+1} - 1$, $0 \leq |y_2| \leq 2^{c+1} - 1$.

Let $y = \max(y_1, y_2)$, then

$$a + b = x_1 2^{y_1} + x_2 2^{y_2} = (x_1 2^{y_1 - y} + x_2 2^{y_2 - y}) 2^y$$

and $|a + b| < 2^{y+1}$. If $y + 1 \le 2^{c+1} - 1$, i.e. if $\max(y_1, y_2) \le 2^{c+1} - 2$,

then the floating point sum $a \hat{+} b$ defined by

$$a \hat{+} b = \operatorname{sgn}(a + b) \left[|a + b| 2^{-y+n-1} \right] 2^{y-n+1}$$

is in S_4 . Furthermore,

$$|(a + b) - (a \hat{+} b)| \cdot 2^{-y} = |a + b| 2^{-y} - \left[|a + b| 2^{-y+n-1} \right] 2^{-n+1}$$

hence $|(a + b) - (a \hat{+} b)| < 2^{-n+1+y}$. If $a = x_1 \cdot 2^{y_1}$ is in $\hat{S}_4$ but b is zero, i.e. $x_2 = 0$, then we define $a \hat{+} b = b \hat{+} a = a$.

To illustrate floating point addition, let $n = 3$, $c = 2$ and $s = n + c + 3 = 8$, then the numbers $.101 . 2^{100} = (2^{-1} + 2^{-3}) . 2^{2^2}$, and $-.110 . 2^{11} = -(2^{-1} + 2^{-2}) . 2^{2^1 + 2^0}$ are in $\hat{S}_4$ and $y = \max(100, 11) = \max(4, 3) = 4$. We have

$$(.101 . 2^{100}) + (-.110 . 2^{11}) = 1010.0 - 110.0 = 100.0 = 4 ;$$

also

$$(101 . 2^{100}) \hat{+} (-.110 . 2^{11} = (+1) \left[|4| \; 2^{-4+3-1} \right] . 2^2$$
$$= 4$$

However if $a = .101 . 2^{100}$ and $b = .110 . 2^1$, then a and b are in $\hat{S}_4$; but $a + b = 1010.0 + 1.1 = 1011.1 = 11\ 1/2$ while $a \hat{+} b = (+1) \left[|11\ 1/2| \; 2^{-4+3-1} \right] 2^2 = 8$.

In binary notation, $a + b = 1011.1$, $a \hat{+} b = 1000.0$; and in floating point notation $a + b = .10111 . 2^{100}$, $a \hat{+} b = .100 . 2^{100}$.

Finally, $|(a + b) - (a \hat{+} b)| = |11\ 1/2 - 8| = 3\ 1/2$ and $2^{-n+1+y} = 2^{-3+1+4} = 4$ and $3\ 1/2 < 4$ as promised.

The floating point numbers $.100 . 2^{100}$, $.010 . 2^{101}$, $.001 . 2^{110}$ are all in S_4 and all represent the integer 8 , however only $.100 . 2^{100}$ is in $\hat{S}_4$. The definition given above for "$\hat{+}$" guarantees that if a and b are in $\hat{S}_4$, then $a \hat{+} b$ is in S_4 .

If $x . 2^y$ is in S_4 and if $1/2 \leq |x| . 2^p$ for some integer p such that $p \leq y + 2^{c+1} - 1$, then $x . 2^y$ is in $\hat{S}_4$. To show this let $\bar{p}$ be the smallest p for which $p \leq y + 2^{c+1} - 1$ and $1/2 \leq |x| . 2^p$. Since $|x| < 1$, $\bar{p}$ is non-negative.

If

$$x = (-1)^{bo} \sum_{j=1}^{n} b_j \, 2^{-j}$$

and

$$y = (-1)^{b_{n+1}} \sum_{j=o}^{c} b_{n+2-j} \, 2^{c-j}$$

then

$$x . 2^{\bar{p}} = (-1)^{bo} \sum_{j=\bar{p}+1}^{n} b_j \, 2^{-j+\bar{p}}$$

with $b_{\bar{p}+1} = 1$, and $-(2^{c+1} - 1) \leq y - \bar{p} \leq y$, therefore $x . 2^y = (x . 2^{\bar{p}}) . 2^{y-\bar{p}}$ is in $\hat{S}_4$.

As a matter of fact if $x . 2^y$ is in S_4 , then either $x = 0$ or $|x| \geq 2^{-n}$. If $x \neq 0$, then $|x| . 2^{n-1} \geq 1/2$ and if $n - 1 \leq y + 2^{c-1} - 1$, then there is some p such that $p \leq n - 1$ and $1 > |x| . 2^p \geq 1/2$ and $p \leq y + 2^{c+1} - 1$. Therefore, if $y \geq n-2^{c-1}$, then either $x = 0$ or $x . 2^y$ is in $\hat{S}_4$.

Again, if a and b are in $\hat{S}_4$, then $a = x_1 . 2^{y_1}$ and $b = x_2 . 2^{y_2}$ for some x_1, x_2, y_1, y_2 such that $1/2 \leq |x_1| < 1$, $1/2 \leq |x_2| < 1$, $0 \leq |y_1| \leq 2^{c+1} - 1$, $0 \leq |y_2| \leq 2^{c+1} - 1$. Now if $|y_1 + y_2| \leq 2^{c+1} - 2$, then we define the floating point product $a \hat{x} b = \text{sgn}(ab) \left[|ab| \, 2^{-y_1 - y_2 + n} \right] 2^{-n + y_1 + y_2}$. Clearly $a \hat{x} b$ is in S_4; in fact it is in $\hat{S}_4$. Since $a \hat{x} b = x . 2^{y_1 + y_2}$ for some x such that $1/4 \leq |x| < 1$. If a in $\hat{S}_4$ and b is 0, then we define $a \hat{x} b = b \hat{x} a = 0$.

For floating point multiplication we evidently have for a and b in $\hat{S}_4$.

$$\left| \frac{a\,b - a \mathbin{\hat{\times}} b}{a^{b}} \right| < 2^{-n+2}$$

If a and -b are in the domain of the operation $\hat{+}$ defined above, then we define the floating point difference $a \mathbin{\hat{-}} b = a \mathbin{\hat{+}} (-b)$.

If a and b are in $\hat{S}_4$ and $a = x_1 \cdot 2^{y_1}$, $b = x_2 \circ 2^{y_2}$, then when $|y_1 - y_2| \leq 2^{c+1} - 2$, we define the floating point quotient

$$a \mathbin{\hat{\div}} b = (\operatorname{sgn} ab) \left[|a/b| \, 2^{-y_1 + y_2 + n - 1} \right] 2^{y_1 - y_2 - n + 1}$$

Clearly $a \mathbin{\hat{\div}} b$ is in $\hat{S}_4$ and

$$\left| \frac{a/b - a \mathbin{\hat{\div}} b}{a/b} \right| \leq 2^{-n+2}$$

For $a = 0$ and b in $\hat{S}_4$ we define $a \mathbin{\hat{\div}} b = 0$.

Note that we cannot bound the expression $\left| \frac{(a + b) - (a \mathbin{\hat{+}} b)}{a + b} \right|$ even for a and b in $\hat{S}_4$ since $a + b$ may be zero.

Section 5
ARITHMETIC COMPUTATIONS BY THE COMPUTER

5.1 INTRODUCTION

In the previous sections several digital representations of numbers have been considered. For some of these representations digital or "rounded" arithmetic operations have been defined in such a way that the result of an operation is again representable in the same representation as the operands. In doing so, it was necessary to design the operations to take care of two things: the number of digits in the result and the numerical size of the result. The former was required because of the fixed length of the words used in each representation and the latter was required because the range of each representation was a bounded set of numbers.

In order to make possible a sequence of digital arithmetic operations in which the result of one operation is to become an operand of a subsequent digital arithmetic operation, the sequence of operations must be so arranged that each pair of operands will lie in the domain of the operation that is to be performed upon them. This is called <u>scaling</u> the computation.

For example, in single precision fixed point arithmetic we may compute $a - b + c$ for $s = 4$ and $a = .111$, $b = .101$, $c = .010$ providing we order the computation as $(a \ominus b) \oplus c$. We have $a \ominus b = .010$ and $(a \ominus b) \oplus c = .010 \oplus .010 = .100$. The computation $(a \oplus c) \ominus d$ is impossible. Since $a + c = 1.001 > 1$, the pair a, c is not in the domain of the operation $\oplus$.

In this section we will describe the types of operations which will be included in machine language in order to obtain the automatic computer execution of the computations with which we will be concerned. We will then show how a computation by the computer involving digital arithmetic may be programmed.

5.2 MACHINE LANGUAGE

We recall that a state σ is an ordered set $\{(C), (0), (1), \ldots, (m-1)\}$ whose elements are words contained in the address cell C and in the cells of the memory $M_{s,m}$. We recall that for a fixed positive integer k, $m = 2^k$; so that (C) is in W_k i.e., (C) is a word of the form $b_o b_1 \ldots b_{k-1}$.

We define R_o to be the representation mapping W_k onto the set of positive integers $I[0, m-1]$, such that for $u = b_o b_1 \ldots b_{k-1}$ is in W_k, $R_o(u) = \sum_{j=o}^{k-1} b_{k-1-j} 2^j$.

In the state σ, $R_o((C))$ is an integer in $I[0, m-1]$ and hence the address of some cell in $M_{s,m}$ say $R_o((C)) = A$. Also in state σ, (A) is some word in W_s, say $w = b_o b_1 \ldots b_{s-1}$.

Assume that $s \geq 3k + 4$. We define four functions A_1, A_2, A_3, and T in the following way:

1) A_1, A_2, A_3 are mappings from W_s onto W_k such that

$$A_1(b_o b_1 b_2 \ldots b_{s-1}) = b_o b_1 \ldots b_{k-1}$$

$$A_2(b_o b_1 b_2 \ldots b_{s-1}) = b_k b_{k+1} \ldots b_{2k-1}$$

$$A_3(b_o b_1 b_2 \ldots b_{s-1}) = b_{2k} b_{2k+1} \ldots b_{3k-1}.$$

2) T is a mapping from W_s onto W_{s-3k} such that $T(b_o b_1 b_2 \ldots b_{s-1})$ $= b_{3k} b_{3k+1} \ldots b_{s-1}$

For any word w in W_s, $A_1(w)$, $A_2(w)$, and $A_3(w)$ are words in W_k and are therefore in the domain of R_o. In fact, $R_o(A_1(w))$, $R_o(A_2(w))$, $R_o(A_3(w))$ are intergers in $I[0, m-1]$, hence addresses of cells in $M_{s,m}$. We call $A_1(w)$, $A_2(w)$, $A_3(w)$, the first, second, and third address parts of w, respectively.

The word T(w) has length at least 4 , since $s - 3k \geq 4$. It will be used to represent the type of operation to be executed in computing σ' from σ . We chose 4 as the minimum length of T(w) because we wish to list 15 distinct operations. We can identify an object out of a set of 15 things by a word of length 4.

If α and β are addresses we use the notation $(\alpha) \rightarrow \beta$ to stand for the act of replacing the content of the cell β with the word in cell α . We say that "(α) goes to β ". This act is assumed to involve the passage of some time and changes the state of affairs in such a way that upon completion of the act we have $(\alpha) = (\beta)$; whereas before the act it may have been that $(\alpha) \neq (\beta)$. The act is assumed to leave (α) intact. It merely "copies" the word (α) into the cell β , obliterating whatever was there previously.

We will describe 15 operations each of whose names will be the symbol given first. The descriptions will list the sets of acts of which the operations consist. The operations in the list are chosen for illustrative purposes and the list is fairly redundant; for example $a \oplus b$ may be replaced by $a \ominus (0 \ominus b)$.

If u is a word in W_k , let $\bar{u}$ be the word in W_k such that $R_o(\bar{u}) \equiv R_o(u) + 1 \pmod{2^k}$.

In state σ , let $R_o((C)) = A$, let $(A) = w$, and let $R_o(A_1(w)) = \alpha$, $R_o(A_2(w)) = \beta$, $R_o(A_3(w)) = \gamma$.

$\oplus$: $(\alpha) \oplus (\beta) \rightarrow \gamma$

$(\bar{C}) \rightarrow C$ (called a "unit increase" of $(\bar{C})$)

$\ominus$: $(\alpha) \ominus (\beta) \rightarrow \gamma$

$(\bar{C}) \rightarrow C$

$\otimes$: $(\alpha) \otimes (\beta) \rightarrow \gamma$

$(\bar{C}) \rightarrow C$

$\odiv$: $(\alpha) \odiv (\beta) \rightarrow \gamma$

$(\bar{C}) \rightarrow C$

$\oplus_2$: $((\alpha), (\alpha+1)) \oplus_2 ((\beta), (\beta+1)) \rightarrow (\gamma, \gamma+1)$

$(\bar{C}) \rightarrow C$

$\ominus_2$: $((\alpha), (\alpha+1)) \ominus_2 ((\beta), (\beta+1)) \rightarrow (\gamma, \gamma+1)$

$(\bar{C}) \rightarrow C$

$\otimes_2$: $((\alpha), (\alpha+1)) \otimes_2 ((\beta), (\beta+1)) \rightarrow (\gamma, \gamma+1)$

$(\bar{C}) \rightarrow C$

$\odiv_2$: $((\alpha), (\alpha+1)) \odiv_2 ((\beta), (\beta+1)) \rightarrow (\gamma, \gamma+1)$

$(\bar{C}) \rightarrow C$

In operations $\oplus_2$, $\ominus_2$, $\otimes_2$, $\odiv_2$ we mean of course that into $(\gamma, \gamma+1)$ go words (γ), $(\gamma+1)$ such that $R_6((\gamma), (\gamma+1))$ $R_6((\alpha), (\alpha+1)) \oplus_2 R_6((\beta), (\beta+1))$, etc.

$\hat{+}$: $(\alpha) \hat{+} (\beta) \rightarrow \gamma$

$(\bar{C}) \rightarrow C$

$\hat{-}$: $(\alpha) \hat{-} (\beta) \rightarrow \gamma$

$(\bar{C}) \rightarrow C$

$\hat{\times}$: $(\alpha) \hat{\times} (\beta) \rightarrow \gamma$

$(\bar{C}) \rightarrow C$

$\hat{\div}$: $(\alpha) \hat{\div} (\beta) \rightarrow \gamma$

$(\bar{C}) \rightarrow C$

The operations $\oplus$, $\ominus$, $\otimes$, $\oslash$ put words into γ such that

$R_3((\gamma)) = R_3((\alpha)) \oplus R_3((\beta))$, etc. The operations $\hat{+}$, $\hat{-}$, $\hat{\times}$, $\hat{\div}$ put words into γ such that

$R_4((\gamma)) = R_4((\alpha)) \hat{+} R_4((\beta))$, etc.

$\circledcirc$: If $R_3((\alpha)) \leq R_3((\beta))$, then $\gamma \to C$, otherwise, i.e. if $R_3((\alpha)) > R_3((\beta))$,

$(\bar{C}) \to C$

$\circledcirc_2$: If $R_6((\alpha), (\alpha+1)) \leq R_6((\beta), (\beta+1))$,

then $\gamma \to C$,

otherwise $(\bar{C}) \to C$

$\hat{\leq}$: If $R_4((\alpha)) \leq R_4((\beta))$,

then $\gamma \to C$

otherwise $(\bar{C}) \to C$

We represent the operations listed above by words in W_{s-3k} . Let $0P$ be a mapping from W_{s-3k} to a set containing the operations above and such that
$0P(00\ldots00) = \oplus$

$$0P(00\ldots01) = \ominus$$

$$0P(00\ldots10) = \otimes$$

$$\cdots$$

$$0P(00\ldots01110) = \hat{\leq}$$

The operation $\hat{\leq}$, for example, is represented in $0P$ by the word $00..01110$. The number of 0's preceding the four 1's will be $s-(3k+4)$.

The operations we have listed consist of three sets of five each. Each set consists of four digital arithmetic operations each with an accompanying "unit increase" in the address contained in C . The fifth operation in each set is a "conditional transfer" and compares the numbers represented by the words in α and β (or by the pairs of words in $(\alpha, \alpha+1)$ and $(\beta, \beta+1)$). If the number represented by (α) is less than or equal to the number represented by (β) , in the representation of concern, then the address γ goes to the address cell, C , otherwise the usual "unit increase" takes place.

By now, we have already defined the transformation ML which carries a state σ into its successor σ' .

We recapitulate the <u>definition of ML</u> . Let $\sigma = \left|(C), (0), (1), \ldots, (m-1)\right|$ and $\sigma' = \left|(C)', (0)', \ldots, (m-1)'\right|$.

If $(C) = A$, and $(A) = w$, such that $R_0(A_1(w)) = \alpha$, $R_0(A_2(w)) = \beta$, and $OP(T(w)) = op$, then

$$(C)' = \begin{cases} \gamma \ \text{ if: } & op = \textcircled{\le} \text{ and } R_3((\alpha)) \le R_3((\beta)), \\ & op = \textcircled{\le}_2 \text{ and } R_6((\alpha), (\alpha+1)) \le R_6(((\beta)), (\beta+1)), \\ & \text{or } op = \hat{\le} \text{ and } R_4((\alpha)) \le R_4(((\beta)) . \\ \bar{C} & \text{otherwise} \end{cases}$$

$$(\gamma)' = \begin{cases} (\alpha)\, op\, (\beta) \ \text{ if } op \in \left| \oplus, \ominus, \otimes, \textcircled{\div}, \hat{+}, \hat{-}, \hat{\times}, \hat{\div} \right| \\ w_1 \quad \text{if } op \in \left| \oplus_2, \ominus_2, \otimes_2, \textcircled{\div}_2 \right| \\ \text{where } (w_1, w_2) = ((\alpha), (\alpha+1)) \ op \ ((\beta), (\beta+1)) \\ \text{for some } w_2 , \\ (\gamma) \ \text{ otherwise} \end{cases}$$

$$(\gamma+1)' = \begin{cases} w_2 \text{ if op } \in \left| \oplus_2 , \ominus_2 , \otimes_2 , \oplus_2 \right| \\ \quad \text{where } ((\gamma)', w_2) = ((\alpha), (\alpha+1)) \text{ op } ((\beta), (\beta+1)) \\ (\gamma+1) \text{ otherwise} \end{cases}$$

$(i)' = (i)$ for $i \in I[0, m-1]$, $i \neq \gamma, \gamma + 1$.

5.3 PROGRAMMING A DIGITAL COMPUTATION IN MACHINE LANGUAGE

If i is the address of some cell in $M_{s,m}$, we may denote the structure of the word in that cell as an <u>instruction</u> by

$$(i) : \alpha \quad \beta \quad \gamma \quad \text{op}$$

meaning (i) is a word w in W_s such that $R_o(A_1(w)) = \alpha$, $R_o(A_2(w)) = \beta$, $R_o(A_3(w)) = \gamma$ and $OP(T(w)) = \text{op}$. Using this notation we easily see that if in state σ

$$(c) = i$$

and

$$(i) : \alpha \quad \alpha \quad i \; \circledS,$$

then σ is a terminal state.

Clearly, for any α, $R_3(\alpha) \leq R_3(\alpha)$, therefore $(c)' = (c)$, $(\gamma)' = (\gamma)$, $(\gamma+1)' = (\gamma+1)$, $(i)' = (i)$ for $i \in I[0, m-1]$, $i \neq \gamma, \gamma + 1$ and hence $\sigma' = \sigma$.

If in state σ, $(c) = i$, then the following also imply that σ is a terminal state:

$$(i) : \alpha \quad \alpha \quad i \; \circledS_2$$

$$(i) : \alpha \quad \alpha \quad i \; \hat{\leq}$$

If x is a number in S_3, the range of R_3, we can "program" <u>the computation of $|x|$</u> as follows: choose $\sigma_0 = \left|(c)_o, (0)_o, (1)_o, \ldots, (m-1)_o\right|$ such that for distinct α, β, i, i + 1, ..., i + 5, we have $x = R_3((\alpha)_o)$, $(c)_o = i$, $(i)_o$, $(i+1)_o, \ldots$ as indicated in the following table:

Instruction List	Comments
$(i)_o$: $\alpha\ \alpha\ \beta$ $\ominus$	computes the number 0, $0 \rightarrow \beta$
$(i+1)_o$: $\beta\ \alpha$ i + 4 $\textcircled{\leq}$	$0 \leq x$? yes: take i + 4 no : take i + 2
$(i+2)_o$: $\beta\ \alpha\ \gamma$ $\ominus$	$0 - x \rightarrow \gamma$ for $x < 0$
$(i+3)_o$: $\alpha\ \alpha$ i + 3 $\textcircled{\leq}$	produces a terminal state when (C) = i + 3 .
$(i+4)_o$: $\beta\ \alpha\ \gamma$ $\oplus$	$0 + x \rightarrow \gamma$ for $x \geq 0$
$(i+5)_o$: $\alpha\ \alpha$ i + 3 $\textcircled{\leq}$	"Transfer" to i + 3 . Note that we could use i + 5 : $\alpha\ \alpha$ i + 5 as well.

The program σ_o just described will produce the computation $\sigma_o, \sigma_o', \sigma_o'', \sigma_o'''$ with (c) running through the values i, i + 1, i + 2, i + 3 if $x < 0$ i.e. if $R_3((\alpha)_o) < 0$. However, if $x \geq 0$ i.e. if $R_3((\alpha)_o) \geq 0$, then the computation $\sigma_o, \sigma_o', \sigma_o'', \sigma_o''', \sigma_o''''$ will result with (c) running through the values i, i + 1, i + 4, i + 5, i + 3 . The two computations begin with different programs, of course, since $(\alpha)_o$ is part of σ_o. In either case, however, the computer is left in a terminal state with $|x|$ in γ.

Since the program uses 9 cells, we must have $m \geq 10$ for the program not to exceed the memory "capacity" of the computer.

If $x_1, x_2, \ldots, x_n$ are in S_3 and $|x_i| < \frac{1}{n}$ for = 1, 2, ., n , we can program the computation $x_1 \oplus x_2 \oplus x_3 \oplus \ldots \oplus x_n$ in two essentially different ways.

a) The "straight line" program.

Choose $\sigma_o = \left\{ (C)_o, (0)_o, (1)_o, \ldots, (n-1)_o \right\}$ such that $x_i = R_3\big((i-1)_o\big)$ for $i = 1, 2, \ldots, n$ and $(c)_o = n$, with $(n)_o$, $(n+1)_o, \ldots, (2\ \ n-2)_o$, as indicated in the following table:

Instruction List	Comments
$(n)_o$: $0\ 1\ \gamma\ \oplus$	$x_1 \oplus x_2 \rightarrow \gamma$
$(n+1)_o$: $\gamma\ 2\ \gamma\ \oplus$	$R_3((\gamma)) \oplus x_3 = x_1 \oplus x_2 \oplus x_3 \rightarrow \gamma$
$(n+2)_o$: $\gamma\ 3\ \gamma\ \oplus$	$x_1 \oplus x_2 \oplus x_3 \oplus x_4 \rightarrow \gamma$
$(n+n-2)_o$: $\gamma\ n-1\ \gamma\ \oplus$	$x_1 \oplus x_2 \oplus --- \oplus x_n \rightarrow \gamma$
$(n+n-1)_o$: $0\ \ 0\ \ n+n-1\ ⊜$	produces terminal state

The program will produce a finite computation ending in a terminal state in which cell γ (which may be chosen to be cell $2n$, for example) will contain word (γ) such that

$$R_3((\gamma)) = x_1 \oplus x_2 \oplus \ldots\ldots \oplus x_n$$

b) The "inductive" program.

Choose $\sigma = \left\{ (C)_o, (0)_o, (1)_o, \ldots, (n-1)_o \right\}$ such that $R_3((0)_o) = 0$, $R_3((i-1)_o) = x_i$ for $i = 1, 2, \ldots, n$ and $(C)_o = n+3$, with $(n+1)_o$, $(n+2)_o \ldots, (n+7)_o$ as indicated in the following table:

Instruction List	Comments
$(n+1)_o$: 0 n 0 $\oplus$	"program constants"
$(n+2)_o$: 0 1 0 0	
$(n+3)_o$: 0 1 0 $\oplus$	Initially $0 \oplus x_1 \rightarrow 0$; <u>during</u> computation, becomes $(o) + x_k \rightarrow 0$
$(n+4)_o$: n+1 n+3 n+7 $\circledleq$	If $n \leq k$, (i.e. $n = k$), take $n + 7$ if $n > k$ take $n + 5$
$(n+5)_o$: n+2 n+3 n+3 $\oplus$	Since $k < n$, put R_3 (0 k+1 0 $\oplus$) $\oplus$ R_3 (0100) $\rightarrow$ n+3 so that (n+3) becomes 0 k+1 0 $\oplus$
$(n+6)_o$: 0 0 n+3 $\circledleq$	(since $0 \leq 0$) take $n + 3$
$(n+7)_o$: 0 0 n+7 $\circledleq$	Produces terminal state, i.e. halts computation

The program produces a finite computation ending with $x_1 \oplus x_2 \oplus \ldots \oplus x_n$ in cell 0 .

Note that the "straight line" program described in a) requires that $m \geq 2n + 1$ and produces a computation of length $n - 1$. The inductive program given in b) requires that $m \geq n + 8$ and produces a computation of length $4n - 2$. If $n > 7$, then the straight line program in a) requires more memory cells than does the inductive program in b) . If $n > 1$, then the computation produced in b) is longer than that produced in a).

We say that a program <u>uses cell a</u> if a appears as an address part of some instruction during the computation produced by the program. That is, if $(C) = w$ in some state σ in the computation such that $A_1(w) = a$ or $A_2(w) = a$, or $A_3(w) = a$.

If p_1 and p_2 are programs which produce finite computations using mutually exclusive sets of cells s_1 and s_2, then p_1 and p_2 may be "joined" by replacing the terminal instructions in p_1 by transfers to the first instruction in p_2.

For example, if in the program above for $|x|$, none of the cells α, β, γ, i, $i + 1$, ..., $i + 5$ are among the first $n + 8$, then we may join that program to the inductive program in b) above for computing $x_1 \oplus x_2, \oplus \ldots \oplus x_n$ by changing $(n + 7)_o$ from $\left| 00\ n + 7 \circledleq \right|$ to $\left| 00i \circledleq \right|$ and starting with σ_o as the composite program such that $(c)_o = n + 3$, $x = R_3((\alpha)_o)$, $(0)_o$, ..., $(n + 6)_o$ as indicated in the table in b) above and $(i)_o$, $(i + 1)_o$, ..., $(i + 5)_o$ as in the table for $|x|$ above, and with $(n + 7)_o = 00\,i \circledleq$ as just mentioned. The composite program will produce a computation which terminates with $|x|$ in cell γ <u>and</u> $x_o \oplus x_1 \oplus - - - \oplus x_n$ in cell 0.

We might also join the two programs just discussed in the same way as described except choosing 0 for γ. Then the number $x_1 \oplus x_2 \oplus - - - \oplus x_n$ will be taken for x and the composite program will produce $|x_1 \oplus x_2 \oplus - - - \oplus x_n|$ in cell 0.

We may test for the equality of two numbers in S_3 represented by words w_1 and w_2 by successively testing, $R_3(w_1) \leq R_3(w_2)$ and $R_3(w_2) \leq R_3(w_1)$. If both inequalities are satisfied, obviously $R_3(w_1) = R_3(w_2)$. For numbers in S_3, we would use the operation $\circledleq$, for numbers in S_6 we use $\circledleq_2$, and for numbers in S_4, we use $\hat{\leq}$. The operations $\circledleq$, $\circledleq_2$, $\hat{\leq}$ are called <u>comparison</u> operations.

Using comparison operations we can obviously program such computations as: $\min \left\{ x_1, x_2, \ldots, x_n \right\}$, $\max \left\{ x_1, x_2, \ldots, x_n \right\}$, for numbers x_i in the range of some digital representation available on the computer.

Section 6
APPROXIMATIONS TO REAL ARITHMETIC

The field, R , of real numbers is closed not only with respect to the arithmetic operations $+$, $-$, $\times$, $\div$ (division by 0 excluded) but also with respect to convergent sequences of real numbers.

If x, y are real numbers and o is one of the operations in the set $\{+, -, \times, \div\}$, then

$$\text{" } z = x \, o \, y \text{ "}$$

is a real <u>computational step</u> or simply a real <u>step</u>. If r and t are real steps, and $u, v \in R$, then

$$\text{" if } u \leq v \text{, do } r \text{; otherwise, do } t \text{ "}$$

is a real step.

Let S be a finite set of real numbers. If c is computable in a finite number of real steps from the set $S \cup \{0, 1\}$, then c is <u>directly computable</u> from the set S .

If $c_1, c_2, \ldots, c_n$ are directly computable from S , then the computation of the set $\{c_1, c_2, \ldots, c_n\}$ is called a <u>direct real computation.</u>

If $c_1, c_2, \ldots$ is a sequence of numbers each of which is directly computable from S , then the computation of the sequence $c_1, c_2, \ldots$ is an <u>indirect real computation.</u> If the sequence is convergent, its limit is <u>indirectly computable</u>.

Example 1) For $x, y \in R$, $\frac{|1 - x|}{1 + y^2}$ is directly computable from the set $\{x, y\}$ in the steps

1. $c_1 = 1 - x$
2. If $c_1 \leq 0$, do 3.; otherwise, do 4.
3. $c_2 = 0 - c_1$
4. $c_2 = 0 + c_1$
5. $c_3 = y^2$
6. $c_4 = 1 + c_3$
7. $\frac{|1 - x|}{1 + y^2} = c_2/c_4$

Example 2) For a, b, c, d in R, $\{ac - bd, \; bc + ad\}$ is directly computable from the set $\{a, b, c, d\}$ in the steps

1. $c_1 = ac$
2. $c_2 = bd$
3. $ac - bd = c_1 - c_2$
4. $c_4 = bc$
5. $c_5 = ad$
6. $bc + ad = c_4 + c_5$

Any number in S is directly computable from S since if $x \in S$, then $x = x + 0$.

If a number is directly computable from S for arbitrary S then it is directly computable.

Any integer n is directly computable. We reach n in $n + 1$ steps by

1. $c_1 = 0 + 0$
2. $c_2 = c_1 + 1$
3. $c_3 = c_2 + 1$

. . . . .

$n + 1$. $n = c_{n+1} = c_n + 1$

for non-negative n, and in $-n$ steps by

1. $c_1 = 0 - 1$

2. $c_2 = c_1 - 1$

$\cdot\ \cdot\ \cdot\ \cdot\ \cdot$

-n. $n = c_{-n} = c_{-n-1} - 1$ for negative n

A rational number is directly computable as the quotient of two integers obtained as above.

A real number is indirectly computable as the limit of a convergent sequence of rational numbers.

There are natural correspondences between <u>direct digital computations</u>, that is, finite computations by the computer and direct real computations. A real step corresponds to an operational cycle. The direct real computation given in example 1) above corresponds to a finite computation by the computer produced by a program describing say:

1. $c_1 = 1 \mathbin{\hat{-}} x$
2. if $c_1 \mathbin{\hat{\leqq}} 0$ do 3 ; otherwise do 4.
3. $c_2 = 0 \mathbin{\hat{-}} c_1$
4. $c_2 = 0 \mathbin{\hat{+}} c_1$
5. $c_3 = y \mathbin{\hat{\times}} y$
6. $c_4 = 1 \mathbin{\hat{+}} c_3$.
7. $|1 \mathbin{\hat{-}} x| \mathbin{\hat{\div}} (1 \mathbin{\hat{+}} y \mathbin{\hat{\times}} y) = c_2 \mathbin{\hat{\div}} c_4$

for 0, 1, x, y in R_4 .

The real computation in example 2) above corresponds to a finite computation by the computer produced by a program describing say:

1. $c_1 = a \otimes c$
2. $c_2 = b \otimes d$
3. $(a \otimes c) \ominus (b \otimes d) = c_1 \ominus c_2$
4. $c_4 = b \otimes c$
5. $c_5 = a \otimes d$
6. $(b \otimes c) \ominus (a \otimes d) = c_4 \oplus c_5$

Since a complex number can be represented by a pair of real numbers and since the results of complex arithmetic operations on pairs of complex numbers are direct real computations, the evaluation of a complex rational function is a direct real computation.

If $u = (a, b)$ and $v = (c, d)$ are complex numbers with a, b, c, d in R, then we recall that

$$\begin{aligned} u + v &= (a + c,\ b + d) \\ u - v &= (a - c,\ b - d) \\ uv &= (ac - bd,\ bc + ad) \\ u/v &= \left(\frac{ac + bd}{c^2 + d^2},\ \frac{bc - ad}{c^2 + d^2}\right) \quad \text{for } v \neq (0, 0)\ . \end{aligned}$$

If $a_1, a_2, a_3, \ldots a_n$ are real numbers such that $|a_n| < 1$ and $\left|\sum_{i=1}^{n} a_i\right| < 1$ for positive integer n, then we can define digital numbers

$$\bar{a}_i = (\text{sgn } a_i)\left[|a_i|\, 2^{s-1}\right] 2^{-s+1}$$

in S_3 such that

$$|\bar{a}_i - a_i| \leq 2^{-s+1}$$

The digital number

$$\bar{S}_n = \bar{a}_1 \oplus \bar{a}_2 \oplus \bar{a}_3 \oplus \bar{a}_4 \oplus \ldots . \oplus \bar{a}_n$$

in S_3 may be regarded as the result a direct digital computation approximating the directly computable number

$$S_n = \sum_{i=1}^{n} a_i\ ,$$

in which case the absolute error in $\bar{S}_n$ is

$$|\bar{S}_n - S_n| \leq n\ 2^{-s+1}$$

We recall from Section 4 that digital arithmetic is not associative or distributive. Although for real arithmetic we have $a(bc) = (ab)c$ and $a(b + c) = ab + ac$, it may be the case that for a, b, c in S_3 the digital numbers $(a \otimes b) \otimes c$ and $a \otimes (b \otimes c)$ are distinct and that $a \otimes (b \oplus c) \neq (a \otimes b) \oplus (a \otimes c)$.

The digital versions of algebraically equivalent real forms do not always produce identical digital approximations. The error in a digital approximation to a real computation will depend on the precise arrangement of operations in the digital approximation.

We will give now an example of an a priori analysis of the total absolute error in a direct digital computation.

Let a, b, c be real numbers such that $|bc| < |d|$ and $|a + bc/d| < 1$, then we can approximate the real computation $r = a + bc/d$ by the direct digital computation.

$$\bar{r} = \left((\bar{b} \otimes \bar{c}) \oslash \bar{d}\right) \oplus \bar{a}$$

with numbers $\bar{a}, \bar{b}, \bar{c}, \bar{d}$ in S_3 such that $|\bar{a} - a| \leq 2^{-s+1}$, $|\bar{b} - b| \leq 2^{-s+1}$ $|\bar{c} - c| \leq 2^{-s+1}$, $|\bar{d} - d| \leq 2^{-s+1}$.

Assume that $|\bar{b}\bar{c}| < |\bar{d}|$ and $|\bar{a} + \bar{b}\bar{c}/d| < 1$.

We decompose the total error as follows:

$$r - \bar{r} = a + \frac{bc}{d} - \left|\left((\bar{b} \otimes \bar{c}) \oslash \bar{d}\right) \oplus \bar{a}\right|$$

$$= a - \bar{a} + \frac{bc}{d} - \frac{\bar{b}\bar{c}}{d}$$

$$+ \left(\frac{b\bar{c}}{d} + \bar{a}\right) - \left|\left((\bar{b} \otimes \bar{c}) \oslash \bar{d}\right) \oplus \bar{a}\right|$$

Now

$$\frac{bc}{d} - \frac{\overline{bc}}{\bar{d}} = \frac{1}{d}\,(bc - \overline{bc}) + \frac{\overline{bc}}{d\bar{d}}\,(d - \bar{d})$$

$$= \frac{b}{d}\,(c - \bar{c}) + \frac{\bar{c}}{d}\,(b - \bar{b}) + \frac{\overline{bc}}{d\bar{d}}\,(d - \bar{d})$$

And

$$\left(\frac{\overline{bc}}{\bar{d}} + \bar{a}\right) - \left|\left((\bar{b} \otimes \bar{c}) \oplus\!\!\!\!\!\div \bar{d}\right) \oplus \bar{a}\right|$$

$$= \frac{\overline{bc}}{\bar{d}} - (\bar{b} \otimes \bar{c}) \,\text{⨸}\, \bar{d} \quad \text{since} \quad x \oplus y = x + y\,.$$

Furthermore

$$\frac{\overline{bc}}{\bar{d}} - (\bar{b} \otimes \bar{c}) \,\text{⨸}\, \bar{d}$$

$$= \frac{\overline{bc}}{\bar{d}} - \frac{\bar{b} \otimes \bar{c}}{\bar{d}} + \frac{\bar{b} \otimes \bar{c}}{\bar{d}} - \left(\bar{b} \otimes \bar{c}\right) \,\text{⨸}\, \bar{d}$$

Finally we have

$$r - \bar{r} = (a - \bar{a}) + \frac{b}{d}\,(c - \bar{c}) + \frac{\bar{c}}{d}\,(b - \bar{b})$$

$$+ \frac{\overline{bc}}{d\bar{d}}\,(d - \bar{d}) + \frac{1}{\bar{d}}\,(\overline{bc} - \bar{b} \otimes \bar{c})$$

$$+ \left(\frac{\bar{b} \otimes \bar{c}}{\bar{d}} - \left(\bar{b} \otimes \bar{c}\right) \,\text{⨸}\, \bar{d}\right)$$

The total error is found to consist of six terms containing the contributions from six sources of error involved: the approximation of the numbers a, b, c, d by $\bar{a}$, $\bar{b}$, $\bar{c}$, $\bar{d}$ and the approximation of the real arithmetic operations $\overline{bc}$ by $\bar{b} \otimes \bar{c}$ and $(\bar{b} \otimes \bar{c})/\bar{d}$ by $(\bar{b} \otimes \bar{c}) \,\text{⨸}\, \bar{d}$.

Recalling that for x, y in S_3

$$| x \otimes y - x y | < 2^{-s+1}$$

and for x, y in S_3 with $| x/y | < 1$,

that
$$| x \oplus y - x/y | < 2^{-s+1},$$

we find
$$| r - \bar{r} | < 2^{-s+1} \left(2 + \frac{4}{|\bar{d}| - 2^{-s+1}}\right) .$$

As a numerical example, take $s = 27$, $a = \bar{a} = -2^{-15}$, $b = \bar{b} = c = \bar{c} = 2^{-14}$, and $d = \bar{d} = 2^{-13}$. We have $r = a + bc/d = 0$ and $\bar{r} = (\bar{b} \otimes \bar{c}) \oplus \bar{d} \oplus \bar{a} = -2^{-15}$, since $\bar{b} \otimes \bar{c} = 2^{-14} \otimes 2^{-14} = 0$ for $s = 27$.

Then $r - \bar{r} = + 2^{-15}$. In this example, we started with "exact" digital numbers, i.e., $a - \bar{a} = 0$, $b - \bar{b} = 0$, $c - \bar{c} = 0$, $d - \bar{d} = 0$ and our digital arithmetic was "good" to 26 binary places. Yet after three such digital arithmetic operations the number computed was only "good" to 15 places!

While it is clear in principle how an a priori error analysis for a direct digital computation could be carried out - see for example: A. S. Householder, "Principles of Numerical Analysis" (1953), and J. von Neumann and H. Goldstein, "Numerical Inversion of Matrices of High Order", Bull. A.M.S. Vol. 53 (1947), it is enormously tedious to do so. Even after such an analysis is carried out the resulting bounds may still be complicated expressions, difficult to evaluate, or not very sharp.

Section 7
RANGE ARITHMETIC

7.1 INTRODUCTION

We wish to develop in this section a number system and an arithmetic dealing with closed real intervals. We call the numbers range numbers and the arithmetic range arithmetic. Numbers of this type were mentioned by P. S. Dwyer in "Linear Computations," Wiley (1951).

A modified digital range arithmetic will serve as the basis for the automatic analysis of total error in any direct digital computation.

7.2 REAL RANGE ARITHMETIC

Let R be the field of real numbers and let $\mathcal{I}$ be the set of closed bounded intervals of real numbers. The elements of $\mathcal{I}$ are called real range numbers.

We define two mappings α and β from $\mathcal{I}$ into R such that if $I \in \mathcal{I}$ and $x \in I$, $\alpha I \le x \le \beta I$. We call αI the left end point of I and βI the right end point of I. We represent I by the notation $[\alpha I, \beta I]$. Then I is the set of real numbers $\{x \mid \alpha I \le x \le \beta I\}$.

For any real numbers a, b such that $a \le b$, there is a unique I in $\mathcal{I}$ such that $\alpha I = a$ and $\beta I = b$.

For $I \in \mathcal{I}$ and $J \in \mathcal{I}$, we make the following definitions

D1. $I = J$ if and only if $\alpha I = \alpha J$ and $\beta I = \beta J$.

D2. $-I = \{-y \mid y \in I\}$

D3. $I + J = \{x + y \mid x \in I,\ y \in J\}$

D4. $I - J = I + (-J)$

D5. $IJ = \{xy \mid x \in I,\ y \in J\}$

D6. $I \cap J = \{x \mid x \in I,\ x \in J\}$

$I \cap J$ is the intersection of I and J as sets of real numbers.

D7. $I \approx J$ if and only if $I \cap J$ is non-empty. i.e.

$I \approx J$ iff $\exists x \ni x \in I,\ x \in J$.

D8. $I \not\approx J$ iff $I \cap J$ is the empty set.

D9. If $I \not\approx [0, 0]$, then $I^{-1} = \{\frac{1}{x} \mid x \in I\}$

D10. If $J \not\approx [0, 0]$, then $I/J = IJ^{-1}$

D11. $\mathcal{C} = \{I \in \mathcal{J} \mid \alpha I = \beta I\}$

$\mathcal{C}$ is the set of all real intervals containing a single real number, i.e. the set of all real intervals whose left and right end points are identical.

D12. $I \subset J$ iff $x \in I \Rightarrow x \in J$. This is the ordinary set inclusion relation for I, J as sets of real numbers.

Obviously, $\mathcal{C}$ is isomorphic to R and hence is itself a field.

From properties of real numbers and the definitions D1 to D11, the following propositions are true for I, J in $\mathcal{J}$

T1. $I + J \in \mathcal{J}$ and $I + J = J + I$

T2. $IJ \in \mathcal{J}$ and $IJ = JI$

T3. $I + [0, 0] = I$

T4. $I\ [1, 1] = I$

T5. $I \approx I$

T6. $I \approx J \Rightarrow J \approx I$

T7. $I - I \approx [0,0]$

Since $I - I = \{x - y \mid x \in I,\ y \in I\}$,
choose $x = y = \alpha I$.

T8. $I \not\approx [0,\ 0] \Rightarrow I^{-1} \in \mathcal{J}$ and $I/I \approx [1,\ 1]$.

Since $I/I = \{x/y \mid x \in I,\ y \in I\}$,

choose $x = y = \alpha I$.

T9. For $C_1 \in \mathcal{C}$ and $C_2 \in \mathcal{C}$, $C_1 \approx C_2$ iff $C_1 = C_2$

T10. $I \approx J \Rightarrow \exists C \in \mathcal{C} \ni I \approx C$ and $J \approx C$.

Since $I \approx J \Rightarrow \exists x \in R \ni x \in I,\ x \in J$,

choose $C = [x,\ x]$

T11. If $\exists\ C \in \mathcal{C} \ni I \approx C,\quad J \approx C$ then $I \approx J$

T12. $I = J$ iff $I \subset J$ and $J \subset I$.

T13. If $C \in \mathcal{C}$ and $I + J \approx C$, then

$\exists\ C_1,\ C_2 \in \mathcal{C} \ni I \approx C_1,\ J \approx C_2$

and $C = C_1 + C_2$

T14. If $C \in \mathcal{C}$ and $I\ J \approx C$, then

$\exists C_1,\ C_2 \in \mathcal{C}\quad I \approx C_1,\ J \approx C_2$ and $C = C_1\, C_2$.

For I, J, K in $\mathcal{J}$ we have

T15. $I + (J + K) = (I + J) + K$

T16. $I\ (J\ K) = (I\ J)\ K$

T17. If $K \not\approx [0,\ 0]$, $\frac{IJ}{K} = I\ (J/K)$

T18. $I\ (J + K) \subset I\ J + I\ K$

T19. For $K \not\approx [0,0]$ $\frac{I + J}{K} \subset I/K + J/K$

Notice we do not have $I \approx J$ and $J \approx K$ implying $I \approx K$, since $[0, 1] \approx [1, 2]$ and $[1, 2] \approx [2, 3]$ but $[0, 1] \not\approx [2, 3]$.

We do have

T19. If $I \approx [0, 0]$ $J \approx [0, 0]$, then $I + J \approx [0, 0]$

T20. If $I \approx [1, 1]$ $J \approx [1, 1]$, then $I\ J \approx [1, 1]$

That we do not have the distributive relation $I\,(J + K) = I\,J + I\,K$ may be seen from the example: $I = [1, 2]$ $J = [-2, 1]$ $K = [1, 2]$ we have $J + K = [-1, 3]$, and $I\,(J + K) = [-2, 6]$ while $I\,J = [-4, 2]$ and $I\,K = [1, 4]$ so that $I\,J + I\,K = [-3, 6]$

The following easily established formulas for end points establish that in the representation $I = [\alpha I, \beta I]$, the range numbers $I + J$, $I - J$, $I\,J$, and I/J are directly computable:

F1. $\alpha\,(I + J) = \alpha I + \alpha J$

$\beta\,(I + J) = \beta I + \beta J$

F2. $\alpha\,(-I) = -\beta I$

$\beta\,(-I) = -\alpha I$

F3. $\alpha\,(I - J) = \alpha I - \beta J$

$\beta\,(I - J) = \beta I - \alpha J$

F4. $\alpha\,(I\,J) = \min\,\{\alpha I\,\alpha J,\ \alpha I\,\beta J,\ \beta I\,\alpha J,\ \beta I\,\beta J\}$

$\beta\,(I\,J) = \max\,\{\alpha I\,\alpha J,\ \alpha I\,\beta J,\ \beta I\,\alpha J,\ \beta I\,\beta J\}$

The end points $\alpha\,(I\,J)$, $\beta\,(I\,J)$ can also be computed using a table of sign discriminations. Let + stand for a non-negative number and - a negative one.

F5.

sd	α I	β I	α J	β J
1	+	+	+	+
2	+	+	-	+
3	+	+	-	-
4	-	+	+	+
5	-	+	-	+
6	-	+	-	-
7	-	-	+	+
8	-	-	-	+
9	-	-	-	-

1) α (I J) = α I α J, β (I J) = β I β J

2) α (I J) = β I α J, β (I J) = β I β J

3) " = β I α J , " = α I β J

4) " = α I β J , " = β I β J

5) " = min (β I α J, α I β J), β (I J) = max (α I α J, β I β J)

6) " = β I α J, β(I J) = α I β J

7) " = α I β J, " = β I α J

8) " = α I β J, " = α I α J

9) " = β I β J , " = α I α J

In F5 , we compute only one product for each end point, except in sign discrimination 5).

F6. $\alpha (I^{-1}) = \dfrac{1}{\beta \; I}$

$\beta (I^{-1}) = \dfrac{1}{\alpha \; I}$

We perform now some illustrative computations in range arithmetic.

$$\begin{aligned}
[a,\ b] + [c,\ c] &= [a + c,\ b + c] \\
[a,\ b]\ [c,\ c] &= [c\,a,\ c\,b] \quad \text{for } c \geq 0 \\
[0,\ 1]\ [c,\ c] &= [0,\ c] \quad \text{for } c \geq 0 \\
[0,\ 1]\ [c,\ c] &= [c,\ 0] \quad \text{for } c < 0 \\
[-1,\ 1]\ [-1,\ 1] &= [-1,\ 1] \\
[a,\ b] - [a,\ b] &= [a - b,\ b - a] \\
[a,\ b] / [a,\ b] &= [a/b,\ b/a] \quad \text{for } a > 0 \\
[1,\ 1] + [1,\ 1] &= [2,\ 2] \\
[a,\ a] - [a,\ a] &= [0,\ 0] \\
[a,\ a] / [a,\ a] &= [1,\ 1] \quad \text{for } a \neq 0
\end{aligned}$$

From the uniqueness of $I + J$ and $I\,J$, we have

T21. $I = J \Rightarrow I + K = J + K$
and $I\,K = J\,K$ for any K in $\mathcal{I}$.

From F1 and D1 we obtain,

T22. If $I + J = I + K$, then $J = K$

Notice we do <u>not</u> have $I\,J = I\,K$ implying $J = K$, since $[0,\ 1]\ [1,\ 1] = [0,\ 1]\,[0,\ 1] = [0, 1]$ but $[1,\ 1] \neq [0,\ 1]$.

If A and B are in $\mathcal{I}$ and $A \approx [0,\ 0]$, the equation $A\,X + B = [0,\ 0]$ will have a solution for X in $\mathcal{I}$ if and only if A and B are in $\mathcal{C}$.

Abbreviate $[0,\ 0]$ by 0 and $[1,\ 1]$ by 1 .

T23. If $I \approx 0$, then $J \subset I + J$

T24. If $I \approx 1$, then $J \subset I\ J$

T25. If $I \subset J$, then $K + I \subset K + J$ for K in $\mathcal{I}$

T26. If $I \subset J$, then $K\ I \subset K\ J$ for K in $\mathcal{I}$.

T27. If A_k is in $\mathcal{I}$ for $k = 0, 1, 2, \ldots, n$ and X is in $\mathcal{I}$, then
$A_o + X\,(A_1 + X\,(A_2 + \ldots + X\,(A_n))\ldots) \subset A_o + A_1\,X + \ldots + A_n\,X^n$

T28. $I + J \approx 0$ iff $I \approx -J$. Choose $x \in I$ and $y \in J \ni x + y = 0$

T29. $I\ J \approx 0$ iff $I \approx 0$ or $J \approx 0$

T30. If $I \approx J$, then $K\ I \approx K\ J$ and $K + I \approx K + J$ for K in $\mathcal{J}$.

T31. If $I \subset J$ and $I \approx K$, then $J \approx K$ for K in $\mathcal{J}$.

T32. If $I \approx 0$ and $I \subset J$, then $0 \subset J$.

-since $I \approx 0$, then $0 \subset I \subset J$.

T33. If $-A^{-1}\ B \subset X$ for A, B, X in $\mathcal{J}$, then $AX + B \approx 0$
If $-A^{-1}\ B \subset X$, then $-AA^{-1}B \subset AX$, and
$B - AA^{-1}B \subset AX + B$
Since $AA^{-1} \approx 1$, then $B \subset AA^{-1}\ B$ and $B \approx AA^{-1}B$, and
$B - AA^{-1}B \approx 0$, and $AX + B \approx 0$

In fact, if $p(x) = a_o + a_1 x + \dots + a_n x^n$ and $P(X) = A_o + A_1 X + \dots + A_n X^n$, with $a_i \in A_i$, then $P(X) \approx 0$ whenever X contains a zero of $p(x)$.

We define a partial ordering relation $<$ for the set $\mathcal{J}$ by the definition

D13. $I < J$ if and only if $x \in I$ and $y \in J \Rightarrow x < y$.

D14. $I \nless J$ iff $\exists\ x \in I$, $y \in J \ni x \geq y$.

From properties of the order relations for real numbers and the definitions D1 to D13 we obtain

T34. $I < J$ iff $\beta I < \alpha J$.

T35. $I < J$ and $J < K \Rightarrow I < K$

T36. $I < J \Rightarrow I \not\approx J$.

T37. $I \not\approx J$ iff either $I < J$ or $J < I$

In the representation $[\alpha I, \beta I]$ for elements of $\mathcal{J}$, the following are directly computable.

F7. $I < J$ if $\beta I < \alpha J$ (D.13)

$I \nless J$ if $\beta I \geq \alpha J$

F8. $I \approx [x, x]$ iff $\alpha I \leq x \leq \beta I$

F9. $I \approx J$ iff $I \nless J$ and $J \nless I$

F10. $I \not\approx J$ iff $I < J$ or $J < I$

F11. $I \subset J$ iff $\alpha J \leq \alpha I \leq \beta I \leq \beta J$

F12. $I \in \mathcal{C}$ iff $\alpha I = \beta I$

If $a_o + a_1 x + \cdots + a_n x^n$ is a real polynomial with $a_i \in A_i$ and $x \in X$, then

$p(x) = a_o + a_1 x + \cdots + a_n x^n \in (A_o + A_1 X + \cdots + A_n X^n) = P_1(X)$

In fact, since

$p(x) = a_o + a_1 x + \cdots a_n x^n = a_o + x (a_1 + \cdots x (a_n)) \cdots),$

then $p(x) \in A_o + X (A_1 + X (A_2 + ,,, + X (A_n)) \cdots) = P_2 (X)$

In T27 above we stated the interesting theorem $P_2(X) \subset P_1(X)$, (T27. follows easily from T18, T25, and T26).

In other words, if $p(x)$ has coefficients such that $\alpha A_i \leq a_i \leq \beta A_i$, then $p(x) \in P_2(X) \subset P_1(X)$ for x such that $\alpha X \leq x \leq \beta X$.

The range polynomial $P_2(X)$ may not be minimal for $p(x)$, however, as can be seen from the following example.

Take $p(x) = x - x^2$; then $P_2 (X) = X(1 - X)$ and for x such that $0 \leq x \leq 1$, i.e. for $X = [0, 1]$, $[0, 1] \left([1, 1] - [0, 1] \right) = [0, 1][0, 1] = [0, 1]$

Therefore $p(x) \in [0, 1]$. Since $p(x) = x(1 - x) = 1/4 - (x - 1/2)^2$,

then also $p(x) \in \left([1/4,\ 1/4] - (X - [1/2,\ 1/2])^2\right)$

so that $p(x) \in \left([1/4,\ 1/4] - ([0,\ 1] - [1/2,\ 1/2])^2\right)$

Since

$$\begin{aligned} & [1/4,\ 1/4] - \left([0,\ 1] - [1/2,\ 1/2]\right)^2 \\ &= [1/4,\ 1/4] - [-1/2,\ 1/2]^2 \\ &= [1/4,\ 1/4] - [-1/4,\ 1/4] \\ &= [1/4,\ 1/4] + [-1/4,\ 1/4] \\ &= [0\ ,\ 1/2] \end{aligned}$$

then $p(x) \in [0,\ 1/2] \subset [0,\ 1]$.

Of course, the actual range of values $p(x) = x(1 - x)$ for $0 \le x \le 1$, is the interval $[0,\ 1/4]$.

But what is the range of values for $p(x) = 2 - 9x - 6x^2 - 5x^4 - 7x^5 + 5x^6 + 2x^7 + 2x^8 - x^9 + 8x^{10}$
when $0 \le x \le 1$?

Putting $A_0 = [2,\ 2]$, $A_1 = [-9,\ -9]$, . . . , $A_{10} = [8,\ 8]$ and $X = [0,\ 1]$, we find $P_2(X) = [-25,\ 2]$ and $P_1(X) = [-25,\ 19]$ for the range polynomials of type P_1 and P_2 described above. Therefore $-25 \le p(x) \le 2$.

7.3 DIGITAL RANGE ARITHMETIC

If the end points of range numbers are restricted to lie in a set S of digitally representable numbers, the resulting range numbers are called digital range numbers. Thus if αI, $\beta I \in S_3$, the digital range number $I = [\alpha I,\ \beta I]$ is the set of real numbers x such that $\alpha I \le x \le \beta I$. With a pair of numbers a, b in S_3 , representable in R_3 by words w_1 and w_2 in W_s , the interval $a \le x \le b$ of real numbers x is represented by $[a,\ b]$.

The following digital range operations are defined for I, J in $\mathcal{J}$ such that αI, βI, αJ, βJ are in S_3 and in the domain of the appropriate digital arithmetic operation.

D15. $I \oplus J = [\alpha I \oplus \alpha J, \beta I \oplus \beta J]$
restricted to I, J such that $|\alpha I + \alpha J| < 1$
and $|\beta I + \beta J| < 1$ as we recall from Section 4.

D16. $I \ominus J = [\alpha I \ominus \beta J, \beta I \ominus \alpha J]$

D17. $I \otimes J = [\alpha (I \otimes J), \beta (I \otimes J)]$

where

$$\bar{\alpha}(I \otimes J) = \min\{\alpha I \otimes \alpha J, \alpha I \otimes \beta J, \beta I \otimes \alpha J, \beta I \otimes \beta J\}$$
$$\bar{\beta}(I \otimes J) = \max\{\alpha I \otimes \alpha J, \alpha I \otimes \beta J, \beta I \otimes \alpha J, \beta I \otimes \beta J\}$$

and

$$\alpha(I \otimes J) = \begin{cases} \bar{\alpha}(I \otimes J) & \text{if } \sigma\bar{\alpha} > 0 \\ 0 & \text{if } \sigma\bar{\alpha} = 0 \\ \bar{\alpha}(I \otimes J) \ominus 2^{-s+1} & \text{if } \sigma\bar{\alpha} < 0 \end{cases}$$

By $\sigma\bar{\alpha}$ we mean (sgn xy) where x, y are the factors in the term which produces min for $\bar{\alpha}$. Similarly for $\sigma\bar{\beta}$.

and
$$\beta(I \otimes J) = \begin{cases} \bar{\beta}(I \otimes J) \oplus 2^{-s+1} & \text{if } \sigma\bar{\beta} > 0 \\ 0 & \text{if } \sigma\bar{\beta} = 0 \\ \bar{\beta}(I \otimes J) & \text{if } \sigma\bar{\beta} < 0 \end{cases}$$

The computation of all four products in the formulas for $\bar{\alpha}$ and $\bar{\beta}$ may be avoided in all but one of nine cases by using the table of sign discriminations as given above in F5.

D18. For $J \not\ni 0$,
$I \oslash J = [\alpha(I \oslash J), \beta(I \oslash J)]$

where

$$\bar{\alpha}(I \oslash J) = \min\{\alpha I \oslash \alpha J, \alpha I \oslash \beta J, \beta I \oslash \alpha J, \beta I \oslash \beta J\}$$
$$\bar{\beta}(I \oslash J) = \max\{\alpha I \oslash \alpha J, \alpha I \oslash \beta J, \beta I \oslash \alpha J, \beta I \oslash \beta J\}$$

and

$$\alpha(I \oslash J) = \begin{cases} \bar{\alpha}(I \oslash J) & \text{if } \sigma\bar{\alpha} > 0 \\ 0 & \text{if } \sigma\bar{\alpha} = 0 \\ \bar{\alpha}(I \oslash J) \ominus 2^{-s+1} & \text{if } \sigma\bar{\alpha} < 0 \end{cases}$$

and

$$\beta(I \oslash J) = \begin{cases} \bar{\beta}(I \oslash J) \oplus 2^{-s+1} & \text{if } \sigma\bar{\beta} > 0 \\ 0 & \text{if } \sigma\bar{\beta} = 0 \\ \bar{\beta}(I \oslash J) & \text{if } \sigma\bar{\beta} < 0 \end{cases}$$

Again, the number of quotients to be computed may be reduced by a table of sign discriminations.

D19. $I < J$ if $\alpha\ J > \beta\ I$
$I \nless J$ if $\alpha\ J \leq \beta\ I$

The digital range operations $I \oplus J$, $I \ominus J$, $I \otimes J$, $I \oplus J$, $I < J$ can clearly be programmed as digital computations in machine language, (see Section 5.3).

The <u>range step</u>, "$K = I \circ J$", for $\circ \in \{+, -, \times, \div\}$ and the corresponding digital range computation automatically bound the real step " $z = x \circ y$" and the corresponding digital computation. In fact, if $x \in I$ and $y \in J$, then $z \in K$ and K is contained in the digital version of $I \circ J$.

The operation $I \circledleq J$ should be programmed so that the following range step is represented:

"if $I \circledleq J$, do r; if $J < I$, do t; otherwise stop" where r and t are range steps.

Note that, by D8 and D13, exactly one of the following is true for I, J in $\mathcal{J}$:

$$I < J$$
$$J < I$$
$$I \approx J$$

Furthermore, by D19, if $I \circledleq J$, then $x \circledleq y$ for $x \in I$, $y \in J$.

Therefore, the range step just described chooses r as the next step if $x \circledleq y$ for every $x \in I$, $y \in J$, and chooses t as the next step if $x > y$ for every $x \in I$, $y \in J$. It halts the computation in case neither $x \leq y$, nor $x > y$ is true for every $x \in I$, $y \in J$.

The "round-off factor" 2^{-s+1} was used to extend the result of a digital range arithmetic operation only where required in order to guarantee that

$$I + J \subset I \oplus J$$
$$I - J \subset I \ominus J$$
$$I\,J \subset I \otimes J$$
$$I/J \subset I \oslash J$$

From Section 4.3, in particular from the definitions of $\oplus$, $\ominus$, $\otimes$, $\oslash$ for pairs of numbers x, y in S_3, it follows for example, that if $xy > 0$, then $x \otimes y = (\operatorname{sgn} xy) \left[|xy|\, 2^{s-1} \right] 2^{-s+1}$ and $x \otimes y \leq xy \leq (x \otimes y) \oplus 2^{-s+1}$

Notice that if $x = y = 1 - 2^{-s+1}$, then $x \otimes y = 1 - 2^{-s+2}$ and the operation $(x \otimes y) \oplus 2^{-s+1}$ is valid; in fact, $(1 - 2^{-s+2}) \oplus 2^{-s+1} = 1 - 2^{-s+1}$. On the other hand, if $x\,y < 0$, then $(x \otimes y) \ominus 2^{-s+1} \leq xy \leq x \otimes y$.

In any case, $\alpha\,(I \otimes J) \leq \alpha\,(I\,J) \leq \beta\,(I\,J) \leq \beta\,(I \otimes J)$

Similarly,

$$\alpha\,(I \oslash J) \leq \alpha\,(I/J) \leq \beta\,(I/J) \leq \beta\,(I \oslash J)\,.$$

As an example of digital range arithmetic consider $\bar{R} = \big((B \otimes C) \oslash D\big) \oplus A$, $D \not\ni 0$, for A, B, C, D in $\mathcal{J}$ with end points αA, βA, ..., αD, βD in S_3.

If $a \in A$, $b \in B$, $c \in C$, $d \in D$, then $r = a + bc/d \in \bar{R}$

In Section 6, an example was considered with $a = -2^{-15}$, $b = c = 2^{-14}$, $d = 2^{-13}$, and $s = 27$. Choose $A = \left[-2^{-15}, -2^{-15}\right]$, $B = C = \left[2^{-14}, 2^{-14}\right]$, $D = \left[2^{-13}, 2^{-13}\right]$, then $B \otimes C = \left[2^{-14}, 2^{-14}\right] \otimes \left[2^{-14}, 2^{-14}\right] = \left[0, 2^{-26}\right]$

$$(B \otimes C) \oslash D = \left[0, 2^{-26}\right] \oslash \left[2^{-13}, 2^{-13}\right] = \left[0, 2^{-13} + 2^{-26}\right]$$

$$\bar{R} = \big((B \otimes C) \oslash D\big) \oplus A = \left[0, 2^{-13} + 2^{-26}\right] \oplus \left[-2^{-15}, -2^{-15}\right]$$

$$\bar{R} = \left[-2^{-15}, 2^{-13} - 2^{-15} + 2^{-26}\right]$$

Then $r = -2^{-15} + \dfrac{2^{-14} \cdot 2^{-14}}{2^{-13}} = 0 \in R$ as promised, i.e.

$-2^{-15} \leq r \leq 2^{-13} \quad -2^{-15} + 2^{-26}$.

Notice also that for $\bar{r} = \left((b \otimes c) \oslash d\right) \oplus a$, the a priori bound developed in Section 6 gives

$$|r - \bar{r}| < 2^{-s+1} \left(2 + \frac{4}{|\bar{d}| - 2^{-s+1}}\right)$$

or

$$|r - \bar{r}| < 2^{-26} \left(2 + \frac{4}{2^{-13} - 2^{-26}}\right)$$

Since

$$\bar{r} = \left((2^{-14} \otimes 2^{-14}) \oslash 2^{-13}\right) \oplus (-2^{-15}) = -2^{-15}$$

then the a priori bound gives

$$\frac{-2^{-11}}{1 - 2^{-13}} - 2^{-15} - 2^{-25} < r < \frac{2^{-11}}{1 - 2^{-13}} - 2^{-15} + 2^{-25}$$

Digital range numbers can also be defined with end points representable in R_4, R_6, etc. and appropriate definitions can be given for

$$I \oplus_2 J, \quad I \ominus_2 J, \quad I \otimes_2 J, \quad I \oslash_2 J$$

and for

$$I \mathbin{\hat{+}} J, \quad I \mathbin{\hat{-}} J, \quad I \mathbin{\hat{\times}} J, \quad I \mathbin{\hat{\div}} J, \qquad \text{etc.}$$

<u>A computer program written for a direct digital range computation will produce as results sets of real intervals with digital numbers as end points. In these intervals will lie the exact results of the corresponding real arithmetic computations.</u>

A computer program has been written for the 1103AF (with built-in floating point) computer at Lockheed Missiles and Space Division for performing direct digital computations in range arithmetic. The program is called RANGE and it contains operations providing for digital range computations in two digital representations: single precision fixed point fractional numbers, as in our representation R_3 , and single precision floating point numbers, as in our representation R_4.

Computations using RANGE have included: the evaluation of polynomials of high degree with coefficients defined by recursion formulas, the inversion of matrices and the solution of systems of linear algebraic equations by direct methods, and the solution of systems of rational difference equations. In each case the machine solution obtained consisted of intervals containing the exact infinite precision solution of the corresponding real computation.

Section 8
ITERATIVE COMPUTATIONS

A particular class of indirect computations is the set of iterative computations.

If $c_1, c_2, c_3, \ldots$ is a sequence of directly computable real numbers, then the computation of the sequence is <u>iterative</u> if c_{i+1} is directly computable from the set $\{x_1, x_2, \ldots x_n, c_i\}$

and $$C_1 = f(x_1 x_2 \ldots, x_n Co)$$

and $$C_{i+1} = f(x_1, x_2 \ldots, x_n C_i), \quad i \geq 1$$

for some function f which is rational in all its arguments.

The evaluation of the function f is a direct computation consisting of a finite number of arithmetic steps

1. $r_1 = u_1 \text{ o } v_1$
2. $r_2 = u_2 \text{ o } v_2$

- - - - - - -

p. $f(x_1 x_2, \ldots, x_n, c_i) = r_p = u_p \text{ o } v_p$

where in the k^{th} step o is one of the arithmetic operations $+$, $-$, $\times$, $\div$ and u_k, v_k is a pair of numbers from the set $\{0, 1, x_1, x_2, \ldots x_n, c_i, r_1, r_2, \ldots r_{k-1}\}$ for $k = 1, 2, \ldots, p$.

Suppose that $X_1, X_2, \ldots X_n, C_o$ are range numbers in $\mathcal{J}$ such that

$x_k \in X_k$, $c_o \in C_o$

and

1. $R_1 = U_1 \odot V_1$

2. $R_2 = U_2 \odot V_2$

- - - - - - - -

p. $F(X_1 , X_2 , \ldots , X_n , C_i) = R_p = U_p \odot V_p$

defines a range computation such that $\odot$ in the k^{th} step is the range arithmetic operation corresponding to the real arithmetic operation o in the k^{th} step of the real computation defining r_k . Assume that if $\odot$ is a division at the k^{th} step, then $V_k \not\ni 0$, so that $U_k \odot V_k = U_{k/V_k}$ is defined.

We have $r_k \in R_k$ for $k = 1, 2, \ldots, p$

In particular, if

$$C_{i+1} = F(X_1, X_2, \ldots, X_n, C_i),$$

then $r_p \in R_p$ or $c_{i+1} \in C_{i+1}$

Since C_{i+1} is a bounded interval, $f(x_1 x_2, \ldots x_n, c)$ is continuous in c for $c \in C_i$. A bounded rational function is continuous.

If $C_{i_o+1} \subset C_{i_o}$ for some i_o then for each set $\{x_1, x_2, \ldots, x_n\}$ such that $x_k \in X_k$, $f(x_1, x_2, \ldots, x_n, c)$ has a fixed point $\bar{c}$ such that $\bar{c} \in C_{i_o+1}$ and $\bar{c} = f(x_1, x_2, \ldots, x_n, \bar{c})$.

Since $I \subset J$ (by T25 and T26 in Section 7) implies that $I + K \subset J + K$, $I K \subset J K$, $I - K \subset J - K$, $I/K \subset J/K$ for $K \not\ni 0$, for K in $\mathcal{J}$, then evidently

$$C_{i_o} \supset C_{i_o+1} \supset C_{i_o+2} \cdots$$

forms a nested sequence of range numbers with

$$\bar{c} \in C_{i_o+j} \qquad \text{for} \qquad j = 0, 1, 2, \ldots$$

As an illustration, consider the iterative computation defined by $r_{i+1} = \dfrac{3 + r_i^2}{4}$ for $i = 0, 1, 2, \ldots$ with r_o such that $0 \le r_o \le 2$. Choose $R_o = [0, 2]$ and define

$$R_{i+1} = ([3, 3] + R_i^2) / [4, 4] \text{ for } i = 0, 1, 2, \ldots$$

Clearly,

$$R_1 = \frac{[3, 3] + [0, 2]^2}{[4, 4]} = [3/4 \, , \, 7/4] \subset \bar{R}_o \, .$$

In fact, $R_o, R_1, R_2, \ldots$ is a nested sequence of intervals converging to the interval $[1, 1]$ containing the single real number 1 which is indeed a fixed point of

$$f(r) = \frac{3 + r^2}{4}$$