A SMALL GROUP OF RESEARCH PROJECTS

IN

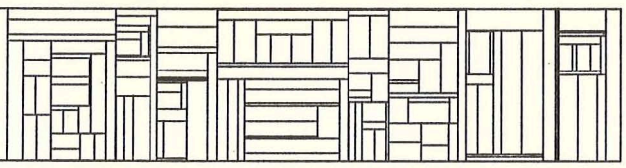
MACHINE DESIGN FOR SCIENTIFIC COMPUTATION

by

Bruce D. Shriver

DAIMI PB-14
June 1973

Institute of Mathematics University of Aarhus
DEPARTMENT OF COMPUTER SCIENCE
Ny Munkegade - 8000 Aarhus C - Denmark
Phone 06-12 83 55



A Small Group of Research Projects in Machine Design for Scientific Computation*

by

Bruce D. Shriver

1.0 Introduction

This paper outlines several research projects in an area which can loosely be termed "machine design for scientific computation". There are several sources of motivation for undertaking the line of research discussed herein:

1) important and fundamental questions in numerical analysis are raised within the framework of investigaring machine designs for scientific computation,

2) the extremely rapid growth of capability in hardware technology both allows and compells us to re-examine the utility of present day machines for scientific computation, and

3) there exists a growing need to establish standards within this area.

The influence of the first source will be seen when reading the descriptions of the research topics and their related projects. The third source of motivation, the standards problem, will be delt with in Section 3 . Let us here offer a few comments with respect to the second source of motivation, increased hardware capability through advanced technology. Developments in large scale integration and solid state technology have led both Foster [1] and Auerbach [2] to prognosticate the existence of mass produced "computers-on-a-chip" within ten years. If such computers are to be the tool of the numerical analysts and scientific and industrial users of numerical algorithms, then certainly these machines should have the computational properties required for this purpose. Consider a recent related statement by Kahan [3]:

"There is a natural analogy between illness and numerical inaccuracy.

*) This work is being supported by the Danish Research Council, Grant No. 511-1546.

Germs and rounding errors are small, numerous, and best combatted by sanitary precautions, which, alas, are all to frequently neglected, not so much because of their intrinsic difficulty as because of indifference or ignorance." We can extend this medical analogy slightly by enumerating a few illnesses. We have anomalies in floating point units, undiagnosed underflows and overflows, unuseful treatment of indeterminate values which some language processors inflict upon us, and an incomplete knowledge of the finite floating point number system which is used as a basis for a great deal of scientific computation. Should we possibly not view mass-produced "computers-on-a-chip" as "carriers" of a potential epidemic and initiate preventative measures to forestall any harmful effects?

Even if such mass-produced machines are not to be the normal tool of the numerical analysts, they still should go through a re-examination of the usefulness of the machines they are using unless they are (a) satisfied with machines as they currently are or (b) feel they can do little to influence the design of machines. We support Cody's contention [4] that we must "make a lucid statement of needs and desires with supporting evidence" if we are to effect change. The research outlined here may take many man-years of effort, however, the results of the various proposed projects should, hopefully, assist in making such a statement.

It should be the purpose of such research as this to provide the numerical analyst with a more powerful, versatile, and efficient computational facility than currently available, as well as providing a theoretical basis for the use of that facility. In this context, the proposed research is concerned with identifying machine organizations, primitive data types, and basic machine operations useful in scientific computation and which can be supported by existing software techniques, as well as developing a related mathematical theory. It is hoped to gain insight into the impact of scientific computational requirements on overall computer systems organization and implementation, and vise-versa.

The area termed "machine design for scientific computation" encompasses efforts in the design of arithmetic units for standard and non-standard arithmetics, enhancement of matrix manipulation facilities, provision for the efficient evaluation of the elementary

functions and of special frequency user routines (mathematical software), and the provision of related non-numerical techniques. In this short note we will only concern ourselves with projects in arithmetic , extending those presented in Shriver [5].

2.0 Studies in Machine Arithmetic

Numerical analysis is, by definition, concerned with the arithmetization of mathematical problems. The arithmetic unit of a computer should be designed in such a way as to facilitate the work of the user. That this is normally not the case has been documentated a number of times in the literature; see, for example, Cody [4, 7], Gregory [8], Kahan [3, 9, 10, 11], Levitan [12], and Lawson [13]. Motivated by this fact and also because much numerical experimentation requires a versatile arithmetic unit, we first suggest investigations concerned with the theory, design, and construction of arithmetic units. The mnemonic SIMA will be used to denote those projects concerned with conducting theoretical and experimental Studies in Machine Arithmetic.

The variety of arithmetics to be considered should at least consist of the two standard arithmetics, fixed point and floating point, and the following non-standard arithmetics, rational, significance-indicating (end point interval, triplex interval, unnormalized, and significance index), extended fixed point, extended floating point, residue, complex, and polunomial. Each of these arithmetics is characterized by a set of operations (represented by symbols such as +, /, etc.), a set of operands (objects upon which the operations are defined), and a set of derived properties associated with the operations and operands (such as associativity, distributivity, etc.). The operations and operands will frequently be named according to a particular arithmetic we are concerned with. Thus we may write, for example, "a floating point add". The type of the operation is floating point. We may also write, for example, "a rational number". The type of the operand is rational. This merely allows us to write short hand expressions such as, "the operation '+' should be defined for operands of the same type".

Operands have abstract, finite-machine, and implementation representations. In order to define these terms, let us introduce the following concepts.

Definition: A representation tuple, $r^n = (r_1, \ldots, r_n)$, is an n-tuple whose elements, r_i, are members of given sets, S_i, called the representation sets, i.e., $r_i \in S_i$, $i = 1, \ldots, n$.

Definition: A representation function, $M^n = M(y_1, \ldots, y_n)$, is a function whose domain, D, is a set of representation tuples and whose range is a set H, i.e., M^n associates with the object $y^n \in D$ a single object $M(r^n) \in H$.

Let t be an element from a set of objects T.

Definition: A representation, $R^n = (M^n, r^n)$, of $t \in T$ is defined to be a representation function, M^n, together with a representation tuple, r^n, such that M^n associates with $r^n \in D = S_1 \times S_2 \times \ldots \times S_n$ a single object $M(r^n) \in H$ where $H = T$.

Let x be an element from a set of objects X. Let $\mathfrak{R}$ be a relation which exists between elements from the set X and elements from the set T.

Definition: A $\mathfrak{R}$-basis abstract representation, $AR_{\mathfrak{R}}$, of x is a representation $R^n = (M^n, r^n)$ of some t such that

(i) $x \mathfrak{R} t$ holds,

(ii) $H = T = X$, and

(iii) the representation sets, S_i, $i = 1, \ldots, n$, may be infinite.

Definition: A $\mathfrak{R}$-basis finite-machine representation, $FR_{\mathfrak{R}}$, of x is a representation $R^q = (M^q, r^q)$ of some t such that

(i) $x \mathfrak{R} t$ holds,

(ii) $H = T \subset X$, and

(iii) the representation sets S_i, $i = 1, \ldots, q$, are finite sets.

The restriction of the (possibly) infinite representation sets of an $AR_{\mathfrak{R}}$ to the finite representation sets of a $FR_{\mathfrak{R}}$, and the restriction of the set T to be a subset of X reflect the finiteness of the machine.

Definition: A $\Re$-basis implementation representation, $IR_{\Re}$, of x is a representation $R^p = (M^p, r^p)$ of some t such that

(i) $x \Re t$ holds,

(ii) $H = T \subset X$, and

(iii) the representation sets S_i, $i = 1, \ldots, p$, are isomorphic to the set $\{0, 1, \ldots, \beta-1\}$ where β is the base of the machine upon which the representation is implemented.

The restriction of the representation sets reflects the binding of the representation to a particular machine.

Let us give an example of the above. Suppose we wish to represent $x \in S_b^n$ where

S_b^n: {set of n significant-digit, base b, real numbers}.

Let $X = T = S_b^n$, J be the integers, N be the positive integers, and $\mathbb{R}$ the reals.

(1) An abstract representation of x based on the relation of equality, $AR_=$, could be given by the following:

$$AR_= : (M^3, r^3)$$

where the representation function is

$$M^3 = M(f, b, e) = fb^e.$$

Here the representation parameters are f, b, and e. f is called the coefficient, e the exponent, and b the base. The elements of the representation tuple $r^3 = (r_1, r_2, r_3)$ have the following representation sets,

$$r_1 \in S_1 : \{k \mid k \in J, |k| < b^n\},$$

$$r_2 \in S_2 = N \setminus \{1\}, \text{ and}$$

$$r_3 \in S_3 = J.$$

The domain and range of M are $D = S_1 \times S_2 \times S_3$ and $H = S_b^n \subset \mathbb{R}$.

(2) Suppose we wish to model this abstract representation on a class of word oriented finite-machine whose words are w units wide. Furthermore, j units of w represent the value of the coefficient and k units of w represent the exponent. Let $T = S^n_{b,w}$: {the set of n significant digit, base b, real numbers representable in 1 word}. A finite-machine representation of $x \in S^n_b$ based on the relation $|x-t| \leq \delta$, $(0 < \delta < 1)$, could be:

$$FR_{|x-t| \leq \delta} : (M^3, r^3)$$

where the representation function is

$M^3 = M(f_m, b_m, e_m) = f_m b_m^{e_m}$ where $0 \leq |e_m| \leq b_m^k$, $b_m^{j-1} \leq b_m^j |f_m| \leq b_m^j$, and $j + k = w$. Here k is used to specify the finite range of the exponent e_m and j is used to specify the finite precision of the coefficient, f_m. The constraint on f_m also indicates that f_m is normalized. The elements of the representation tuple $r^3 = (r_1, r_2, r_3)$ elements of the following representation sets,

$$r_1 \in S_1 \subset [-1, {}^{-1}/_{b_m}] \cup [{}^{1}/_{b_m}, 1],$$
$$r_2 \in S_2 = \{2, 3, 8, 10, 16\}, \text{ and}$$
$$r_3 \in S_3 = \{-b_m^k, -b_m^k+1, \ldots, 0, 1, \ldots, b_m^k\}.$$

The domain and range of M^3 are $S_1 \times S_2 \times S_3$ and $H = S^n_{b,w} \subset \mathbb{R}$. Note the class of machines considered are given by S_2 as binary, ternary, octal, decimal, and hexidecimal.

(3) If we wish to implement this FR on a binary computer with a fixed j and k. A possible interpretation based on the relation $|x-t| \leq \delta$, $(0 < \delta < 1)$, where $T = S^n_{b,w}$ could be:

$$IR_{|x-t| \leq \delta} : (M^{j+k}, r^{j+k})$$

where the representation function is

$$M^{j+k} = M(b_0, \ldots, b_{j-1}, c_0, \ldots, c_{k-1}) =$$

$$\left[(-1)^{b_0} \sum_{i=1}^{j-1} b_i 2^{i-j}\right] 2^{((-1)^{c_0} \sum_{i=1}^{k-1} c_i 2^i)}$$

This representation function corresponds to a signed magnitude interpretation of both the fraction and the exponent. The elements of the representation tuple $r^{j+k} = (r_1, \ldots r_{j+k})$ are members of the following representation sets:

$$r_i \in S_i = \{0,1\}, \quad i = 1, \ldots, j+k$$

The domain and range of M^{j+k} are $D = \{0,1\}^{j+k}$ and $H = S_{b,w}^{n} \subset \mathbb{R}$.

This example points out where some of the problem areas in the representation of the reals by a finite precision, finite range number system implemented on a computer lie.

We have an incomplete knowledge of the properties of various representations and of the mappings which are normally defined to exist between representations, e.g., a rounding or trunctation mapping which establishes a correspondence between elements of an AR and an IR. There has been recent theoretical (Matula [14, 15, 16, 17, 18, 19] and Garner [20])and statistical investigations concerning various properties of given representations and associated mappings (Marosa and Matula [21], Tienari [22], Kuki and Cody [23], Cody [24], Urabe [25], Hull and Swenson [26], and Ashenhurst [27]). We should study alternate representations to the positional notation normally used (such as logarithmic, condinued fraction, continued produce, etc.) and consider the realization of arithmetic units capable of handling such representations.

Definition: An arithmetic unit, AU, of type A is a realization of the operations defined in arithmetic A upon operands of type A in a fashion consistent with the properties associated with arithmetic A.

This realization may be in hardware, software, or firmware. Having given these introductory remarks, we are now prepared to introduce several projects in the SIMA area. These projects fall into three broad categories:

(1) the design of arithmetic units for the execution of various standard and nonstandard arithmetics,

(2) the development of a theory of the mathematical properties of various abstract, finite-machine, and implementation representations of arithmetic operands, and

(3) the development of a theory of error analysis and functional approximation in various non-standard arithmetics.

There are several research efforts recently or presently underway in these areas; papers 3 and 5-27 already cited,as well as the presentations at the recent symposium on Computer Arithmetic at the University of Maryland [20] should be mentioned along with a host of others. Our projects should complement and extend some of these efforts. The areas above are not disjoint and in the project descriptions given below they will not be separated out. We begin by describing a project to design a basic fixed and floating point unit and then extending it to allow operations in several non-standard arithmetics.

SIMA. 1 Basic Arithmetic Unit

Propose and evaluate various schemes for the realization of the following two arithmetics in 1 and 2 word precision.

a) fixed point arithmetic, and
b) normalized floating point arithmetic.

One of the objectives of this project is to gain historical perspective and insight into the design and construction of arithmetic units (technology), the machine and implementation representations of numbers on which they operate, and the desirable features in an arithmetic unit or-

ganization for the efficient execution of the above arithmetics. It is also hoped to gain an understanding of the impact of arithmetic unit design on computer organization and vise-versa. The following should be done within the framework of this project:

a) An abstract, finite-machine, and implementation representation should be chosen for fixed point numbers (F_p), and normalized floating point numbers (N_f) in both 1 and 2 word precision. The rational as to the choice of the particular representations should be given.

b) Several mathematically consistent, useful, and flexible schemes for the treatment of underflow, overflow, indeterminate forms, rounding strategies, catastrophic significance loss, and other associated phenomena should be defined and analyzed. A technique whereby the user of a particular airhtmetic may optionally choose which schemes he wishes to employ should be proposed. (That is, the schemes are not bound with the design of the arithmetic unit, but binding is delayed until execution time.) A default scheme should be defined which can be employed by the user who does not wish to worry about these phenomena. See, for example, Kahan [10], Cody [4], and Neely [6].

c) Conversion algorithms should be constructed so that numbers of one type can be converted to numbers of another type when such conversion is meaningful. A mechanism should be developed whereby any of the phenomena in (b) above (e.g., overflow) which can occur or have special meaning during such conversion can be recognized by the user.

d) The arithmetic unit should be capable of executing the following operations when the operands are of the same type (see, for example, Knuth [30]).

Operations	Comment
+, −, *, /	add, subtract, multiply divide; r and n
Σ (+, −, *, /) dp	double precision accumulation of sums, differences, products, and quotients; r and n
signum(x)	signum(x)=1 if x>0, 0 if x=0, −1 if x<0
x^i	x to the integer power i; r and n
$\lfloor x \rfloor$	floor of x
$\lceil x \rceil$	ceiling of x
round	rounding operation

where r means rounded
n means unrounded.

The arithmetic unit should be capable of determining the truth value of the follwoing relations (see, for example, van Wijngaarden [29] and Knuth [30]).

12 11

Relation	Comment
$<_\epsilon$	approximately less than
$\approx_\epsilon$	approximately equal to
$>_\epsilon$	approximately greater than

e) The above operations and tests should be defined when the operands are not of the same type.

f) Choose a machine on which to simulate the proposed arithmetic unit. Establish techniques and criteria to evaluate the performance of arithmetic units. Implement the arithmetic unit and evaluate it, identifying what aspects of the system it was realized on are not particularly useful or even hamper the implementation, as well as what additional features might enhance it.

g) A logical design of a family of arithmetic units having variou cost/performance ratios which would realize these arithmeti in hardware should be proposed. Any enhancements of or de- letions to the requirements given here (e.g., additional prim tive operations) should be clearly stated and justified. Propo a virtual machine instruction set for the use of these units.

SIMA. 2 Extended Basic Arithmetic Unit

Extend Project SIMA. 1 so that all of the operations and test defined for operands of

a) up to a specified constant length > 2 words,
b) arbitrary length.

Thus we could have, for example, extended/arbitrary precision fi point arithmetic and extended/arbitrary range and precision floati point arithmetic.

SIMA. 3 Interval Arithmetic Unit

Propose and evaluate various schemes for the realization of

a) end point interval arithmetic
b) triplex interval arithmetic.

This project should meet the same requirements set forth in the Ba Arithmetic Unit Project (SIMA. 1) where the table of operations an tests has been appropriately modified for this arithmetic (see, Moo [31] and Apostollottos et al. [32]). The components of the machine implementation representations chosen for interval numbers (I_f) ca be assumed to be the same as those chosen for normalized floating point numbers in SIMA. 1. (A proposal for a unified number represe tation giving normalized, unnormalized and interval arithmetic is g by Kornerup in [33] and might be considered as a possible alternat

SIMA. 4 Extended Interval Arithmetic Unit

Extend Project SIMA. 3 in the spirit of the Extended Basic A
metic Unit Project, SIMA. 2.

SIMA. 5 Integer and Rational Arithmetic Unit

Propose and evaluate several schemes for the realization of
teger and rational arithmetic where the range of the numbers is

a) limited to 1 and 2 word precision
b) extended to n> 2 words, n a constant.

The project should meet the same requirements set forth in the Ba
Arithmetic Unit Project (SIMA. 1) where the table of operations a
tests has been appropriately modified.

SIMA. 6 Complex Arithmetic Unit

Propose and evaluate several schemes for the realization of
plex arithmetic. The components of the machine and implementatio
representations for complex numbers (N_c) can be thought of being
presented as floating point normalized numbers in fixed, extended
arbitrary length. Representations are not to be restricted to this
terpretation. Alternate representations, using the bases $\sqrt{2}i$ and
for example, should be considered, (see Knuth [30]). This projec
should meet the same requirements set forth in the Basic Arithme
Unit Project (SIMA. 1) where the table of operations and tests ha
appropriately modified.

SIMA. 7 Unnormalized Arithmetic Unit

Propose and evaluate various schemes for the realization of
point significance of

a) the Ashenhurst/Metropolis type [34],
b) the Gray/Harrison type [35].

as an extension to the Basic Arithmetic Unit (SIMA. 1). The proje should meet the same requirements as set forth in SIMA. 1 where table of operations and tests remains the same. An Extended Unno malized Arithmetic Unit along the lines of the Extended Basic Ari metic Unit should also be designed.

It is certainly within the framework of these projects to encou investigations of alternative number representations and algorithm for the execution of the primitive arithmetic operations and tests. logarithmic representation of Marasa and Matula [21], the combin representation of Kornerup [33], the multiplication and division a gorithms of Mitchell [36], the Cordic (coordinate rotation) repres tation of Volder [40], and the negative base algorithms of Krishn et al. [41] are examples of alternate representations and algorith The following project is typical of a project specification particul one of these efforts and can actually be considered part of project SIMA. 1.a.

SIMA. 8 "Reduced Significance" Arithmetic Unit

The following references, Mitchell [36], Combet, et al. [37] Hall, et al. [38], and Marino [39] deal with miltiplication and div in what may be termed "reduced significance" arithmetic using bin logarithms. Propose and evaluate various schemes for the realiza of this arithmetic. This project should meet the requirements set in the Basic Arithmetic Unit Project (SIMA. 1) where the table of erators and tests has been modified accordingly.

We can now consider the following project.

SIMA. 9 Combined Arithmetic Unit

Propose and evaluate several schemes for the realization of t arithmetics of projects SIMA. 1, SIMA. 3, SIMA. 5, SIMA. 6, an SIMA. 7 (or any subset of these containing at least two elements) in the same arithmetic unit. The operands may be of fixed, extend variable length. The operators should be polymorphic: for example should be only one '+' operator even though the operands may be r senting N., I., etc. numbers.

After having completed the Combined Arithmetic Unit Project, SIMA. 9, one has an arithmetic unit which is capable of executing operations and performing tests in the standard arithmetics (fixed and floating point) as well as a variety of non-standard arithmetics. However, major problem areas still remain before the user has appropriate access to such a unit. How can one allow for the use of such a variety of arithmetics in standard high level programming languages such as Algol or Fortran? How can one "switch" from one arithmetic environment to another? How will the user bind his interpretation of several features of the arithmetic unit which have not been frozen into the design, for example, the treatment of underflow and overflow, catastrophic loss of significance, rounding strategy, and the like? How does all of this relate to the attributes of variables in a high level language? This leads to the following general project statement.

SIMA. 10 High Level Language Support for a Combined Arithmetic Unit

Study methods by which the user of a standard high level language can utilize the full capabilities of a Combined Arithmetic Unit; the language should allow the user to construct algorithms which use anywhere from 1 to all of the arithmetics available on the unit.

Some of the uses of the Combined Arithmetic Unit, now that convenient access to it has been made (i.e., SIMA 2.10 has been completed and a given method implemented), are the following: (this list is not exhaustive, but merely representative of work in this area)

(1) automatic error analysis; theoretical and experimental investigations of the application of different types of error tracing arithmetic, e.g. expanding and contracting interval arithmetic, on a given class of problems - e.g. linear algebra; see, for example, [42] and [43],

(2) conduct extensive tests of probabilistic models for the propagation of roundoff errors in various arithmetics; see, for example, [21], [22], [23], [24], [25], [26], and [27],

(3) testing of function libraries of a given precision p through execution and evaluation in a precision q, $q > p$; see, for example, [44], and [46],

(4) generation of accurate constants for tables and function libraries in precision p; examples of this are the coefficients in an expansion of a function, Gaussian quadrature formulas, etc.; see, for example, [45],

(5) generation of accurate test data in precision p: an example of this is to employ an integer preserving matrix inversion technique to generate exact test inverses - this requires extended precision integer arithmetic; see, for example, [47],

(6) use of methods which are marginally unstable in precision p or require an extended range in precision p,

(7) conduct tests of storage requirements required when using various variable or arbitrary length arithmetics in particular highly used algorithms, e.g. polynomial zero determination, linear system solution, etc. in variable range and precision floating point arithmetic or variable length rational arithmetic.

3.0 Standards and Arithmetic Units

The MIX machine of Knuth [48] has an identifying number - the 1009. It is formed as the average of the numbers associated with 16 different actual machines,

"$\lfloor$(360+650+709+7070+U3+SS80+1107+1604+G20+B220+S200+920+601 +H800+PDP4+11)/16$\rfloor$ = 1009".

Now add to this list of machines additional familiar machines: 7094-II, 7030, 6600, 645, 370, PDP10, STAR, and on and on. The reader is asked to answer the following and similar questions: "On how many of these machines are the fixed and floating operations implemented in a mathematically consistent and useful way for users for all operand pairs given to the arithmetic unit? On how many machines would the fixed and floating point operations yield the same answer as the result of an arithmetic operation if the same precision were being used?" The work of Kahan [3, 9, 10, 11], Cody [4, 7] and Neely [6]

as well as many others cited herein and the author's own experience leaves one with the disquieting feeling that the number is small. The author believes that current state of affairs with respect to the non-standardization of the attributes of fixed and floating point arithmetic units, after almost 30 years of machine construction and programming effort, is a disgrace. It is hoped that the completion of Project SIMA. 1, Basic Arithmetic Unit, will be accompanied by results which can be used to standardize these attributes. The de facto standards which computer manufacturers and system designers impose on users are not often the best standards.

REFERENCES

It should be obvious from the list of references here that we need at least one additional project, SIMA. 0.

SIMA. 0 Annotated Bibliography
Prepare and distribute a comprehensive annotated bibliography of theoretical and experimental studies in machine arithmetics.

[1] Foster, Caxton, "A View of Computer Architecture", CACM, Vol. 15, No. 7, July 1972, pp. 557-565.

[2] Auerbach, Isaac L., "Technological Forcast 1971", Proceedings of the IFIP Congress 1971, Vol. 2, Editor C.V. Freeman, North-Holland Publishing Co., 1972, pp. 764-775.

[3] Kahan, W., "A survey of Error Analysis", Proceedings of the IFIP Congress 1971, Vol. 2, Editor C.V. Freeman, North-Holland Publishing Co., 1972, pp. 1214-1239.

[4] Cody, W.J., "Desirable Hardware Characteristics for Scientific Computation", ACM SIGNUM Newsletter, Vol. 2, No. 1, January 1971, pp. 16-31 (and Appendix, not in SIGNUM, but available from W.J. Cody).

[5] Shriver, B.D., "Microprogramming and Numerical Analysis", IEEE Transactions on Computers, Vol. c-20, No. 7, July 1971, pp. 808-811.

[6] Neely, Peter M., "On Conventions for Systems of Numerical Representations", in Proceedings of the ACM Annual Conference, Boston 1972, pp. 644-651.

[7] Cody, W.J., "The Influence of Machine Design on Numerical Algorithms", in 1967 Spring Joint Computer Conference, AFIPS Conference Proceddings, Vol. 32, Thompson Publishing, 1968, pp. 305-309.

[8] Gregory, Robert T., "On the Design of the arithmetic unit of a fixed word elngth computer from the standpoint of computational accuracy", IEEE Transactions on Electronic Computer, Vol. EC15, No. 4, pp. 255-257.

[9] Kahan, W., "Further Remarks on Reducing Truncation Errors", CACM, Vol. 8, No, 1, January 1965, p. 40.

[10] Kahan, W., "7094-II System Support for Numerical Analysis" SHARE Publication # 159, item C4537, December 1966.

[11] Kahan, W., "Four Aphorisms Concerning Floating Point Hardware Design", ACM SIGNUM Newsletter, Vol.3, No. 2, July 1968.

[12] Levitan, E.S., "A Problem Resolved - Almost", ACM SIGNUM Newsletter, Vol. 3, No. 3, October 1968.

[13] Lawson, C.L., "Study of the Accuracy of the Double-Precision Arithmetic Operations on the IBM 7094 Computer", Jet Propulsion Laboratory Technical Memorandum No. 33-142, Pasadena, California, July 1963.

[14] Matula, D.W., "Number Theoretic Foundations of Finite Precision Arithmetic" in Applications of Number Theory to Numerical Analysis, Editor S.K. Zaremba, Academic Press, 1972, pp. 479-489.

[15] Matula, D. W., "Significant Digits: Numerical Analysis or Numerology", Proceedings of the IFIP Congress 1971, Vol. 2, Editor C.V. Freeman, North-Holland Publishing Co., 1972, pp. 1278-1283.

[16] Matula, D. W., "A Formalization of Floating-Point Numeric Base Conversion", IEEE Transactions on Computers, Vol. C-19, No. 8, August 1970, pp. 681-692.

[17] Matula, D. W., "Towards an abstract mathematical theory of floating-point arithmetic", in 1969 Spring Joint Computer Conference, AFIPS Conference Proceedings, Vol. 34, Thompson Publishing, 1969, pp. 765-772.

[18] Matula, D. W., "In-and-Out Conversions", CACM, Vol.11, No. 1, January 1968, pp. 47-50.

[19] Matula, D. W., "Base conversion mappings", in 1967 Spring Joint Conference, AFIPS Conference Proceedings, Vol. 30, pp. 311-318.

[20] Garner, H.L., "The Classification of Finite Number Systems", in Proceedings of the IFIP Congress 1968, Vol. 1, Editor A. J.H. Morrell, North-Holland, Publishing Co., 1969, pp. 256-259.

[21] Marasa, J.D., and Matula, D.W., "A Simulative Study of Correlated Error Propagation in Various Finite Precision Arithmetics", Department of Applied Mathematics and Computer Science Report, Washington University, St. Louis, Missouri, 1971.

[22] Tienari, M., "A Statistical Model of Roundoff Error for Varying Length Floating-Point Arithmetic", BIT, Vol. 10, 1970, pp. 355-365.

[23] Kuki, H and Cody, W.J., "A Statistical Study of the Accuracy of Floating Point Number Systems", CACM, Col. 16, No. 4, April 1973, pp. 223-230.

[24] Cody, W.J., "Static and Dynamic Numerical Characteristics of Floating Point Arithmetic", to appear in IEEE Transactions on Computers.

[25] Urabe, M., "Roundoff Error Distribution in Fixed-point Multiplication and a Remark about the Rounding Rule", SIAM Journal of Numerical Analysis, Vol, 5, No. 2, June 1963, pp. 202-210.

[26] Hull, T.E., and Swenson, J.R., "Tests of Probabilistic Models for Propagation of Roundoff Errors", CACM, Vol. 9, No. 2, February 1966, pp. 108-113.

[27] Ashenhurst, R.L., "Experimental Investigation of Unnormalized Arithmetic", in Error in Digital Computation, Vol. 2., Editor L.B. Rall, John Wiley and Sons, Inc., 1965, pp pp. 3-37.

[28] Symposium on Computer Arithmetic, May 15-16, 1972, Department of Electrical Engineering, University of Maryland; presentations to be edited into a special issue of the IEEE Transactions on Computers.

[29] van Wijngaarden, A., "Numerical Analysis as an Independent Science", BIT, Vol. 6., 1966, pp. 66-81.

[30] Knuth, D.E., The Art of Computer Programming, Vol. 2, Seminumerical Algorithms, Addison-Wesley, 1968.

[31] Moore, R.E., Interval Analysis, Prentice Hall, Inc., 1966.

[32] Apostollottos, N., Kulisch, U., Krawczyk, R., Lortz, B., Nickel, K., and Wippermann, H., "The Algorithmic Language Triplex-Algol 60", to appear in Numerische Mathematik.

[33] Kornerup, P., "A proposal for a unified number representation giving normalized, unnormalized, and interval arithmetic operands" , Department of Computer Science Note, University of Aarhus, Aarhus Denmark.

[34] Ashenhurst, R.L., "Number Representation and Significance Monitoring" in Mathematical Software, Editor J.R. Rice, Academic Press, 1971, pp. 67-92.

[35] Gray, H.L., and Harrison, C., Jr., "Normalized Floating-Point Arithmetic with an Index of Significance, in Proceedings of the Eastern Joint Computer Conference, AFIPS, Vol. 16, 1956, pp. 244-248.

[36] Mitchell, J.N., Jr., "Computer Multiplication and Division Using Binary Logarithms", IRE Transactions on Electronic Computers, Vol. EC-11, No. 8, August 1962, pp. 512-517.

[37] Combet, M., von Zonneveld, H., and Verbeek, L., "Computation of the Base Two Logarithm of Binary Numbers", IEEE Transactions on Electronic Computers, Vol. EC-14, No. 6, December 1965, pp. 863-867.

[38] Hall, E.L., Lynch, D.D., and Dwyer, S.J., III, "Generation of Products and Quotients Using Approximate Binary Logarithms for Digital Filtering Applications", IEEE Transactions on Computers, Vol. C-19, No. 2, February 1970, pp. 97-105.

[39] Marino, D., "New Algorithms for the approximate evaluation of binary logarithms and elementary functions", Instituto di Fisica Report, Universita de Bari, Bari, Italy.

[40] Walther, J.S., "A unified algorithm for elementary functions", in Spring Joint Computer Conference, AFIPS, vol. 38, 1971, pp. 379-385.

[41] Krishnamurthy, E.V., Chakraborti, S., and Sankar, P.V., "Arithmetic Algorithms in a Negative Base", and "Deterministic Division Algorithm in a Negative Base", IEEE Transactions on Computers, Vol. C-22, No. 2, February 1973, pp. 120-125 and pp. 125-128.

[42] Richman, P.L., "Automatic Error Analysis for Determining Precision", CACM, Vol. 15, No. 9, September 1972, pp. 813-817.

[43] Chartres, B.A., "Automatic Controlled Precision Calculations", JACM, Vol. 13, No. 3, July 1966, pp. 386-403.

[44] Cody, W.J., "Software for the Elementary Functions", in Mathematical Software, Editor J.R. Rice, Academic Press, 1971.

[45] Thatcher, H.C., Jr., "Making Special Arithmetics Available", in Mathematical Software, Editor J.R. Rice, Academic Press, 1971.

[46] Miller, T.H., "Accurach Enhancement of the Fortran V Math. Library", Univac Product Development, Salt Lake City, Utah, 1970.

[47] Bareiss, E.A., "Sylvester's Identity and Multistep Integer-Preserving Gausian Elimination", Mathematics of Computation, Vol. 22, 1968, pp. 565-578.

[48] Knuth, D.E., The Art of Computer Programming, Vol. 1, Fundamental Algorithms, Addison-Wesley, 1968.