ISSN 0105-8517

FINITE PRECISION RATIONAL ARITHMETIC: SLASH NUMBER SYSTEMS*

David W. Matula
Peter Kornerup

DAIMI PB-166
October 1983

* This research was supported by the National Science Foundation under grant MCS-8012704 and by the Halibarton Foundation.

FINITE PRECISION RATIONAL ARITHMETIC:
SLASH NUMBER SYSTEMS*

Abstract

Fraction number systems described by fixed-slash and floating-slash formats are specified. The structure of arithmetic over such systems is prescribed by the rounding obtained from "best rational approximation". Multi-tiered precision hierarchies of both fixed-slash and floating-slash type are proposed and analyzed with regards to their support of both exact rational and approximative real computation.

I. INTRODUCTION AND SUMMARY

There are compelling reasons for investigating arithmetic in number systems composed of limited precision fractions. The simplicity and naturalness of specification of fraction number systems and arithmetic leads to a substantive body of number theory available to analyse and document properties of computation in such systems. To implement a fraction number system consider that:

- a set of fractions {p/q} can be represented as a set of integer pairs, where the finite precision limitation characterizing the set can be specified simply by bounds imposed on the size of the numerator and denominator in each allowed pair, and

- any real value not exactly representable in such a finite precision fraction system can be canonically rounded to a value of the system by choosing what in number theory literature is called a "best rational approximation".

Our companion paper [KM83] describes an arithmetic unit capable of performing addition, subtraction, multiplication and division incorporating this canonical rounding by best rational approximation on such fractional operands. Computation time is shown essentially equivalent to a floating-point divide operation of comparable precision per arithmetic operation. This speed is sufficiently competitive to make fraction number systems viable candidates for special application supplements to, or possibly alternatives for, a floating-point computation system.

Concern for numeric software portability has motivated several efforts [KM81], [IEEE81], [CHH83], for defining unique and efficient finite precision computation systems with user specifiable precision levels available at the programming language level. To prescribe a particular floating-point arithmetic system recall that parameters for base, precision, and exponent range, must be set. Other choices including rounding rule and possible allowance of denormalized numbers must be resolved to fully characterize the resultant number system and its attendant approximate real arithmetic. In contrast, fraction number systems may be fully characterized by a single (radix independent) integral valued parameter denoting the bound on numerator and denominator size. This parameter then serves quite naturally as an hierarchical precision specification variable. Approximate real arithmetic in fraction number systems is then uniquely prescribed by the canonical rounding obtained from best rational approximation of the exactly computed results. Lest this rounding rule appear esoteric consider the following.

The familiar process of rounding in radix representation illustrated by rounding the decimal representation 0.431464... (= 277/642) to four digits yielding 0.4315, is elegantly simple as it corresponds to a <u>truncation of the radix representation</u>. Although less familiar, the "best rational approximation rounding" process, illustrated by rounding 277/642 to the four digit fraction 22/51, can be seen to be equivalently simple as it results from <u>truncation of the corresponding (but not visible) continued fraction representation</u>.

The correspondence between fractions and continued fractions is fundamental to the characterization and implementation of finite precision rational arithmetic. A brief review of terminology will be useful here. A <u>fraction</u>, denoted p/q or $\frac{p}{q}$ is an ordered pair composed of a nonnegative integer <u>numerator</u> p, and a nonnegative integer <u>denominator</u> q, which are not both zero. The <u>quotient</u> (value) of p/q is the rational number determined by the ratio of p to q for $q \neq 0$, and is taken to be positive infinity when $q = 0$. The numerator and denominator of an <u>irreducible</u> fraction must have a greatest common divisor (gcd) of unity, other fractions being termed <u>reducible</u>. Two fractions are equal, denoted $p/q = r/s$, if $qr = ps$ ($p/q = r/s$ does not necessarily imply identical numerators and denominators). We do not always distinguish between "p/q" denoting a

fraction (the ordered pair) or the quotient (a rational number).

Every nonnegative rational number x has both an irreducible fraction representation p/q and a finite <u>continued fraction</u> expansion

$$x = \frac{p}{q} = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \ddots \, \cfrac{1}{a_m}}} \qquad a_i \geq 0, \tag{1}$$

also denoted $p/q = [a_0, a_1, \ldots, a_m]$, where the <u>partial quotients</u> a_i are integral and unique (canonical) with the added requirements $a_0 \geq 0$; $a_1 \geq 1$ for $1 \leq i \leq m-1$; and $a_m \geq 2$ when $m \geq 1$. The truncated continued fractions

$$\frac{p_i}{q_i} = a_0 + \cfrac{1}{a_1 + \ddots \, \cfrac{1}{a_i}} = [a_0, a_1, \ldots, a_i], \quad i = 0, 1, \ldots, m, \tag{2}$$

termed the <u>convergents</u> (or <u>best rational approximations</u>) of p/q, constitute a series of successively more accurate approximations.

Note then that

$$\frac{277}{642} = 0 + \cfrac{1}{2 + \cfrac{1}{3 + \cfrac{1}{6 + \cfrac{1}{1 + \cfrac{1}{3 + \frac{1}{3}}}}}} = [0,2,3,6,1,3,3].$$

The sequence of convergents to 277/642 are illustrated in continued fraction, fraction, and decimal radix representation along with the relative error of the approximations in Table 1, from which the rounding of 277/642 to the four digit fraction 22/51 is readily made visible.

continued fraction	fraction	decimal representation	relative error
[0]	0/1	0.0...	1
[0,2]	1/2	0.50...	0.15
[0,2,3]	3/7	0.428...	0.0067
[0,2,3,6]	19/44	0.4318...	0.00082
[0,2,3,6,1]	22/51	0.43137...	0.00021
[0,2,3,6,1,3]	85/197	0.431472...	0.000018
[0,2,3,6,1,3,3]	277/642	0.4314641...	0

Table 1. The sequence of best rational approximations to 277/642 and the relative errors of these approximations.

The rounding rule prescribed by best rational approximation is simply to truncate the continued fraction representation at the largest index such that the corresponding fraction fits the format limitation. Importantly:

(i) this process is unequivocally unique, i.e. we do not have to speculate on alternative rules for rounding "in the last place";

(ii) this process is in fact radix independent, i.e. the sequence of best rational approximations does not depend on the radix employed for representation of the fractions, the final rounded value being determined simply by the last (largest indexed) convergent not exceeding the bound on size of numerator and denominator for the target fraction number system.

The canonical nature of the rounding is important not just in the fact that there is a resultant substantive body of number theory to analyse the properties of computation in these systems. For the computer architect, the availability of a natural standard to guide the number system and arithmetic specification is the ultimate key to effective design in support of numeric software portability.

In [MK80] we presented the foundations of finite precision rational arithmetic drawing heavily on classical material [HW79] from the number theoretic topics of Farey fractions and continued fractions. Basic results from these sources will be utilized (without proofs) as needed in our development here. Our companion paper [KM83] should be consulted for details on the efficient implementation of an arithmetic unit operating on rational operands.

In this paper, number systems composed of finite precision fractions in convenient binary formats facilitating exception handling features for underflow/overflow, infinity, exact zero, and not-a-number are prescribed. Innovations include a status bit for the exact/approximate status of each value. This bit is included in the word format to facilitate run time monitoring of the compatible exact-rational/approximate-real arithmetic that can be hosted by finite precision rational arithmetic. Accepting as a design goal the support of user specifiable precision at the programming language level, we show that finite precision rational arithmetic provides a conveniently parameterized hierarchical precision environment. Hierarchical precision is investigated both for support of exact rational and approximate real arithmetic.

We prescribe formats characterizing both fixed-slash and floating-slash number systems [Ma75].

fixed-slash: $\boxed{\text{numerator}}$ / $\boxed{\text{denominator}}$

floating-slash: $\boxed{\text{slash position}}$ $\boxed{\text{numerator.../...denominator}}$

One format has an implicit slash "fixed" between equal sized numerator and denominator fields. The other has the "floating" slash position explicitly set by the value in an associated slash position field, with the sum of the number of bits in numerator and denominator prescribed.

In Section II we describe fixed-slash formats and analyse the resulting number systems and arithmetic. A major property of these systems is that the result of all arithmetic operations (+,-,x,/) on single word operands are exact in double word representation, yielding a hierarchical exact rational subsystem within an approximate real computation environment. The feature of "graduated double-to-single precision rounding bias towards simple fractions" derived from best rational approximation in these systems is described. The benefits of this rounding bias for hosting "approximate rational" arithmetic (approximation of arithmetic over the rational field) are demonstrated. Exaggeration of precision degradation for "approximate real" arithmetic due to this rounding bias is examined in detail, with result that a small loss of effective precision compared to format length should be assumed.

Floating-slash formats and properties of the resulting number systems and arithmetic are described in Section III. Important features shared in common with fixed-slash systems are briefly summarized. Emphasis is placed on describing the two features distinguishing floating-slash from fixed-slash number systems:

(i) the larger underflow-to-overflow range for comparable length formats,

(ii) the more uniform behaviour of relative-error-of-approximation over the whole underflow-to-overflow range.

Of major importance is the characterization of extended range floating-slash systems and the identification of a precision fill feature through interpretation of "denormalized numbers".

The extended range and precision fill features together allow specification of "extended floating-slash" systems having the traditional range and maximum relative gap size of comparable format length floating-point systems while containing an embedded "standard" floating-slash number system with "standard" floating-slash arithmetic as an accessible subsystem.

II. FIXED-SLASH NUMBER SYSTEMS

A. Format. The (2k+2)-bit fixed-slash number system is composed of 4-tuples (a, s, num, den) conveniently described by reference to the defining binary word format. The component fields illustrated in Figure 1 are: the sign bit s, the k-bit integer field num, the exact bit a (for exact/approximate status), and the k-bit integer field den.

s	num	a	den
0	1 ... k	k+1	k+2 ... 2k+1

Figure 1: (2k+2)-bit format fixed-slash number representation.

A fixed-slash number is termed normalized when gcd(num,den) = 1, and so corresponds to an irreducible fraction. An unnormalized fixed-slash number corresponds to a reducible fraction.

The value v of the fixed-slash number is.

(a) If num ≠ 0, den ≠ 0, then $v = (-1)^s(\text{num}/\text{den})$. [signed rational number]
(b) If num = 0 and den ≡ 1 mod 2, then $v = (-1)^s 0$. [signed zero]
(c) If den = 0 and num ≡ 1 mod 2, then $v = (-1)^s \infty$. [signed infinity]
(d) If den = 0 and num ≡ 1 mod 2, or if num = 0 and den ≡ 0 mod 2, then v denotes "not-a-number". [not-a-number]

Standard fixed-slash arithmetic shall denote rounding (by best rational approximation to the fixed-slash format limit) of the exactly computed operation (+,−,×,/) on finite valued fixed-slash operands. When v is a number, a = 0, denotes that the value is exact. a = 1 denotes that the value is an approximation. The state a = 1 should be set when the corresponding represented value results from any of the following conditions:

- Rounding error: The represented value is the result of rounding an otherwise exactly computed value that cannot be represented in the format provided.

- Inherited error: A computed number where one of the arguments had a = 1.

- Initial error: An initial number which is explicitly acknowledged to be not necessarily exact.

The exact bit and sign bit are unspecified when v is not-a-number.

A characterization of the numbers represented in a fixed-slash number system and an assessment of the space efficiency of the representation both follow from established number theory [HW79, MK80], as summarized in the following observations.

Observation 1: Independence of Base. The set of representable extended real values of the (2k+2)-bit fixed-slash number system are the extended rational values of the Farey fractions F_{2^k-1}, where the order-n Farey fractions F_n are defined by

$$F_n = \left\{ \pm\frac{p}{q} \;\middle|\; 0 \le p,q \le n,\ \gcd(p,q) = 1 \right\}. \tag{3}$$

Note further that representation of the order-n Farey fractions for any particular $n \le 2^k - 1$ can be achieved by restricting the fixed-slash numbers to num≤n, den≤n. □

Observation 2: Redundancy and Representation Efficiency. Redundancy in fixed-slash number systems entails a loss of less than one bit in storage efficiency independent of k.

- Basis: Redundancy occurs in fized-slash number systems since reducible as well as irreducible fractions can be represented. Dirichlet has shown (see [Kn81, p. 324] for a proof):

Theorem 1.

$$\lim_{n\to\infty} \frac{\left|\left\{\frac{p}{q} \middle| 1 \le p,\ q \le n,\ \gcd(p,q) = 1\right\}\right|}{n^2} = \frac{6}{\pi^2} = .6079\ldots \quad (4)$$

Thus approximately 60% of the representable fractions are irreducible for sufficiently large k, and direct computation of the percentage of irreducible fractions for small k is in close agreement with this limit. For a numeric word format to attain 60% of the possible bit patterns as distinct values is to lose less than one bit (in this case $\log_2(\pi^2/6) = 0.718$ bits) per word. □

Suppose convenience at the user language interface dictates prescribing fixed-slash number systems by the number of decimal digits allowed. By Observation 1 there is no need to employ binary coded decimal (BCD) representation to realize a faithful binary word implementation of such systems. E.g., imposing num, den $\le 10^9-1$ in the 64-bit fixed-slash number system yields exactly those fractions having at most 9 decimal digits each in numerator and denominator. In contrast a 64-bit BCD string would contain only 16 decimal digits (with no sign). Note that BCD strings use only 10 of 16 possible patterns (losing $\log_2(16/10) = 0.68\ldots$ bits) for every 4 bits, hence losing about 1/6 of the storage capacity. From Observation 2 it follows that faithful binary fixed-slash representation of "k-digit decimal" fixed-slash number systems will entail only a bounded loss of storage capacity independent of k.

B. Exception Handling. A variety of implementation dependent exception handling procedures for non-trapping modes can be supported by the fixed-slash format described in section II A. Proceeding with a computation after exception by message passing is readily facilitated. The primary features are summarized in the following observations.

Observation 3: Extended Arithmetic. Signed infinity and zero are provided in a natural and symmetric manner. An implementation of the standard rules of arithmetic with fractions implicity ignores the signs of zero and/or infinity and correctly realizes extended arithmetic with infinity in the projective mode. Signed infinity is available for implementation of extended arithmetic in the affine mode. Note that the presence of a zero field in either numerator or denominator and a unit in the low order position of the other field is sufficient to characterize a zero or infinity, respectively. Then the leading k-1 bits of that field are available to hold a message related to the corresponding zero or infinity value. □

Observation 4: Underflow/Overflow. The underflow and overflow thresholds are the reciprocals of each other. Underflow to either positive or negative zero and overflow to either positive or negative infinity may, by the exact/approximate bit, be distinguished from the occurrence of an exactly computed zero or infinity. The format then allows an implementation dependent message of k-1 bits to be stored with the zero or infinity to describe the underflow/overflow situation (as noted in Observation 3). The message may be passed on, or otherwise becomes transparent to standard arithmetic on the zero or infinity value. □

Observation 5: Not-a-Number. When either the numerator field or denominator field is zero, it is sufficient that the other field have a low order bit set to zero to determine that the value is not-a-number. Then the leading k-1 bits of that field are available to hold an implementation dependent message that may be passed on for exception handling or as debugging information. Note that the null word is in the not-a-number class and is conveniently available to designate an "unassigned value". □

Observation 6: Denormalized Numbers. Standard fixed-slash arithmetic shall be expected to return normalized fixed-slash numbers, a feature readily obtained with the arithmetic unit described in [KM83]. A denormalized fixed-slash number shall refer to an unnormalized fixed-slash number where a specific meaning is associated with the value of gcd(num,den). Such a message will be transparent to standard fixed-slash arithmetic on the denormalized number, but can be visible in an enhanced environment to exception handling or extended arithmetic procedures. □

C. Exact Unary Operations. Fixed-slash number systems allow exact computation for the primary unary rational operators and certain conversion operations as summarized in the following observations.

Observation 7: Exact Additive and Multiplicative Inverses. Every member of a fixed-slash number system has an exact (efficiently computed) additive inverse (by changing sign) and multiplicative inverse (by swapping the contents of the num and den fields). The exact bit is not changed for these operations. □

Observation 8: Absolute Value. The absolute value of a fixed-slash number is exactly and efficiently computed by setting the sign bit to the positive state, leaving the remaining portion of the word unchanged. □

Observation 9: Integer and Fraction Parts. For the fixed-slash number x, floor(x), ceiling(x), and x mod 1 are all exactly computable each yielding a numerator and denominator no larger than the respective components of x. Recalling from (1)

$$x = \frac{p}{q} = [a_0, a_1, a_2, \ldots, a_m],$$

we obtain floor(x) and ceiling(x) using the value of a_0; and x mod 1 using the value of $[a_1, a_2, \ldots, a_m]$, with proper modifications to account for the sign. Repeated application provides access to all of the partial quotients. The exact bit is not changed by these operations. □

Observation 10: Numerator, Denominator, GCD and Normalization. The numerator and denominator of any fixed-slash number can be directly extracted. If unnormalized (corresponding to a reducible fraction) the rounding provided by the arithmetic unit described in [KM83] efficiently normalizes the fixed-slash number providing at the same time the value of gcd(num,den). □

Observation 11: Radix Represented Input Conversion. Floating-point and fixed-point numbers for any radix can be represented as fractions with integral numerator, and denominator a power of the radix. Such input can always be exactly represented in a (2k+2)-bit fixed-slash number system having sufficiently large k, and otherwise properly rounded by best rational approximation. □

- Note that decimal data specified by a fixed-point format F i.j denoting an i decimal digit field with $j \leq i$ places assumed to the right of the decimal point can always be exactly input into a 64-bit fixed-slash number system for any $i \leq 9$. Furthermore, data expressed in non-decimal or mixed radix units, e.g. feet/inches, weeks/days/hours/minutes, and degrees/seconds, are exactly representable in any of the measurement units in fixed-slash number systems. Exact output conversion to mixed radix systems employing floor(r_ix) and (r_ix) mod 1 is also conveniently available. These features provide added capacity beneficial to faithful hosting of data processing applications.

D. Hierarchical Precision Exact Rational Arithmetic. One functional goal of providing an arithmetic precision hierarchy in a programming language is that the user be able to control that the results of certain arithmetic operations shall be exactly represented. Hardware supported precision hierarchies are generally of the multi-tiered single/double or single/double/quad form. These tiers are usually defined simply to provide comparable arithmetic on operands whose format widths are corresponding multiples of a certain base word format width, rather than by attempting to determine the minimal most efficient hardware needed to realize particular functional arithmetic goals on base word operands.

For fixed-slash arithmetic the functional goal of realizing exact arithmetic on base word operands can be achieved with essentially optimal hardware efficiency by a convenient single/double-tiered hierarchy as shown in the following theorem.

Theorem 2: Let $n = 2k-2 \geq 4$. For any two n-bit (format) fixed-slash numbers with finite values x, y, the values

$$(x + y),\ (x - y),\ (x \times y) \text{ and } (x/y) \tag{5}$$

are exactly representable as 2n-bit (format) fixed-slash numbers. This result is best possible for addition and subtraction in that some values $(x+y)$, $(x-y)$ are not representable as $2(n-1)$-bit fixed-slash numbers, and nearly best possible for multiplication and division in that some values $(x \times y)$, (x/y) are not representable as $2(n-2)$-bit fixed-slash numbers.

Proof: It is sufficient to consider nonnegative $x = p/q$ and $y = r/s$. By ordinary algebra of fractions

$$x \pm y = \frac{p}{q} \pm \frac{r}{s} = \frac{ps \pm qr}{qs},$$

$$x \times y = \frac{p}{q} \times \frac{r}{s} = \frac{pr}{qs},$$

$$x / y = \frac{p}{q} / \frac{r}{s} = \frac{ps}{qr}.$$

For the $n = (2k+2)$-bit (format) fixed-slash numbers x, y we obtain p, q, r, $s \leq 2^k-1$. Hence ps, pr, qs, $qr < 2^{2k}-1$, and $|ps \pm qr| < 2^{2k+1}-1$. Thus $(x \pm y)$ is exactly representable in a $2n = 2(2k+2) = 2(2k+1)+2$ bit format, and $(x \times y)$, (x/y) are exactly representable in a $2(n-1) = 2(2k+1) = 2(2k)+2$ bit format.

Choosing $x = 1/y = (2^k-1)/(2^k-2)$, we obtain

$$ps + qr = (2^k-1)^2 + (2^k-2)^2 > 2^{2k} \quad \text{for } k \geq 3,$$

where $\gcd(ps+qr, qs) = \gcd\left((2^k-1) + (2^k-2)^2, (2^k-1)(2^k-2)\right) = 1$. This confirms the necessity of a fixed-slash format of at least 2n bits for addition (similarly for subtraction). The necessity of a fixed-slash format of at least $2(n-1)$ bits for multiplication follows readily for $x = y = (2^k-1)/(2^k-2)$, and for division for $x = 1/y = (2^k-1)/(2^k-2)$. □

Letting SINGLE and DOUBLE denote n-bit (word) and 2n-bit (double word) formats of the same generic arithmetic type, the merger of functional goal with architectural convenience for fixed-slash arithmetic is succintly expressed in the following observation.

Observation 12: Single-to-Double Exact Arithmetic. All arithmetic operations $(+,-,\times,/)$ on any SINGLE fixed-slash operands yield exact DOUBLE fixed-slash results. □

Certain useful functional computations on SINGLE fixed-slash operands can also be shown to have exact DOUBLE or QUAD fixed-slash results, where QUAD denotes the corresponding generic 4n-bit (quadruple word) format. We state the following (noting that the proof follows in the same elementary manner as that of Theorem 2) as indicative of many that can be derived.

Lemma 3: Let a_i, b_i, $i = 0,1,2,3$, be integer valued SINGLE, and x a finite rational valued SINGLE, fixed-slash numbers. Then

$$y = \frac{a_0 + a_1 x}{b_0 + b_1 x} \tag{6}$$

is exactly representable as a DOUBLE fixed-slash result, and

$$y = \frac{a_0 + a_1 x + a_2 x^2 + a_3 x^3}{b_0 + b_1 x + b_2 x^2 + b_3 x^3} \tag{7}$$

is exactly representable as a QUAD fixed-slash result. □

It should be noted that the rational arithmetic unit described in [KM83] allows the computation of the full expression (6) in the same time as any one of the individual operations +,−,×,/, (a time roughly equivalent to one floating-point divide of comparable precision).

Fixed-slash arithmetic can host exact rational arithmetic in a manner that may be viewed as an extension of "type INTEGER" arithmetic. At the same time convenient support of traditional type INTEGER arithmetic as a subset of fixed-slash arithmetic (inserting the appropiate truncated integer divide) is feasible as summarized in the following observation.

Observation 13: Integer Compatibility. The (2k+2)-bit fixed-slash number system supports exact integer arithmetic over the full finite range $[-(2^k-1), 2^k-1]$ of the system. The format provides convenient architectural compatibility with sign-and-k-bit-magnitude integer representation by restriction to the leading k+1 bits, where $a = 0$ and den = 1 are maintained constant for all exact finite integer computations. □

Example: A four-tiered hierarchical precision fixed-slash arithmetic system is described in Table 2. Word formats of

Fixed-Slash Format	Word Width (in bits)	Numerator and Denominator Widths (in bits)	Underflow-to-Overflow Range (Decimal Equivalent)
HALF	32	15	$10^{\pm 4.5}$
SINGLE	64	31	$10^{\pm 9.3}$
DOUBLE	128	63	$10^{\pm 18.9}$
QUAD	256	127	$10^{\pm 38.2}$

Table 2. Format sizes and numeric ranges for a four-tiered fixed-slash arithmetic system.

32-, 64-, and 128-bits are consistent with sizes of commercially available hardware supported floating-point arithmetic formats, e.g. the IBM370 system (where, however, SINGLE denotes 32 bits).

Designation of the 64-bit format as SINGLE width corresponds in size most closely to the 60-bit format SINGLE width employed in the CDC Cyber architecture. A 64-bit SINGLE width for fixed-slash arithmetic is practically necessary both for sufficient range and accuracy of approximate arithmetic (accuracy is considered in the next two subsections).

Provision of a QUAD precision of 256-bits is suggested primarily to achieve a range comparable to the range of several commercially available floating-point systems, e.g. the DEC Vax and Honeywell 6000 series both have ranges of approximately $10^{\pm 38}$ for SINGLE and DOUBLE width floating-point formats. The IEEE proposed standard [IEEE81] also has range $\sim 10^{\pm 38}$ for SINGLE, however a broader range for DOUBLE. For the body of scientific problems where a tradeoff favoring greater range to format size than that implicit in fixed-slash systems is desired, the floating-slash and extended floating-slash systems described in Section III are recommended.

Provision for HALF width allows memory saving in the storage of small integers and the relatively simple fractions frequently encountered as exact input, e.g. in many linear programming applications. Furthermore, provision of the four tiers, HALF, SINGLE, DOUBLE and QUAD, provides a broader spectrum of user specifiable exact rational arithmetic control that could be efficiently supported at the programming language interface.

The fact that all fixed-slash arithmetic operands, including division, are exact in no more than twice the word length provides great utility to the feature of having an exact/approximate bit in the format of each fixed-slash number. This hardware facility can efficiently support a more comprehensive computation environment at the programming language level as noted in the following observation.

Observation 14: Synergistic Exact Rational and Approximate Real Computation. Fixed-slash numeric representation with the exact/approximate bit associated with each value provides support for compatible exact rational and approximate real arithmetic. Thus a synergistic dynamic precision controlled exact-rational/approximate-real computation environment could be accessible to the user at the programming language interface. □

Areas where a compatible exact-rational/approximate-real arithmetic computation environment could be beneficial include:

- Symbolic computation;
- Arithmetic for knowledge based systems;
- Combinatorial optimization, e.g. linear programming with sparse 0-1 constraint matrices.

Efficient hardware realization of compatible exact rational and approximate real arithmetic at relatively low cost in both computation time and architectural logic design complexity is an appealing dividend of fixed-slash representation. The strength of this exact-rational/approximate-real synergism also critically depends on the adequacy of support of approximate real arithmetic provided by fixed-slash computation, which is the subject of our next two subsections.

E. Precision of Approximation. The parameter k implicitly determines the precision of a (2k+2)-bit fixed-slash approximation as noted in the following observation.

Observation 15: Single Parameter Specification of Precision and Range. For the family of (2k+2)-bit fixed-slash number systems the parameter k hierarchically specifies both the precision of approximate representation and the underflow-to-overflow range. Arbitrarily high precision over any finite region and an arbitrarily large range are achieved for sufficiently large k. There is a natural tradeoff between growth of precision-of-approximation and growth of underflow-to-overflow range in the family of fixed-slash number systems. □

The ability of any specific finite precision number system to host approximate real arithmetic is determined at the microscopic level by the spacing, or size of gaps, between representable values of the system. For floating-point systems the "gap function" [Ma70], and/or equivalent "reciprocal-relative-spacing function" [BF80], are reasonably well behaved functions yielding the spacing at x (for x over the whole underflow-to-overflow range) in terms of the precision level of the system. Provision of an analogous "precision-of-approximation" function over the range of a fixed-slash system is accessible, but more complex. The analysis is aided by some pertinent facts from number theory.

Recalling that the values of the (2k+2)-bit fixed-slash numbers are just the order (2^k-1)-Farey fractions (3), we obtain [HW79, MK80]:

<u>Theorem 4</u>: Let $\frac{p}{q}$, $\frac{r}{s}$ be representable (2^k+2)-bit fixed-slash numbers bounding the open interval $\left(\frac{p}{q}, \frac{r}{s}\right)$, termed a <u>gap</u>, containing no other representable (2^k+2)-bit fixed-slash numbers. Further, assume $0 \le \frac{p}{q} < \frac{r}{s}$ are irreducible fractions. Then

(i) <u>Absolute gap size</u>:

$$\frac{r}{s} - \frac{p}{q} = \frac{1}{sq}, \tag{8}$$

(ii) <u>Relative gap size</u>:

$$\frac{\frac{r}{s} - \frac{p}{q}}{\frac{p}{q}} = \frac{1}{sp}, \qquad \text{for } \frac{p}{q} \ne 0, \tag{9}$$

(iii) <u>Gap fill</u>: Letting $\mathrm{FILL}\left(\frac{p}{q}, \frac{r}{s}\right)$ denote the set of all irreducible fractions in the gap $\left(\frac{p}{q}, \frac{r}{s}\right)$,

$$\mathrm{FILL}\left(\frac{p}{q}, \frac{r}{s}\right) = \left\{\frac{ip+jr}{iq+js} \;\middle|\; \gcd(i,j) = 1\right\}. \tag{10}$$ □

<u>Corollary 4.1</u>: The extremes of gap size over the (2k+2)-bit fixed-slash number system are:

(i) Absolute gap sizes over $\left[0, 2^k-1\right]$:

$$\text{max gap} = 1, \tag{11}$$

$$\text{min gap} = \frac{1}{(2^k-1)(2^k-2)} \sim 2^{-2k}, \tag{12}$$

where also

$$\text{max gap over } [0,1] = \frac{1}{2^k-1} \sim 2^{-k}, \tag{13}$$

(ii) Relative gap sizes over $\left[\frac{1}{2^k-1}, 2^k-1\right]$:

$$\text{max rel gap} = 2^{-(k-1)}, \tag{14}$$

$$\text{min rel gap} = \frac{1}{(2^k-1)(2^k-3)} \sim 2^{-2k}. \tag{15}$$ □

<u>Corollary 4.2</u>: For any x in any gap $\left(\frac{p}{q}, \frac{r}{s}\right)$ of the (2k+2)-bit fixed-slash numbers, the <u>gap size at x</u>, $\gamma(x)$, is given by

$$\gamma(x) = \frac{1}{sq} \sim \begin{cases} \dfrac{1}{\min\{q,s\}2^k} & \text{for } 0 < x < 1. \\[2ex] \dfrac{x}{\min\{q,s\}2^k} & \text{for } x > 1. \end{cases} \tag{16}$$

<u>Proof</u>: Let x be in the gap $\left(\frac{p}{q}, \frac{r}{s}\right)$ and assume without loss of generality that $q < s$. The fraction $(p+r)/(q+s)$ is in the gap $\left(\frac{p}{q}, \frac{r}{s}\right)$, hence not representable as a (2k+2)-bit fixed-slash number. It follows

(i) for $0 < x < 1$, that $s < 2^k \le q + s < 2s$, so $2^{k-1} < s < 2^k$ and $qs \sim q2^k$,

(ii) for $x > 1$, that $r < 2^k \le p + r < 2r$, so $qs \sim qr/x \sim q2^k/x$,

where the approximations are tight when $q \ll s$. □

The variation in gap sizes from $O(2^{-2k})$ to $O(2^{-k})$ noted in Corollary 1 for a (2k+2)-bit fixed-slash number system is very broad, being of the order "double-to-single" precision. This variation is, however, not capricious. By appropriate interpretation, Corollary 4.2 reveals a graduated precision hierarchy within a given fixed-slash number system that will be shown later to have its own merit for certain types of approximate computation.

Consider , for example, the 8-bit fixed-slash numbers (Farey fractions F_7) over the unit interval as illustrated in Figure 2. Visualization of gap sizes in left-to-right order

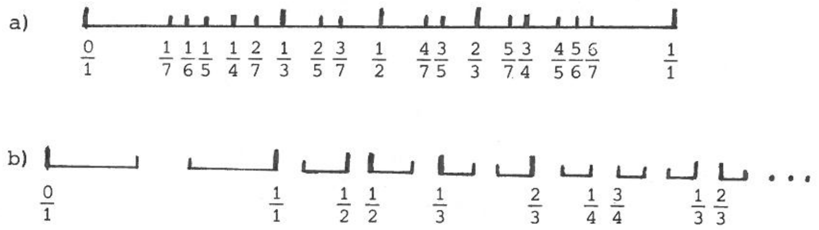

Figure 2. Gaps between representable values of the 8-bit fixed-slash number system over [0,1] arranged in a) continuous order, and b) in order by nonincreasing gap size.

(Figure 2a) yields an erradic (apparently chaotic) pattern. Alternatively, an enumeration of the gaps by decreasing size tagged only by the simpler fraction bound (Figure 2b) reveals the graduated precision hierarchy of gap sizes. The hierarchy is made more explicit in the following, where for simplicity we restrict attention to the unit interval.

Corollary 4.3: Let j be the smallest integer (level) such that the given irreducible fraction $0 \le p/q \le 1$ is representable as a (2j+2)-bit fixed-slash number. Then the gap size on either side of p/q in the (2k+2)-bit fixed-slash number system for $k > j$ is approximately $2^{-(j+k)}$. □

Recall for binary floating-point systems that the addition of one more bit to the mantissa (significand) doubles the number of representable values, by adding exactly one new representable value to each gap (bisecting each gap). There is an analogous but more complex situation for binary fixed-slash systems which we briefly summarize.

Corollary 4.4: Let F denote the (2k+2)-bit fixed-slash numbers, and F' the fixed-slash numbers obtained by allowing one more bit each in numerator and denominator, i.e. F' is the (2 (k+1) + 2)-bit fixed-slash numbers. Then we obtain [note: refer to Figure 2 for illustration with F for k = 2 having $\left\{\frac{0}{1},\frac{1}{3},\frac{1}{2},\frac{2}{3},\frac{1}{1}\right\}$ representable over [0,1], and F' for k = 3 having all values shown in the figure]:

(i) Density: F' has about four times the number of representable values as F,

(ii) Refinement: The <u>mediant</u> $\frac{p+r}{q+s}$ of each gap $\left(\frac{p}{q}, \frac{r}{s}\right)$ of F is representable in F'.

Furthermore, for $\frac{p}{q} < \frac{u_1}{v_1} < \frac{u_2}{v_2} < \ldots < \frac{u_m}{v_m} < \frac{r}{s}$, with $\frac{u_i}{v_i}$, $1 \le i \le m$, giving all values representable in F' falling in the gap $\left(\frac{p}{q}, \frac{r}{s}\right)$ of F,

(iii) Gap Fill Intensity: for $q < s$, $\left\lfloor \frac{s}{q} \right\rfloor \le m \le \left\lfloor \frac{s}{q} \right\rfloor + 3$,

(iv) Gap Fill Bias: for $q << s$, $\frac{u_1}{v_1}$ is approximately the midpoint of the gap $\left(\frac{p}{q}, \frac{r}{s}\right)$, so the gap $\left(\frac{p}{q}, \frac{u_1}{v_1}\right)$ of F' is still about one-half the size of the (previous) larger gap $\left(\frac{p}{q}, \frac{r}{s}\right)$ of F, where then the gaps $\left(\frac{u_1}{v_1},\frac{u_2}{v_2}\right)$, $\left(\frac{u_2}{v_2}, \frac{u_3}{v_3}\right)$, ..., $\left(\frac{u_m}{v_m}, \frac{r}{s}\right)$ of F' are each of the smaller (double-precision) size $\sim 1/s^2$, e.g. about 2^{-2k} for p/q in the unit interval. □

The precision of approximation available for representation of a real number x is thus biased in the neighbourhood of relatively simple fractions as summarized in the following observation.

Observation 16: Graduated Double-to-Single Precision Gaps. Gap sizes for the (2k+2)-bit fixed-slash numbers over the unit interval vary from a minimum of $\sim 2^{-2k}$ to a maximum of 2^{-k}. For fixed k, and any $j < k$, gaps of the (2k+2)-bit system, having as one bound a "simpler" fraction p/q with $p < q < 2^j$, have size at least as large as $2^{-(j+k)}$, other gaps ranging from $\sim 2^{-(j+k)}$ down to $\sim 2^{-2k}$. However, over [0,1] the "DOUBLE" 2n-bit fixed-slash numbers have a maximum gap size still smaller than the minimum gap size of a "SINGLE" n-bit system. Relative gap sizes vary from a minimum of $\sim 2^{-2k}$ to a maximum of $2^{-(k-1)}$ over the whole underflow-to-overflow range, with bias towards larger relative gap sizes both from a "simpler" fraction bound and with distance of the gap from unity. □

F. Approximate Real Arithmetic. To examine the accuracy of approximate real arithmetic in any finite precision number system it is necessary to specify the rounding employed when the result of an arithmetic operation is not exactly representable in the system. For fixed-slash arithmetic the rounding is canonically specified utilizing the notion of "best rational approximation".

Recall that every nonnegative real number x has a continued fraction expansion

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \ddots}} = [a_0,\ a_1,\ a_2,\ \ldots] \qquad (17)$$

where the partial quotients a_i are integral and unique (canonical) with the added requirements $a_0 \geq 0$; $a_i \geq 1$ for $1 \leq i$, and for terminating (rational) $x = [a_0,\ a_1,\ \ldots a_m]$ also $a_m \geq 2$ when $m \geq 1$. The truncated continued fractions

$$\frac{p_i}{q_i} = [a_0,\ a_1,\ \ldots,\ a_i],\ i = 0,\ 1,\ \ldots,\ m,$$

termed the convergents (or best rational approximations) of p/q, constitute a series of successively more accurate approximations whose principal properties are (see [HW79], [Kh63], or [MK80]):

(i) Recursive ancestry: With $p_{-2}=0, p_{-1}=1, q_{-2}=1$ and $q_{-1}=0$,

$$p_i = a_i p_{i-1} + p_{i-2},$$

$$q_i = a_i q_{i-1} + q_{i-2},$$

(ii) Irreducibility: $\gcd(p_i, q_i) = 1$,

(iii) Adjacency: $q_i p_{i-1} - p_i q_{i-1} = (-1)^i$,

(iv) Alternating convergence:

$$\frac{p_0}{q_0} < \frac{p_2}{q_2} < \cdots < \frac{p_{2j}}{q_{2j}} < \cdots \leq \frac{p}{q} \leq \cdots < \frac{p_{2j-1}}{q_{2j-1}} < \cdots < \frac{p_1}{q_1},$$

(v) Best rational approximation:

$$\frac{r}{s} \neq \frac{p_i}{q_i},\ s \leq q_i \Rightarrow \left|\frac{r}{s} - \frac{p}{q}\right| > \left|\frac{p_i}{q_i} - \frac{p}{q}\right|,$$

(vi) Quadratic convergence:

$$\frac{1}{q_i(q_{i+1} + q_i)} < \left|\frac{p_i}{q_i} - \frac{p}{q}\right| \leq \frac{1}{q_i q_{i+1}} \quad \text{for } i \leq m-1.$$

Let FXS_k denote the values of the (2k+2)-bit fixed-slash numbers (equivalently FXS_k denotes the Farey fractions F_{2^k-1}). The rounding Φ_k: Reals → FXS_k is defined for every real number x, where $p_0/q_0, p_1/q_1, p_2/q_2 \ldots$ are the convergents to |x|, by

$$\Phi_k(x) = \begin{cases} p_m/q_m & \text{if } x \geq 0,\ x = p_m/q_m \text{ with } p_m/q_m \leq 2^k-1, \\ p_i/q_i & \text{if } x > 0,\ p_i, q_i \leq 2^k-1 \text{ and } \max\{p_{i+1}, q_{i+1}\} > 2^k-1, \\ -\Phi_k(-x) & \text{if } x < 0. \end{cases} \qquad (18)$$

To illustrate the rounding suppose we are to round x = 277/642 into a 20-bit fixed-slash number, i.e. at most 9 bits in numerator and denominator. Note that

$$\frac{277}{642} = [0,2,3,6,1,3,3], \text{ with convergents } \frac{0}{1}, \frac{1}{2}, \frac{3}{7}, \frac{19}{44}, \frac{22}{51}, \frac{85}{197}, \frac{271}{642}.$$

(This example may be used also to illustrate the preceding properties (i)-(vi) of convergents). We obtain as the rounded value the truncated continued fraction

$$[0,2,3,6,1,3] = \frac{85}{197} = \frac{001010101_2}{011000101_2}$$

corresponding to the last convergent representable as a 20-bit fixed-slash number.

Theorem 5: The rounding ϕ_k: Reals $\rightarrow FXS_k$ satisfies the following three properties for all real x,y:

(i) Monotonic: $x < y \Rightarrow \phi_k(x) \leq \phi_k(y)$,

(ii) Antisymmetric: $\Phi_k(-x) = -\Phi_k(x)$,

(iii) Fixed points: $|x| = p/q \in FXS_k \Rightarrow \Phi_k(x) = x$.

Theorem 6: Let $\left(\frac{t}{u}, \frac{p}{q}\right)$ and $\left(\frac{p}{q}, \frac{r}{s}\right)$ be the gaps on both sides of $\frac{p}{q}$ in the (2k+2)-bit fixed-slash number system. Then the interval of values rounding to $\frac{p}{q}$ includes all values between the mediants of the two gaps, i.e.

$$\Phi_k(x) = \frac{p}{q} \quad \text{for} \quad \frac{t+p}{u+q} < x < \frac{p+r}{q+s}, \tag{19}$$

and also each mediant itself whenever $\frac{p}{q}$ is the "simpler" fraction bounding the gap, i.e.

$$\Phi_k\left(\frac{p+r}{q+s}\right) = \frac{p}{q} \quad \text{iff} \quad q < s \text{ (hence also } p < r),$$
$$\Phi_k\left(\frac{t+p}{u+q}\right) = \frac{p}{q} \quad \text{iff} \quad q < u \text{ (hence also } p < t). \tag{20}$$

□

Thus the rounding Φ_k: Reals $\rightarrow FXS_k$ effectively satisfies the rule "round away from the mediant towards the boundary of each gap, rounding the mediant to the simpler fraction bounding the gap". Thus we say Φ_k is mediant rounding into FXS_k.

Corollary 6.1: The mediant rounding error

$|x - \Phi_k(x)|$ for $\Phi_k(x) = p/q < \infty$ satisfies:

Absolute error bound:

$$|x - \Phi_k(x)| < \frac{1}{q2^k}, \tag{21}$$

Relative error bound:

$$\left|\frac{x - \Phi_k(x)}{\Phi_k(x)}\right| < \frac{1}{p2^k} \quad \text{for } p \neq 0. \tag{22}$$

□

It is important to note from Corollary 6.1 that most of the interval within a larger gap rounds to the simpler fraction bound. This provides then that the "simplicity" of the fractional result provides a measure of the graduated precision-of-approximation bias of the rounding. More specifically:

Corollary 6.2: If $\Phi_k(x) = p/q$, with $j \leq k$ the smallest integer such that p/q is representable as a simpler (2j+2)-bit fixed-slash number, then

$$\text{(i)} \quad |x - \Phi_k(x)| < 2^{-(j+k)}, \tag{23}$$

which is essentially a best possible bound in that

$$\text{(ii)} \quad |x - \Phi_k(x)| \sim 2^{-(j+k)} \text{ for x the mediant of a gap in } FXS_k. \tag{24}$$

□

To illustrate the rounding suppose we are to round $x = 277/642$ into a 20-bit fixed-slash number, i.e. at most 9 bits in numerator and denominator. Note that

$$\frac{277}{642} = [0,2,3,6,1,3,3], \text{ with convergents } \frac{0}{1}, \frac{1}{2}, \frac{3}{7}, \frac{19}{44}, \frac{22}{51}, \frac{85}{197}, \frac{271}{642}.$$

(This example may be used also to illustrate the preceding properties (i)-(vi) of convergents). We obtain as the rounded value the truncated continued fraction

$$[0,2,3,6,1,3] = \frac{85}{197} = \frac{001010101_2}{011000101_2}$$

corresponding to the last convergent representable as a 20-bit fixed-slash number.

Theorem 5: The rounding Φ_k: Reals $\to FXS_k$ satisfies the following three properties for all real x,y:

(i) Monotonic: $x < y \Rightarrow \Phi_k(x) \le \Phi_k(y)$,

(ii) Antisymmetric: $\Phi_k(-x) = -\Phi_k(x)$,

(iii) Fixed points: $|x| = p/q \in FXS_k \Rightarrow \Phi_k(x) = x$.

Theorem 6: Let $\left(\frac{t}{u}, \frac{p}{q}\right)$ and $\left(\frac{p}{q}, \frac{r}{s}\right)$ be the gaps on both sides of $\frac{p}{q}$ in the (2k+2)-bit fixed-slash number system. Then the interval of values rounding to $\frac{p}{q}$ includes all values between the mediants of the two gaps, i.e.

$$\Phi_k(x) = \frac{p}{q} \quad \text{for} \quad \frac{t+p}{u+q} < x < \frac{p+r}{q+s}, \tag{19}$$

and also each mediant itself whenever $\frac{p}{q}$ is the "simpler" fraction bounding the gap, i.e.

$$\begin{aligned} \Phi_k\left(\frac{p+r}{q+s}\right) &= \frac{p}{q} \quad \text{iff} \quad q < s \text{ (hence also } p < r), \\ \Phi_k\left(\frac{t+p}{u+q}\right) &= \frac{p}{q} \quad \text{iff} \quad q < u \text{ (hence also } p < t). \end{aligned} \tag{20}$$

□

Thus the rounding Φ_k: Reals $\to FXS_k$ effectively satisfies the rule "round away from the mediant towards the boundary of each gap, rounding the mediant to the simpler fraction bounding the gap". Thus we say Φ_k is <u>mediant rounding</u> into FXS_k.

Corollary 6.1: The mediant rounding error

$$|x - \Phi_k(x)| \quad \text{for} \quad \Phi_k(x) = p/q < \infty \text{ satisfies:}$$

Absolute error bound:

$$|x - \Phi_k(x)| < \frac{1}{q2^k}, \tag{21}$$

Relative error bound:

$$\left|\frac{x - \Phi_k(x)}{\Phi_k(x)}\right| < \frac{1}{p2^k} \quad \text{for } p \neq 0. \tag{22}$$

□

It is important to note from Corollary 6.1 that most of the interval within a larger gap rounds to the simpler fraction bound. This provides then that the "simplicity" of the fractional result provides a measure of the graduated precision-of-approximation bias of the rounding. More specifically:

Corollary 6.2: If $\Phi_k(x) = p/q$, with $j \le k$ the smallest integer such that p/q is representable as a simpler (2j+2)-bit fixed-slash number, then

$$\text{(i)} \quad |x - \Phi_k(x)| < 2^{-(j+k)}, \tag{23}$$

which is essentially a best possible bound in that

$$\text{(ii)} \quad |x - \Phi_k(x)| \sim 2^{-(j+k)} \quad \text{for } x \text{ the mediant of a gap in } FXS_k. \tag{24}$$

□

For our preceding example note that 277/642 is the mediant of the gap $\left(\frac{192}{445}, \frac{85}{197}\right)$ in the 20-bit fixed-slash numbers. Hence $\Phi_9\left(\frac{277}{642}\right) = \frac{85}{197}$ with a rounding error of $1/(642 \times 197)$ or $\sim 2^{-17}$, consistent with (24). The most important features of mediant rounding are summarized in the following observation.

<u>Observation 17: Canonical Rounding</u>. The rounding determined by truncating the continued fraction representation (best rational approximation) is unique and may be interpreted as the "round away from mediant" rule. This mediant rounding effects a graduated precision bias towards simpler fractions, i.e. for $j < k$, rounding to a $(2j+2)$-bit fixed-slash number within the set of $(2k+2)$-bit fixed-slash numbers is tolerated with error as large as $2^{-(j+k)}$. □

A functional goal motivating the provision of a user controlled variable or multi-tiered precision hierarchy is to allow for appropriate portions of a computation sequence to be carried out in higher precision. Given that "graduated (downsizing) double-to-single precision rounding bias towards simpler fractions" is implicit without user request in a fixed-slash number system, we must ask if there is a significant class of problems for which this type of adaptive precison applies naturally to the appropriate portions of a computation sequence. We find an affirmative answer by distinguishing two types of approximate computation.

- <u>Type Q: Approximate Rational Computation</u>. This corresponds to the class of finite sequences of rational arithmetic $(+,-,\times,/)$ operations on exact initial rational values (i.e. arithmetic over the rational number field), where approximation is employed whenever intermediate or final exact results would require too much storage. Examples of such computation occur in symbolic computation, combinatorial optimization, and operations on rational matrices.

- <u>Type R: Approximate Real Computation</u>. This corresponds to the class of finite sequences of real arithmetic $(+, -, \times, /, \sqrt{\ }$, exp, log, sin,....) operations on real or approximate operands (exact reals, approximate values, intervals).

A benefit of the "graduated double-to-single precision rounding bias towards simple fractions" feature for hosting approximate rational arithmetic is summarized in the following observation and then illustrated by an example [MK79].

<u>Observation 18: Recovery of Exactness</u>. A moderate length approximate rational computation hosted by fixed-slash arithmetic will have an accumulated error governed with high probability by a near double precision error bound. If the exact result of the same rational computation is a rather simple fraction, the implicit single precision rounding interval associated with this simple fraction is very likely to contain the approximate near double precision computed result prior to the final rounding, so that the final rounding then recovers the exact simple fractional result. □

<u>Example</u>: Figure 3 illustrates the computation of the determinant

$$D = \det \begin{vmatrix} \frac{10}{13} & \frac{20}{17} & \frac{1}{13} \\ \frac{11}{19} & \frac{7}{11} & \frac{77}{95} \\ \frac{69}{91} & \frac{4}{17} & \frac{56}{65} \end{vmatrix} = \frac{5}{13}$$

by evaluation of the rational expression

$$\left(\left(\left(\frac{10}{13}\times\frac{7}{11}\right)\frac{56}{65}+\left(\frac{11}{19}\times\frac{4}{17}\right)\frac{1}{13}\right)+\left(\frac{69}{91}\times\frac{20}{17}\right)\frac{77}{95}\right)-\left(\left(\left(\frac{69}{91}\times\frac{7}{11}\right)\frac{1}{13}+\left(\frac{11}{19}\times\frac{20}{17}\right)\frac{56}{65}\right)+\left(\frac{10}{13}\times\frac{4}{17}\right)\frac{77}{95}\right)$$

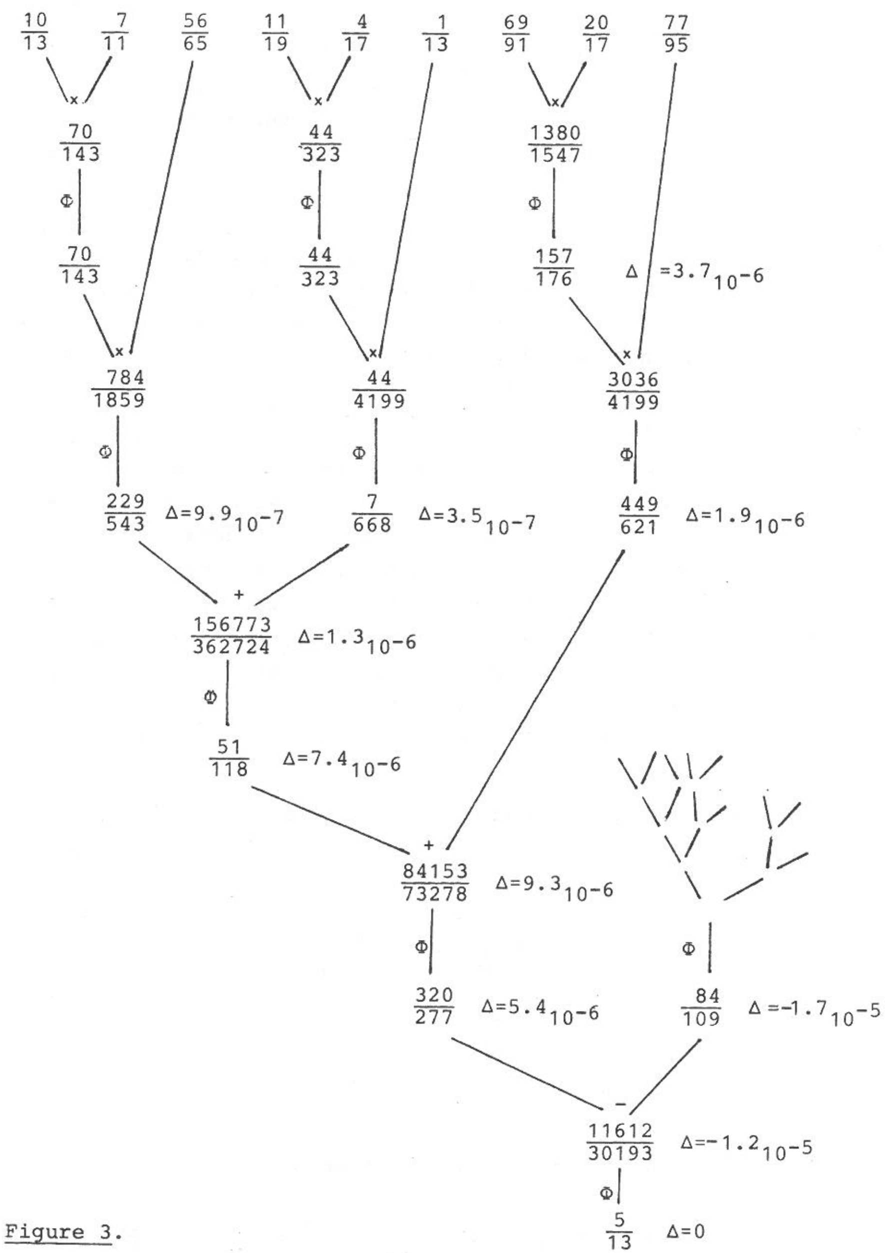

Figure 3.

in the order indicated by the parentheses. The fixed-slash approximate computation employed rounds all intermediate fractions by the mediant rounding Φ: Reals $\rightarrow F_{999}$, i.e. so as not to exceed 3 decimal digits in either numerator or denominator. Rounded values along with the absolute and relative errors accumulated at each stage are illustrated. Note that the final step of the computation involves the rounding of $\frac{320}{277} - \frac{84}{109} = \frac{11612}{30193}$, whose convergents are $0, \frac{1}{2}, \frac{1}{3}, \frac{2}{5}, \frac{3}{8}, \frac{5}{13}, \frac{1288}{3349}, \frac{2581}{6711}, \frac{11612}{30193}$. Thus $\Phi\left(\frac{11612}{30193}\right) = \frac{5}{13}$, and the true result is recovered by the final rounding.

For approximate real computation with no discernible preference for rational valued results, the graduated precision feature tends only to aggravate computational error. We show, however, that the degradation is not so severe as might be anticipated from the size of the maximum gap and maximum relative error. The analysis is aided by the following results from [MK80].

Theorem 7. For the random variable X chosen uniformly on [0,1] the expected value and the variance of the rounding error $|X - \Phi_k(X)|$, for rounding to (2k+2)-bit fixed-slash numbers, are given by:

$$\text{Exp}\left(\left|X - \Phi_k(X)\right|\right) = \frac{6 \log 2}{\pi^2} \frac{k}{2^{2k}} + O\left(\frac{1}{2^{2k}}\right), \tag{25}$$

$$\text{Var}\left(\left|X - \Phi_k(X)\right|\right) = \frac{c}{2^{3k}} + O\left(\frac{k^2}{2^{4k}}\right), \tag{26}$$

for c a constant. □

Theorem 8: If X is chosen uniformly on [0,1], then for any α, $1 \le \alpha \le 2$, with Φ_k denoting rounding to (2k+2)-bit fixed-slash numbers,

$$\text{Prob}\left\{\left|X - \Phi_k(X)\right| > \frac{1}{(2^k-1)^\alpha}\right\} \le \frac{2}{(2^k-1)^{2-\alpha}}. \quad (27)$$

□

From Theorem 7 we note that the expected rounding error into (2k+2)-bit fixed-slash numbers is of an order much closer to the double precision level $\sim 2^{-2k}$, than the single precision level $\sim 2^{-k}$, especially for large k. As the variance in rounding error is high, however, it is more important to ask for a bound on error size that will be violated with probability less than some very small tolerance, e.g. one-in-a-million (10^{-6}) or one-in-a-trillion (10^{-12}), which can be found from Theorem 8.

Fixed-Slash Format	Word Width (in bits)	Gap Size over [0,1]		Precision-of-Approximation				
		min	max	Avg	Mediant	One-in-a-Million	One-in-a Trillion	Max
HALF	32	$10^{-9.0}$	$10^{-4.5}$	$10^{-8.2}$	$10^{-9.0}$	$10^{-4.5}$	$10^{-4.5}$	$10^{-4.5}$
SINGLE	64	$10^{-18.6}$	$10^{-9.3}$	$10^{-17.5}$	$10^{-18.6}$	$10^{-12.3}$	$10^{-9.3}$	$10^{-9.3}$
DOUBLE	128	$10^{-37.9}$	$10^{-18.9}$	$10^{-36.5}$	$10^{-37.9}$	$10^{-31.6}$	$10^{-25.6}$	$10^{-18.9}$
QUAD	256	$10^{-76.4}$	$10^{-38.2}$	$10^{-74.7}$	$10^{-76.4}$	$10^{-70.1}$	$10^{-64.1}$	$10^{-38.2}$

Table 3: Gap size and rounding error bounds over [0,1] for certain fixed-slash number systems.

Table 3 shows a precision-of-approximation profile for each of the four tiers of the hierarchical precision fixed-slash arithmetic system of Table 2. The profile gives, over the unit interval of each fixed-slash number system,

(i) the minimum and maximum gap size,

(ii) the average rounding error,

(iii) the mediant rounding error bound (defined by one-half of all numbers chosen uniformly over [0,1] should incur at most that error),

(iv) the one-in-a-million rounding error bound (defined by only one in a million rounded values over [0,1] should incur an error larger than that value),

(v) the one-in-a-trillion error bound,

(vi) the maximum possible error.

All entries are given in decimal exponent form so the values may be interpreted loosely to give equivalent numbers of "decimal digits of accuracy".

For input of several thousand values the one-in-a-million bound would provide a conservative estimate of precision-of-approximation for all rounded inputs. For an extended computation on a machine performing 100 million (10^8) operations per second, the one-in-a-trillion error bound should be a conservative estimate of rounding precision for all results. Note that the compounded accumulated error of extended computation should probably dominate the infrequent larger errors introduced by the rounding precision bias of fixed-point computation. Thus the "average error" precision-of-approximation value could be used as a first order estimate for comparison with accuracy of other finite precision number systems.

Primary effects of the graduated precision environment on approximate real computation are summarized in the following observation.

Observation 19: Precision of Approximate Real Computation. Rounding values from [0,1] to fixed-slash numbers with k-bit numerators and denominators yields an average error comparable to that of $(2k-\log_2 k)$-bit binary radix representation. Extended computation (assumed normalized to the unit interval) is thus hosted with (absolute) approximation errors much closer to the double precision level than the single precision (worst case guarantee) level of the single-to-double variable precision scale. Larger formats lose proportionally less precision. The compounded accumulated error of extensive computation should likely dominate subsequent precision loss due to the graduated precision environment. □

The net effect from Observations 18 and 19 is that the beneficial aspects of support of approximate rational arithmetic are achieved with no great degradation in support of approximate real computation for the larger format sizes (64, 128, 256)-bit fixed-slash number systems.

The graduated precision environment of fixed-slash number systems should not be considered an insurmountable obstacle for approximate computation with fractions if near uniform spacing of representable values is considered imperative for efficient use of resources. The use of denormalized numbers to yield precision-fill in a manner analogous to that described for floating-slash representation in the next section is possible also for fixed-slash representation. We leave the details of this feature to the floating-slash representation discussion, where both precision fill and range extension become convenient extension options.

There are then two remaining seemingly essential practical considerations applying to use of fixed-slash computation:

(i) there is a rather small numeric range to format size tradeoff for any format size,

(ii) the support for more uniform absolute error behaviour over [0,1] is achieved at the expense of relative error degrading with magnitudes away from unity.

We shall now show in the next section that floating-slash representation affords many of the same desirable features we have documented for fixed-slash representation. In contrast to the limitations just cited for fixed-slash computation, floating-slash systems are shown to provide a larger numeric range to format size tradeoff and more uniform control of relative-error-of-approximation over the whole underflow-to-overflow magnitude range.

III. FLOATING-SLASH NUMBER SYSTEMS

A. Format. The $(k+\ell+1)$-bit floating-slash number system is composed of 4-tuples (a, s, f, exs) conveniently described by reference to the fields of the defining binary word format. The component fields illustrated in Figure 4 are: the sign bit s,

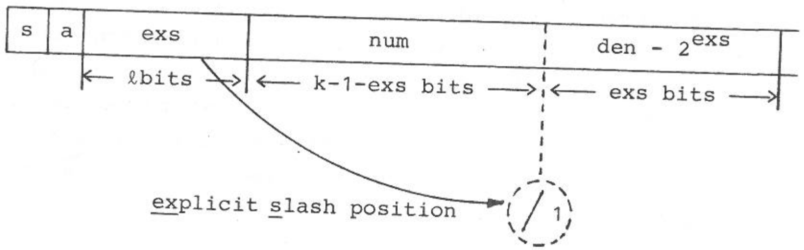

Figure 4. Format for floating-slash number representation incorporating an implicit leading denominator bit.

the exact/approximate bit a, the ℓ-bit integer field exs where $\ell = \lceil \log_2 k \rceil$, and the $(k-1)$-bit fraction field f containing the concatenated numerator and leading-bit-deleted denominator values. This representation uses an implicit leading denominator bit allowing the $(k-1)$-bit fraction field to represent a k-bit fraction, by which is meant any fraction with an i-bit integer numerator and j-bit integer denominator where $i + j \leq k$, $i,j \geq 1$. The value $exs = 2^{\ell}-1$ is reserved for encoding infinity and not-a-number. The value of exs, for $0 \leq exs \leq k-2$ $(\leq 2^{\ell}-2)$, is used to determine the slash position so that num is defined to be the integer in the leading k-1-exs bits of the fraction field. den is composed by adjoining the leading bit (value 2^{exs}) to the remaining exs bits in the fraction field yielding an (exs+1)-bit integer. Thus

$$num = \lfloor f/2^{exs} \rfloor, \; den = (f \bmod 2^{exs}) + 2^{exs}. \qquad (28)$$

With these integer values for exs, num, and den, we then define the value v of a floating-slash number to be:

(a) If $exs = 0$, then $v = (-1)^s$ num. [integers including signed zero]

(b) If $1 \leq exs \leq k-2$, then $v = (-1)^s(num/den)$. [signed rational numbers]

(c) If $exs = 2^{\ell}-1$ and $f \equiv 0 \bmod 2$, then $v = (-1)^s\infty$. [signed infinity]

(d) If $exs = 2^{\ell}-1$ and $f \equiv 1 \bmod 2$, then v denotes "not-a-number". [not-a-number]

For a standard floating-slash number system we shall have exs in the range $k-1 \leq exs \leq 2^{\ell}-2$ simply correspond to undefined values. For the extended range floating-slash number system we allow any exs field width $\ell \geq \lceil \log_2 k \rceil$ and interpret exs in the range $k-1 \leq exs \leq 2^{\ell}-2$ to provide an exponent e for scaling either the numerator or denominator as shown in Figure 5.

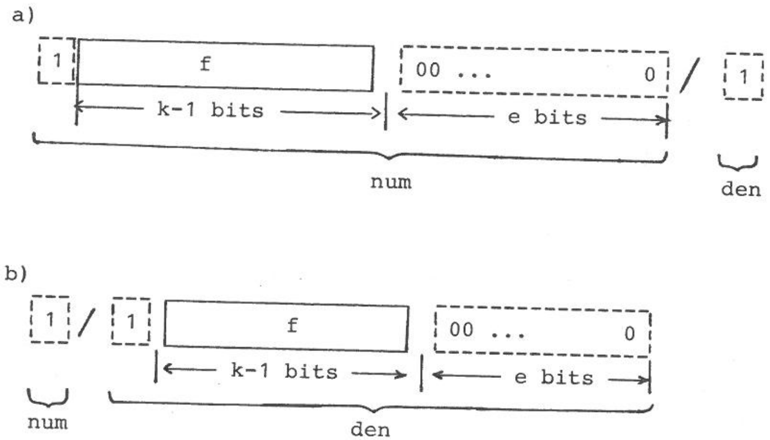

Figure 5. Extended range interpretation of floating-slash number representation.

In this interpretation we designate the $(k-1)$-bit fraction field to be a $(k-1)$-bit integer f which is augmented by a leading bit. The bit string then forms the leading k bits of either the numerator or denominator depending on whether the slash is interpreted to have "floated out" the right or left hand side of the word, respectively. The extended range floating-slash numbers corresponding to Figure 5a are equivalent to k-bit floating-point numbers. The small floating-slash numbers of Figure 5b do not correspond to floating-point numbers, being rather the inverses of the "large" floating-slash (or floating-point) numbers.

The value v of an "extended floating-slash" number in the extended range is

(e) If $\frac{k+2^{\ell}-3}{2} < \text{exs} \leq 2^{\ell}-2$, let $e = 2^{\ell}-2-\text{exs}$, and then
$v = (-1)^{s}2^{e}(2^{k-1}+f)$. [scaled numerators]

(f) If $k-1 \leq \text{exs} < \frac{k+2^{\ell}-3}{2}$, let $e = \text{exs}-k-1$, and then
$v = (-1)^{s}/(2^{e}(2^{k-1}+f))$. [scaled denominators]

(g) If $\text{exs} = \frac{k+2^{\ell}-3}{2}$, v is undefined.

A floating-slash number is termed <u>normalized</u> when gcd(num,den)=1 and so corresponds to an irreducible fraction. An unnormalized floating-slash number corresponds to a reducible fraction.

Standard floating-slash arithmetic shall denote rounding by best rational approximation to the standard (non extended) floating-slash format limit of the exactly computed operation (+,−,×,/) on finite valued floating-slash operands. When v is a number, a = 0 denotes that the value is exact, and a = 1 denotes that the value is an approximation. The state a = 1 should be set corresponding to rounding error, inherited error, and/or initial error as noted for fixed-slash arithmetic.

The values of the standard $(k+\ell+1)$-bit floating-slash numbers $(k\geq 2)$ correspond to the values of all signed irreducible k-bit fractions, denoted <u>FLS_k</u>, where

$$FLS_k=\left\{\pm\frac{p}{q} \;\middle|\; p,q\geq 1, \gcd(p,q)=1, \lfloor\log_2 p\rfloor+\lfloor\log_2 q\rfloor\leq k-2\right\}\cup\left\{\pm\frac{0}{1},\pm\frac{1}{0}\right\}. \quad (29)$$

Binary floating-slash number systems thus have a dependence on the base implicit in their characterization. To determine if this is likely to cause any base dependent anomalies consider the relation between FLS_k and the base independent set of <u>hyperbolic fractions</u> H_n defined by

$$H_n = \{\pm\tfrac{p}{q} \mid pq \leq n, \gcd(p,q) = 1\}. \quad (30)$$

It is readily determined that

$$FLS_{k-1} \subset H_{2^{k-1}-1} \subset FLS_k , \quad (31)$$

and the implications for assessing the extent of base dependance and representation efficiency are summarized in the following observations.

<u>Observation 20: Dependence on Base</u>. The floating-slash numbers corresponding to FLS_k are characterized in a base dependent manner. The natural representation independent system $H_{2^{k-1}-1}$ is contained in FLS_k with less than one bit loss in representation capacity. Properties of $H_{2^{k-1}-1}$ can be analyzed to assure the avoidance of base dependent anomalies in FLS_k. Restriction to those fractions with num×den≤n in part (b) of the value specification given after Figure 4 yields a realization of the values of the hyperbolic fractions H_n for any $n<2^{k-1}$ if such a canonical "representation independent" system is desired. □

Dirichlet has shown [see Di19, p. 283]

<u>Theorem 9</u>.

$$\lim_{n\to\infty} \frac{|H_n|}{|\{\pm\frac{p}{q} \mid pq\leq n,\ p\geq 1,\ q\geq 1\}|} = \frac{6}{\pi^2} = .6079\ldots . \quad (32)$$

Thus restriction to irreducible fractions in H_n loses only $\log_2(\pi^2/6) = 0.718$ bits of representation capacity as noted in Observation 2 for fixed-slash representation. Further losses due to employing FLS_k rather than H_{2^k-1}, and due to possible unutilized values of the exs field are correspondingly small, so:

Observation 21: Redundancy and Representation Efficiency. Floating-slash and extended range floating-slash number systems with the implicit leading denominator bit format of Figures 4, 5 incur a total loss in representation efficiency of only approximately one bit for any size format. □

B. Exception Handling. The important exception handling features of floating-slash systems applying to both standard and extended range formats are summarized here. Further details to explain and contrast these features with corresponding fixed-slash format features is available by reference to Observations 3-6.

- Signed infinities and signed zeroes are provided.
- Infinity may contain an implementation dependent message.
- Underflow and overflow thresholds are the reciprocals of each other.
- Not-a-number is provided.
- Denormalized numbers may be defined by giving meaning to the gcd of the reducible fraction corresponding to an unnormalized floating-slash number.

C. Unary Operators. The following unary rational operations on floating-slash and extended floating-slash numbers are exactly and efficiently computable and representable within the same $(k+\ell+1)$-bit format:

- Additive inverse.
- Multiplicative inverse.
- Absolute value.
- Integer part, fractional part (and successive partial quotient determination).
- Numerator, denominator.
- gcd, normalize.

Efficient numerator and denominator extraction is of great importance since to perform arithmetic it is necessary to move the numerator and denominator into separate registers. Since the unit position of the numerator can vary anywhere within the (k-1)-bit fraction field in the format of Figure 4, shifting is required to extract the numerator. If the numerator bits were stored in reverse order left adjusted in the field, the numerator could possibly be extracted more efficiently by first reversing the full (k-1)-bit field (twisting the wires) and then masking. Alternatively, the numerator could be stored at the right hand side of the field and the denominator at the left hand side with the denominator bits reversed with unit bit left adjusted. Resolution of these architectural questions involving the most efficient procedure for ordering and moving numerator and denominator into registers would not alter the nature of the represented values.

Example: For floating-slash representation of the convergent approximation $\pi = \sim 355/113$, which agrees with π to seven decimal places (rel. err. $\sim 10^{-7}$), note that s=0, a=1. Since $355=101100011_2$, $113=1110001_2=2^6+110001_2$, we must have $exs=6=110_2$. For a 32-bit floating-slash format, $\ell=5$, k=26. Figure 6 illustrates the representation of ~355/113 in the defining format (of Figure 4) in (a), and in the twisted-denominator-left-adjusted format in (b).

a)

b)

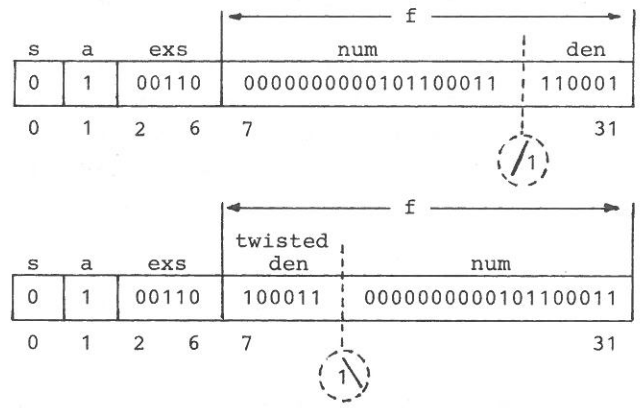

Figure 6: The 32-bit floating-slash representation of ~355/113 in standard format (a) and "twisted-denominator low-order-bit left adjusted" format (b).

D. Hierarchical Precision Rational Arithmetic. Regarding the extend of exact simple rational computation in standard floating-slash number systems note [MK78]:

- Multiplication or division of an i-bit fraction by a j-bit fraction yields a value representable as an (i+j)-bit fraction.

- The sum or difference of an i-bit fraction and a j-bit (j>i) fraction is representable as an (i+2j-1)-bit fraction; and furthermore, if both fractions have values greater than unity, then the result is representable as an (i+j+1)-bit fraction.

- Computing the difference $\frac{p}{q}-\frac{r}{s}=\frac{ps-qr}{qs}$ of two nearly equal fractions, a situation often termed "catastrophic cancellation", implies that $|ps-qr|<<|ps+qr|$. Thus the exact result will be representable without need for a great increase in precision, and possibly even with a smaller precision if $\frac{p}{q},\frac{r}{s}>>1$ so that qs is relatively small. In particular, if the difference of two k-bit fractions of value greater than $2^{k/3}$ is less than $2^{-k/3}$, then the result is also a k-bit fraction.

Let SINGLE, DOUBLE, and TRIPLE denote n-bit (word), 2n-bit (double word), and 3n-bit (triple word) formats of the same generic type. We summarize primary aspects of the exact rational arithmetic hosted by such a multi-tiered standard floating-slash computation system in the following observation.

Observation 22: Exact Rational Computation and Catastrophic Cancellation. Multiplication and division of standard SINGLE floating-slash operands yield exact DOUBLE floating-slash results. Addition and subtraction of standard SINGLE floating-slash operands yield exact TRIPLE floating-slash results. The occurrence of catastrophic cancellation, i.e. the difference of two nearly equal fractions, yields a great simplification in the resulting fraction, often allowing the difference of two SINGLE operands to be exactly represented in SINGLE format. □

Our utilization of an implicit leading denominator bit brings an important architectural property to the format beyond the savings of one bit in the representation.

Observation 23: Integer Compatibility. Floating-slash representation as provided by the format illustrated in Figure 4 will have a=0, exs=0 for exact normalized representation of integers. The normalized representation of any integer in the range $[-2^{k-1}-1,\ 2^{k-1}-1]$ will then be identical to the sign-and-$(k+\ell+1)$-bit-magnitude representation of that integer. This same result also holds for the "twisted" format with the denominator at the left, as illustrated in Figure 6b. □

A multi-tiered floating-slash system provides a synergistic integer/exact-rational/approximate-real computation environment with greater range than for corresponding fixed-slash format lengths.

Example: A four-tiered hierarchical floating-slash system using formats of size 32-, 64-, 128-, and 256-bits is shown in Table 4, showing ranges of about twice the number of digits compared to the same size formats for fixed-slash in Table 2.

Floating-Slash Format	Word Width (in bits)	Slash Position Field Width	Fraction Field Width (in bits) including implicit bit	Underflow-to-Overflow Range (Decimal Equivalent)
HALF	32	5	26	$10^{\pm 7.5}$
SINGLE	64	6	57	$10^{\pm 16.8}$
DOUBLE	128	7	120	$10^{\pm 35.8}$
QUAD	256	8	247	$10^{\pm 74.0}$

Table 4. Format sizes and numeric ranges for a four-tiered floating-slash arithmetic system.

Note that DOUBLE floating-slash representation with a 128-bit format achieves a range essentially equivalent to the DEC Vax and Honeywell 6000 series floating-point range (for their SINGLE or DOUBLE format). QUAD floating-slash (256-bits) gives essentially the range of the IBM 360 SINGLE, DOUBLE, or QUAD formats. Furthermore QUAD contains as a subset all values representable in the proposed IEEE standard 32-bit single floating-point format. (In fact all IEEE single x for $1 \le x \le 2^{56}-1 = \sim 10^{16.8}$ are exactly representable in "SINGLE" 64-bit floating-slash representation, and all IEEE double x for $1 \le x \le 2^{119}-1 = \sim 10^{35.8}$ are exactly representable in "DOUBLE" 128-bit floating-slash representation).

Two approaches have been utilized for hierarchical multi-tiered floating-point range specification:

- Same underflow-to-overflow range for all floating-point precision tiers. This is the case for most commercially available large computer systems, including Cray-1, Dec VAX, Honeywell 6000, IBM 370, Interdata 8/32 (see [BF80]).

- Increased underflow-to-overflow range in higher floating-point precision tiers. This is the case for the proposed IEEE standard [IEEE81], that specified by the Ada language feature for parametrically declared FLOAT precision (see [Co82]), and that suggested for the CADAC arithmetic unit [CHH83].

For one example of a multi-tiered hierarchical extended range floating-slash system we could have QUAD level remain a standard floating-slash system, with DOUBLE, SINGLE, and HALF being extended range floating-slash systems each having the same 8-bit slash position field. This provides downsizing of precision-of-approximation from QUAD to DOUBLE, SINGLE, or HALF, with a more moderate downsizing of range (e.g. for the 64-bit SINGLE format with $\ell=8$, $k=55$, the range is $\sim 10^{\pm 46.3}$, and for the 128-bit DOUBLE format $\ell=8$, $k=119$, the range is $\sim 10^{\pm 55.9}$, compared to QUAD range of $10^{\pm 74.0}$). Alternatively, following the more recent design trend, we may choose graduated broader ranges for extended range floating-slash in a manner comparable to those in existing and proposed floating-point systems, as suggested in Table 5.

Extended Floating-Slash Format	Word Width (in bits)	Slash Position Field Width	Fraction Field Width (in bits) including implicit bit	Underflow-to-Overflo Range (Decimal Equivalent
HALF	32	8	23	$10^{\pm 41.5}$
SINGLE	64	11	52	$10^{\pm 315}$
DOUBLE	128	14	113	$10^{\pm 2482}$

Table 5. Format sizes and numeric ranges for a three-tiered extended range floating-slash arithmetic system.

E. Precision of Approximation. The parameter k implicitly determines the precision-of-approximation as well as the underflow-to-overflow range for standard floating-slash systems. Essential features are summarized in the following observation.

Observation 24: Floating-Slash Range/Precision Specification. Floating-slash representation provides about twice the exponent range of fixed-slash representation for the same word size. The nested sequence $FLS_k \subset FLS_{k+1} \subset FLS_{k+2} \subset \ldots$ results in floating-slash systems having both greater range and precision governed naturally in their growth by a single parameter. Extended range floating-slash representation allows a separately parameterized range/precision specification if desired. The additional representable values of the extended range system all have absolute values outside the positive finite magnitude range $[1/(2^{k-1}-1), 2^{k-1}-1]$ of the included subsystem FLS_k, preserving the integrity of computations and roundings within the finite magnitude range of FLS_k. □

As it can be shown [MK80] that the hyperbolic fractions H_n become log uniformly dense on the positive real line as $n\to\infty$, it follows that floating-slash representation yields "macroscopically" uniform relative error control for all parts of the range. This contrasts favorably with fixed-slash systems where relative errors consistently increase for approximations of quantities farther away from unity towards the overflow, or underflow, boundary.

At the "microscopic" level of gap size, we obtain a single-to-double precision variation in relative gap size graduated by the "simplicity" of the simpler fraction gap bound. Results for floating-slash relative gap size variation over the whole underflow-to-overflow range are analogous to those of absolute gap size variation over [0,1] for fixed-slash representation. The following results adapted from [MK80] provide the relative gap size function and indicate the maximum and minimum bounds.

Theorem 10: For any x in any gap $(\frac{p}{q},\frac{r}{s})$ of the $(k+\ell+1)$-bit floating-slash numbers the relative gap size at x, $\gamma^*(x) = (\frac{r}{s}-\frac{p}{q})/x$, is given by

$$\gamma^*(x) = \frac{1}{sqx} \sim \frac{1}{(\min\{sr,pq\})^{\frac{1}{2}}2^{k/2}} \tag{33}$$

so that over $\left[\frac{1}{2^{k-1}-1}, 2^{k-1}-1\right]$

$$\text{max rel gap} \sim 2^{-k/2}, \qquad \text{min rel gap} \sim 2^{-k}. \tag{34}$$

□

E. Approximate Real Arithmetic. Rounding by best rational approximation into the $(k+\ell+1)$-bit floating-slash numbers is defined by Φ_k^*: Reals $\to$ FLS_k for every real number x, where $p_0/q_0, p_1/q_1, p_2/q_2, \ldots$ are the convergents to $|x|$, by

$$\Phi_k^*(x) = \begin{cases} p_m/q_m & \text{if } x\ge 0,\ x = \frac{p_m}{q_m} \in FLS_k, \\ p_i/q_i & \text{if } x>0,\ \frac{p_i}{q_i} \in FLS_k,\ \frac{p_{i+1}}{q_{i+1}} \notin FLS_k, \\ -\Phi_k^*(-x) & \text{if } x<0. \end{cases} \tag{35}$$

The rounding Φ_k^* satisfies the monotonic, antisymmetric, and fixed point properties just as noted for Φ_k in Theorem 5. Also Φ_k^*: Reals $\to$ FLS_k satisfies the rule "round away from mediant to the boundary of each gap, rounding the mediant to the simpler fraction bounding the gap". Thus we term Φ_k^*: Reals $\to$ FLS_k mediant rounding to FLS_k.

From [MK80] we obtain the following results on the average relative rounding error and on the distribution of relative error. For X chosen in a log uniform manner over the whole underflow-to-overflow range $\left[\frac{1}{2^{k-1}-1}, 2^{k-1}-1\right]$,

$$\text{Exp}\left(\frac{|X-\Phi_k^*(X)|}{X}\right) < \sim \frac{k}{2^{k-1}}, \tag{36}$$

and for $\frac{1}{2}\le\alpha\le 1$,

$$\text{Prob}\left\{\frac{|X-\Phi_k^*(X)|}{X} > \frac{1}{2^{(k-1)\alpha}}\right\} < \sim \frac{4\alpha}{2^{(k-1)(1-\alpha)}}. \tag{37}$$

These formulas are used to compute the precision-of-approximation profile for relative error of approximation in Table 6 for the same four-tiered floating-slash hierarchy utilized in Table 4.

Floating-Slash Format	Word Width (in bits)	Relative Gap Size		Precision-of-Approximation [Relative Error]			
		min	max	Avg	One-in-a-Million	One-in-a-Trillion	Max
HALF	32	$10^{-7.8}$	$10^{-3.9}$	$10^{-6.5}$	$10^{-3.9}$	$10^{-3.9}$	$10^{-3.9}$
SINGLE	64	$10^{-17.1}$	$10^{-8.5}$	$10^{-15.5}$	$10^{-10.4}$	$10^{-8.5}$	$10^{-8.5}$
DOUBLE	128	$10^{-36.1}$	$10^{-18.0}$	$10^{-34.2}$	$10^{-29.3}$	$10^{-23.4}$	$10^{-18.0}$
QUAD	256	$10^{-74.3}$	$10^{-37.1}$	$10^{-72.1}$	$10^{-67.4}$	$10^{-61.5}$	$10^{-37.1}$

Table 6: Relative gap sizes and bounds on relative rounding error for log uniform data over the whole underflow-to-overflow range for certain floating-slash number systems.

To avoid the graduated rounding precision bias of floating-slash representation note that for every "simple fraction" $\frac{p}{q}$ bounding a rather large gap in the $(\ell+k+1)$-bit floating-slash number system, the number of distinct unnormalized fractions equal to $\frac{p}{q}$ is of the same order as the size of the gap. Thus the possibility exists to give a meaning (as denormalized values) to these unnormalized fractions. By interpreting the value of the gcd of such unnormalized fractions to imply a distinct real value of the gap (rather than the value $\frac{p}{q}$), we may obtain a gap fill through denormalized values in each gap so that the overall relative error of each gap throughout the system is $O(2^{-2k})$.

Details of the encoding of such denormalized values and their decoding and use in an extension of the arithmetic unit of [KM83] will be developed in a subsequent paper.

REFERENCES

[BF80] W.S. Brown and S.I. Feldman: "Environment Parameters and Basic Functions for Floating-Point Computation". TOMS, Vol. 6, No. 4, Dec. 1980, pp. 510-23.

[CHH83] M. Cohen, T.E. Hull and V.C Hamacher: "CADAC: A Controlled-Precision Decimal Arithmetic Unit". IEEE TC Vol. C-32, No. 4, April 1983, pp. 370-77.

[Co82] W.J. Cody: "Floating-Point Parameters, Models and Standards", in "The Relationship between Numerical Computation and Programming Languages". Ed. J.K. Reid, North-Holland Publ. Co., 1982, pp. 51-65.

[Di19] L.E. Dixon: "History of the Theory of Numbers", Vol. 1, 1919, Reprint Chelsea Publ. Co. 1971.

[HW79] G.H. Hardy and E.M. Wright: "An Introduction to the Theory of Numbers", Oxford University Press, 5th Ed. 1979.

[IEEE81] "The Proposed IEEE Floating-Point Standard", four articles in Computer, Vol. 14, no. 3, March 1981.

[Kh63] A.Y. Khintchin: "Continued Fractions", translated from Russian by P. Wynn. P. Noordhoff Ltd., Grooningen, 1963.

[Kn81] D.E. Knuth: "The Art of Computer Programming, Vol. 2, Seminumerical Algorithms". Addison-Wesley Publ. Co., 2nd Ed. 1981.

[KM83] P. Kornerup and D.W. Matula: "Finite Precision Rational Arithmetic: An Arithmetic Unit". IEEE TC, Vol. C-32, No. 4, April 1983, pp. 378-87.

[KM81] U. Kulish and L. Miranker: "Computer Arithmetic in Theory and Practice", Academic Press, 1981.

[Ma70] D.W. Matula: "A Formalization of Floating-Point Numeric Base Conversion". IEEE TC, Vol. C-19, No. 8, Aug. 1970, pp. 681-92.

[Ma75] D.W. Matula: "Fixed-Slash and Floating-Slash Arithmetic". Proc. 3rd Symposium on Computer Arithmetic, IEEE Publ. No. 75CH 1017-3C, 1975, pp. 90-91.

[MK78] D.W. Matula and P. Kornerup: "A Feasibility Analysis of Binary Fixed-Slash and Floating-Slash Number Systems". Proc. 4th Symposium on Computer Arithmetic, IEEE Publ. No. 78CH 1412-6C, 1978, pp. 29-38.

[MK79] D.W. Matula and P. Kornerup: "Approximate Rational Arithmetic Systems: Analysis of Recovery of Simple Fractions during Expression Evaluation". Proc. EUROSAM 79, Lecture Notes in Computer Science, Vol. 72, Springer Verlag, 1979, pp. 383-97.

[MK80] D.W. Matula and P. Kornerup: "Foundations of Finite Precision Rational Arithmetic". Computing, Suppl. 2, Springer Verlag, 1980, pp. 85-111.