RATIONALLY BIASED ARITHMETIC

Warren E. Ferguson, Jr.
Mathematics

David W. Matula
Computer Science and Engineering

Southern Methodist University, Dallas, Texas 75275

Abstract

One can naively view a computer number system as a pair (F,P) consisting of a finite set F of real numbers and a rounding rule P. One such number system is a hyperbolic rational number system which has as F a finite set of rational numbers and as P the so-called mediant rounding rule. In this paper we demonstrate how one can simulate a hyperbolic rational number system in any high level language that supports floating point computation. From this simulation we infer that hyperbolic rational number systems form viable alternatives to traditional binary floating point number systems. Many properties of hyperbolic rational number systems are derived from the relationship of their rounding rule to the well-developed theory of best rational approximation.

1. Introduction and Summary

One can naively view a computer number system as a pair (F,P) consisting of a finite set F of real numbers called the machine representable numbers and a rounding rule P which maps real numbers into machine representable numbers. One of the fundamental purposes of the rounding rule P is to round the result of intermediate computations into machine numbers. Thus, if x and y represent machine numbers, then P(x+y) would be the machine number which represents the computed value of x+y.

For example, a binary floating point number system has as F those numbers in the set

$$\{ \pm \ (0.d_1 \ d_2 \ \ldots \ d_k)2^e \mid d_1 \neq 0, \ L < e < U \}$$

consisting of signed normalized binary fractions of bounded length k multiplied by bounded powers of 2 and as P that rule which chooses P(x) to be the value obtained by suitably truncating the normalized binary fraction representing x. Note that in this number system some common fractions such as 1/3, 1/5 and 1/10 are not machine numbers.

As an alternate number system let us consider a hyperbolic rational number system which has as F the numbers in the set

$$\{ \pm \ p/q \mid p,q > 0, \ \gcd(p,q)=1, \ pq < m \}$$

consisting of signed irreducible fractions of bounded complexity m and as P the mediant rounding rule. Here the complexity of a fraction p/q is defined to be the integer pq. Although the mediant rounding rule will be defined in Section 2 we can briefly describe the mediant rounded value P(x) of x as the number obtained by suitably truncating the ordinary continued fraction representing x, thus relating the mediant rounding rule to the well developed theory of best rational approximation. One important property of the mediant rounding rule is that it is biased towards simple fractions, that is fractions with small complexity [MK80].

The main result of this paper is the demonstration that hyperbolic rational number systems, with their biased rounding rule, form viable alternatives to traditional floating point number systems. To support this statement we show how one can simulate a hyperbolic rational number system using the host computer's floating point number system. In this simulation each fraction p/q is represented by the host computer's floating point approximation of p divided by q. To study the effectiveness of computations performed in hyperbolic rational number systems we consider a very special problem frequently studied in numerical linear algebra, the inversion of a Hilbert matrix.

The computation of the inverse of a Hilbert matrix is interesting for two reasons. The first reason concerns the fact the inversion of a Hilbert matrix is an inherently ill-conditioned problem. Therefore stable algorithms for computing matrix inverses, such as the Gaussian elimination algorithm to be described in Section 3, will have difficulty producing an accurate estimate of the inverse of a Hilbert matrix. The second reason concerns the fact that the numbers which arise during the computation of the inverse of a Hilbert matrix can

CH2146-9/85/0000/0194$01.00

be described using simple fractions. In Table 1 we present formulas describing the order n Hilbert matrix and its inverse [Ch83, Co39, FM67, GK78]. The ill-conditioning of the Hilbert matrix manifests itself in the large magnitude of some of the entries in the inverse of the Hilbert matrix, with the condition number of the order n Hilbert matrix growing like exp(3.525n) [To54]. For example the order 5 Hilbert matrix

$$H_5 = \begin{bmatrix} 1/1 & 1/2 & 1/3 & 1/4 & 1/5 \\ 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7 \\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \end{bmatrix}$$

has as its inverse [GK78]

$$H_5^{-1} = \begin{bmatrix} 25 & -300 & 1050 & -1400 & 630 \\ -300 & 4800 & -18900 & 26880 & -12600 \\ 1050 & -18900 & 79380 & -117600 & 56700 \\ -1400 & 26880 & -117600 & 179200 & -88200 \\ 630 & -12600 & 56700 & -88200 & 44100 \end{bmatrix}$$

Since the base 10 logarithm of the condition number estimates the number of digits of accuracy lost during the computation of the inverse of the Hilbert matrix we expect to lose about 1.53n digits of accuracy when we attempt to compute the inverse of the order n Hilbert matrix.

$$H = LU$$

$$H^{-1} = OHD$$

$H = (n_{i,j})$ ... Hilbert matrix

$L = (\lambda_{i,j})$ ... unit lower triangular matrix

$U = (u_{i,j})$ ... upper triangular matrix

$L^{-1} = (\lambda^{-1}_{i,j})$ ... inverse of L

$D = \mathrm{diag}(\delta_i)$ ... diagonal matrix

$$n_{i,j} = \frac{1}{i + j - 1}$$

$$\lambda_{i,j} = \left[\frac{(i-1)!}{(j-1)!}\right]^2 \frac{(2j-1)!}{(i+j-1)!(i-j)!} \quad \cdots \quad (i \geq j)$$

$$u_{i,j} = \frac{[(i-1)!(j-1)!]^2}{(i+j-1)!(j-1)!(2i-2)!} \quad \cdots \quad (i \leq j)$$

$$\lambda^{-1}_{i,j} = (-1)^{i+j}\left[\frac{(i-1)!}{(j-1)!}\right]^2 \frac{(i+j-2)!}{(i-j)!(2i-2)!} \quad \cdots \quad (i \geq j)$$

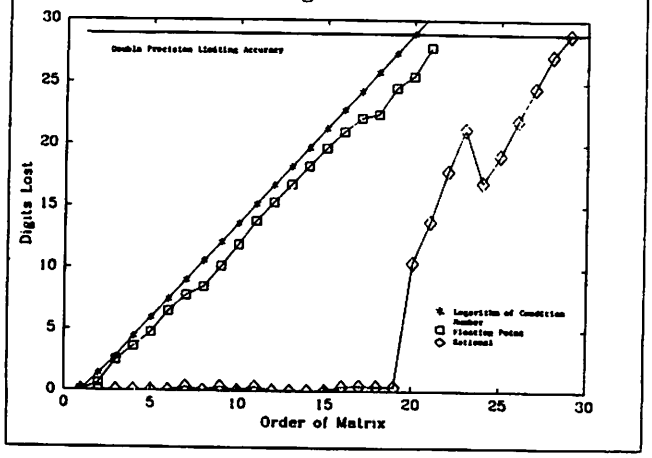
$$\delta_i = \frac{(-1)^{i-1}(n+i-1)!}{[(i-1)!]^2(n-i)!}$$

Table 1: Properties of the Order n Hilbert Matrix

In Figure 1 we present curves describing the base 10 logarithm of the condition number of the Hilbert matrices and the number of digits of accuracy lost when the inverse of the Hilbert matrices are computed using binary floating point arithmetic and simulated hyperbolic rational arithmetic. These computations were performed on a CDC 6600, the floating point computations utilize 96 bit fractions while the simulated rational computations allocates an equivalent number of bits to the storage of its fractions. Note that for Hilbert matrices of orders less than 20 the floating point computation shows a steady loss of accuracy as predicted by the graph of the base 10 logarithm of the condition number while the comparable simulated rational arithmetic computation is exact! Of course the fact that rational arithmetic computation can be exact is a property of this problem. The significant observation is that in our floating point simulation of hyperbolic rational arithmetic simple rationals must be approximated by floating point numbers inevitably introducing rounding error. The cancellation of these errors in subsequent computations is clearly a property of the bias of the rounding rule towards simple fractions. This behavior on problems with a preference for simple rational results leads to our characterization of the rounding derived from best rational approximation as an "intelligent" rounding. With this intelligent rounding we can compute inverses of Hilbert matrices eight orders higher with the same underlying 96 bit floating point representation on the CDC 6600.

Of course one might feel that for problems in which simple fractions do not play an important role that computations performed in a hyperbolic rational number system might be significantly less accurate than when performed in a floating point number system. To test this hypothesis we considered the inversion of scaled Hilbert matrices, Hilbert matrices which were scaled on the left and right by the same diagonal matrix. This diagonal scaling

Figure 1

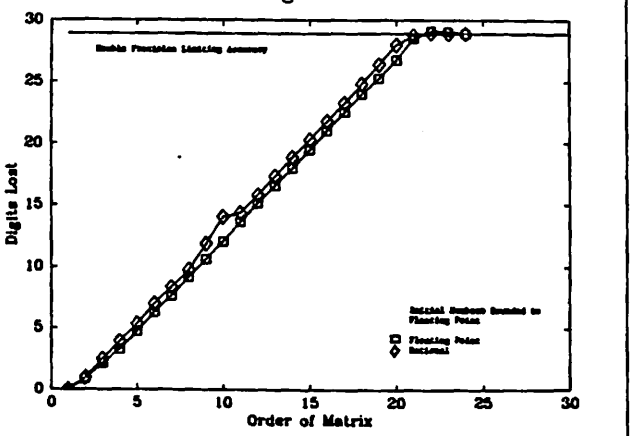


had the property that the entries of the scaled Hilbert matrix were no longer simple fractions, nor were any of the intermediate results that occured during the computation of the inverse of the scaled Hilbert matrix. In Figure 2 we present curves describing the number of digits of accuracy lost when the inverses of the scaled Hilbert matrices are computed using binary floating point arithmetic and simulated hyperbolic rational arithmetic of comparable precision. Note that both computations show the usual steady loss of accuracy as predicted by the base 10 logarithm of the condition number, with the floating point computation showing a slightly greater accuracy than enjoyed by the simulated rational computation.

From these comparisons we feel that rational number systems are viable alternatives to the traditional floating point number systems currently used by many computers. For computations in which simple fractions play an important role rational number systems can produce potentially exact results, while for computations in which simple fractions play no important role rational number systems can enjoy an accuracy comparable to that enjoyed by floating point number systems.

Let us now turn to a more precise description of hyperbolic rational number systems and of the computational experiments performed. In Section 2 we present some properties of hyperbolic rational number systems, in particular relating these properties to the well developed theory of best rational approximation. In Section 3 we describe the computation of the inverse of a Hilbert matrix and its scaled variant. In Section 4 we describe how we simulate computations in a hyperbolic rational number system using computations performed in the host computer's floating point number system.

Figure 2



2. Hyperbolic Rational Number Systems

A hyperbolic rational number system has as its machine representable numbers signed versions of the numbers in a hyperbolic chain of rational numbers and as its rounding rule the mediant rounding rule [MK80]. In this section we will describe both the hyperbolic chain of rational numbers and the mediant rounding rule. In the paragraphs that follow the letters p and q, as well as their derivatives, will denote nonnegative integers.

Recall that a rational number is merely a number that can be represented as a ratio of two integers. A fraction is just a pair of nonnegative integers (p,q) whose associated value is the rational number p/q. Without further comment we will represent the fraction (p,q) as the rational number p/q. Two important concepts related to a single fraction are the concepts of complexity and irreducibility. We say that the complexity of a fraction p/q is the integer pq. Thus we will talk about those fractions p/q with pq large as complex fractions and those fractions p/q with pq small as simple fractions. We also say that a fraction p/q is irreducible if the greatest common divisor of p and q, written gcd(p,q), is unity. Two important concepts related to pairs of fractions are the concepts of mediant and adjacency. If p/q and p'/q' are two fractions then their mediant, written med(p/q,p'/q'), is defined to be the fraction (p+p')/(q+q'). We also say that two fractions p/q and p'/q' are adjacent if the absolute value of the difference p'q-pq' is unity. Since the properties of the adjacent fractions play such an important role in the theory of best rational approximation we list some of these properties in the following:

Observation 1: Let p/q and p'/q' be adjacent fractions with p/q < p'/q'. Then
a) p/q and p'/q' are irreducible,
b) p/q and p'/q' have the same complexity only when p/q=0/1 and p'/q'=1/0,
c) p'q = pq'+1 = 1/2 + sqrt(1/4 + pqp'q'),
d) med(p/q,p'/q') is adjacent to p/q and p'/q',
e) med(p/q,p'/q') lies strictly between p/q and p'/q',
f) med(p/q,p'/q') is irreducible,
g) med(p/q,p'/q') is more complex than either p/q or p'/q',
h) med(p/q,p'/q') is the unique fraction lying strictly between p/q and p'/q' of least complexity.

The proofs of many of these observations can be found in [MK80, HW60]. The rational number system we use has as its machine numbers signed versions of the fractions in the set

$$\{\, p/q \mid p,q \geq 0,\ \gcd(p,q) = 1,\ pq \leq m \}\ .$$

Note that if we represent the fraction p/q by the point (p,q) then we find that the fractions in this set all lie under the graph of the hyperbola pq=m. Let us therefore agree to call the ordered set formed by listing the fractions in this set in increasing numeric order the hyperbolic chain of fractions H(m). From elementary number theory we find that the asymptotic cardinality of H(m), for large m, is given by

$$6m \ln(m) / \pi^2 .$$

Note that no more than $2+\log_2(m)$ bits are needed to store the numerator and denominator of any fraction in H(m).

Table 2, whose rows list the members of H(m) for m<10, was constructed by using a slight generalization of a process attributed to Farey [NZ66]. Start in row 0 by placing in order the fractions zero 0/1 and infinity 1/0. Then for each k>0 construct row k of this table by copying down in order the fractions in the row k-1, but inserting between two consecutive fractions of row k-1 their mediant if the complexity of that mediant is equal to k. A simple proof by induction on the rows of this table shows that any two consecutive fractions in row k of this table are adjacent and their mediant has complexity greater than k. We therefore conclude, using observation 1h, that row k of this table consists of the members of H(k), listed in their proper order. This leads us to the following inductively established:

n	H(n)
0	0/1 1/0
1	0/1 1/1 1/0
2	0/1 1/2 1/1 2/1 1/0
3	0/1 1/3 1/2 1/1 2/1 3/2 3/1 1/0
4	0/1 1/4 1/3 1/2 1/1 2/1 3/2 4/3 3/1 1/0
5	0/1 1/5 1/4 1/3 1/2 1/1 2/1 3/2 4/3 5/4 5/3 1/0
6	1/1 1/6 1/5 1/4 1/3 1/2 1/1 2/1 3/2 4/3 5/4 6/5 5/3 1/0
7	0/1 1/7 1/6 1/5 1/4 1/3 1/2 1/1 2/1 3/2 4/3 5/4 6/5 7/6 5/3 1/0
8	0/1 1/8 1/7 1/6 1/5 1/4 1/3 1/2 1/1 2/1 3/2 4/3 5/4 6/5 7/6 8/7 5/3 1/0
9	0/1 1/9 1/8 1/7 1/6 1/5 1/4 1/3 1/2 1/1 2/1 3/2 4/3 5/4 6/5 7/6 8/7 9/8 5/3 1/0

Table 2: Members of the Hyperbolic Chains

Observation 2: Let p/q and p'/q' be consecutive fractions in H(m) for some m>0.
a) p/q and p'/q' are adjacent,
b) p/q and p'/q' have distinct complexities,
c) med(p/q,p'/q') has complexity greater than m,
d) med(p/q,p'/q') has complexity less than 4m-1.

Recall that our rational number system has as its rounding rule P the so-called mediant rounding rule. To describe this rule and its connection with the theory of best rational approximation we must discuss some of the results from the theory of ordinary continued fractions [HW60, Kh63, MK80, NZ66, Ri81].

By repeatedly extracting the integer portion of a number and taking the reciprocal of the fractional remainder one is led quite naturally to the representation

$$x = [a_0,a_1,a_2,\ldots] = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \ldots}}$$

of a nonnegative number x as an ordinary continued fraction. Here the number a_i is called the i^{th} partial quotient of x and represents a nonnegative integer when i=0 or a positive integer when i>0. The truncated continued fractions defined by

$$p_k/q_k = [a_0,a_1,\ldots,a_k] = a_0 + \cfrac{1}{a_1 + \cfrac{1}{\ldots + \cfrac{1}{a_k}}}$$

form important fractional approximations of x called the convergents of x. From the theory of ordinary continued fractions we obtain the following:

Observation 3: The convergents of x admit the following properties.
a) $p_k/q_k = (a_k p_{k-1}+p_{k-2})/(a_k q_{k-1}+q_{k-2})$ for k≥0 if one defines $p_{-2}/q_{-2}=0/1$ and $p_{-1}/q_{-1}=1/0$,
b) Consecutive convergents of x are adjacent fractions which lie on opposite sides of x,
c) $|x-p_k/q_k| < |x-p_{k-1}/q_{k-1}|$ for k≥0,
d) If p/q satisfies the inequality $|x-p/q| < |x-p_k/q_k|$ for some k≥0, then $p_k q_k < pq$.

Note that observation 3d justifies our description of the convergents of x as best rational approximations of x.

The recursion relation described in observation 3a leads to the following algorithmically described rounding rule.

Rational Rounding Algorithm
Input: A nonnegative real number x to be rounded and an integer m.
Output: RATRND, the best rational approximation to x with complexity not exceeding m.
1. Value=x, Pold=0, Qold=1, Pnow=1, Qnow=0
2. Whole=IntegerPortion(Value)
   Pnew=Whole*Pnow+Pold, Qnew=Whole*Qnow+Qold
   Value=Value-Whole
3. If Pnew*Qnew $\leq$ m then
     Pold=Pnow, Qold=Qnow, Pnow=Pnew, Qnow=Qnew
     If Value $\neq$ 0 then
       Value = 1/Value, GoTo 2.
4. RATRND = Pnow/Qnow

We are now in a position to describe the rounding rule P used in our rational number system. Given a nonnegative real number x we define P(x) to be the value of RATRND as determined by the above rational rounding algorithm. For negative x we then use the rule P(x) = -P(-x). Some of the more important properties of this rounding rule are summarized in the following [MK80]:

Observation 4. The rational rounding rule P admits the following properties.
a) If $x \leq y$ then $P(x) \leq P(y)$,
b) $P(-x) = -P(x)$,
c) If x is a member of H(m) then $P(x) = x$.

If p/q and p'/q' are consecutive members of H(m) then it follows from observations 4a,c that there is a real number y such that P(x)=p/q when p/q<x<y and P(x)=p'/q' when y<x<p'/q'. The theory of simple continued fractions tells us that this 'splitting point' of the rounding rule P is always the mediant of p/q and p'/q'. For this reason we term the rounding rule P the mediant rounding rule. More precisely we obtain the following [MK80]:

Observation 5: Let x lie between the consecutive members p/q and p'/q' of H(m) where m>0. If p"/q" is the least complex of p/q and p'/q' then we have

$$P(x) = \begin{cases} p/q & \text{if } x < \text{med}(p/q,p'/q') \\ p'/q' & \text{if } x > \text{med}(p/q,p'/q') \\ p''/q'' & \text{if } x = \text{med}(p/q,p'/q') \end{cases}$$

When p/q is a member of H(m) the representation of p/q as a continued fraction allows us to determine the neighbors of p/q in H(m). Let p/q admit the following representation

$$p/q = [a_0,a_1,\ldots,a_k]$$

as a continued fraction with $a_k \geq 2$. Then $(p'+a'p)/(q'+a'q)$ and $(p''+a''p)/(q''+a''q)$ are the neighbors of p/q in H(m) where

$$p'/q' = [a_0,a_1,\ldots,a_{k-1}] \text{ , and}$$

$$p''/q'' = [a_0,a_1,\ldots,a_k-1]$$

with a' and a" the largest nonnegative integers for which

$$(p'+a'p)(q'+a'q) \leq m, \text{ and}$$
$$(p''+a''p)(q''+a''q) \leq m.$$

When numerical calculations are performed in this rational number system the gap between consecutive members of H(m) affects the propagation of roundoff errors. To describe the size of these gaps we proceed as follows. Let p'/q' be a finite nonzero fraction in H(m). Then p'/q' is the middle fraction in a sequence p/q, p'/q', p"/q" of three consecutive fractions in H(m) and those real numbers which round to p'/q' lie in the interval between med(p/q,p'/q') and med(p'/q',p"/q"). The relative size of this gap admits the bounds:

$$\frac{1}{b} < \frac{\text{med}(p'/q',p''/q'') - \text{med}(p/q,p'/q')}{p'/q'} < \frac{2}{b}$$

were $b = \text{sqrt}(mp'q')$. These bounds imply that the relative size of the gap associated with a given fraction in H(m) is roughly proportional to the reciprocal of the square root of the complexity of that fraction. For this reason we say the mediant rounding rule is biased towards simple fractions, i.e. fractions with small complexity.

3. Hilbert Matrix Inversion

Hilbert matrices arise quite naturally when one considers the least squares approximation of continuous functions by polynomials [FM67]. For example, let us suppose that we are asked to find the coefficients $c_1, c_2, \ldots, c_n$ of the polynomial

$$p(x) = c_1 + c_2x + c_3x^2 + \ldots + c_nx^{n-1}$$

of order n which minimizes the error

$$E = \int_0^1 | f(x) - p(x) |^2 \, dx$$

in the approximation of a continuous function f by p. Since E is a differentiable function of the coefficients of p we find that a necessary condition that E be minimized is that the partial derivatives of E with respect to c_i be zero for i=1,2,...,n. These n conditions can be succinctly written as the matrix equation

$$Hc = b$$

where

$$H = \{ \frac{1}{i+j-1} \}$$

is the order n Hilbert matrix,

$$c = (c_1, c_2, \ldots, c_n)', \text{ and}$$
$$b = (b_1, b_2, \ldots, b_n)', \text{ with}$$
$$b_i = \int_0^1 x^{i-1} f(x)\, dx ,$$

where ' denotes transpose. Since

$$c'Hc = \int_0^1 |p(x)|^2 \, dx$$

we infer that the order n Hilbert matrix is a symmetric positive definite, and therefore nonsingular, matrix.

Let A be a matrix of order n. In numerical linear algebra the following three step process is frequently recommended as an algorithm to solve the linear system Ax=b [FM67].

Solution of Ax=b.

1. Compute the triangular factorization A=LU of A using the Gaussian elimination algorithm (no pivoting.) Here L is a unit lower triangular matrix and U is an upper triangular matrix.
2. Solve the lower triangular system Ly=b for y using the forward elimination algorithm.
3. Solve the upper triangular system Ux=y for x using the backward substitution algorithm.

Wilkinson has shown that when the effect of roundoff error is considered the above algorithm produces not the exact solution x of Ax=b but rather the exact solution x^* of the linear system $(A-S)x^*=b$, the matrix S accounting for the effect of the roundoff error that accumulates during the solution algorithm [Wi65]. When the matrix A is a symmetric positive definite matrix, e.g. a Hilbert matrix, it has been shown that this algorithm is stable in the sense that the entries of S have a size that is comparable to the size of the error which is incurred when the entries of A and b are rounded to machine numbers. To assess how accurate an approximation x^* is of x let us follow Turing's analysis in which we suppose that S is a matrix of random numbers, the numbers being uncorrelated but having the same variance. To first order in S we find that $x^*=x+A^{-1}Sx$, and so it follows that

$$\frac{\text{RMS error in } x}{\text{RMS size of } x} = \frac{1}{n} N(A)N(A^{-1}) \frac{\text{RMS size of } S}{\text{RMS size of } A}$$

where RMS denotes root mean square and, for example,

$$N(A) = \text{sqrt}(\text{trace}(A'A))/n ,$$
$$\text{RMS size of } x = \text{sqrt}(x'x/n), \text{ and}$$
$$\text{RMS size of } S = \text{sqrt}(\text{trace}(E[S'S]))/n$$

where E[.] denotes the expected value function. If

$$M(A) = \max_{i,j} \{ |a_{i,j}| \}$$

then

$$\frac{1}{n} N(A)N(A^{-1}) \leq n\, M(A)\, M(A^{-1}) .$$

Turing suggested that the easily computed number $nM(A)M(A^{-1})$ be used as a measure of the conditioning of the problem of solving Ax=b [Tu48]. We will call this number Turing's maximum element condition number of A, or simply the condition number of A. Note that this condition number is a statistical quantity which estimates how an initial error in A is amplified into a final error in the computed solution of Ax=b [FM67, Wi65]. Therefore, the base 10 logarithm of the condition number can be viewed as a statistical estimate of the number of digits of accuracy that could be lost during the computation of the solution of Ax=b. Note that we use the maximum element condition number rather than other condition numbers since rigorous estimates of the maximum element condition number of Hilbert matrices exist [To54].

When the identity

$$AA^{-1} = I$$

is viewed as a collection of n linear systems, one linear system for each column of the order n identity matrix I, one arrives at the following frequently recommended algorithm for computing the inverse of A [FM67].

Inversion Procedure

1. Compute the triangular factorization A=LU of A using the Gaussian elimination algorithm (no pivoting.)
2. Solve, on a column by column basis, the linear system $LL^{-1}=I$ for the inverse of L using the forward elimination algorithm.
3. Solve, on a column by column basis, the linear system $UA^{-1}=L^{-1}$ for the inverse of A using the backward substitution algorithm.

When A is a symmetric positive definite matrix, e.g. a Hilbert matrix, we know from our previous comments that this is a stable algorithm for computing the inverse of A [Wi65]. Therefore the only way this algorithm can fail to compute an accurate estimate of the inverse of A is for A to be ill-conditioned, i.e. for the condition number of A to be large.

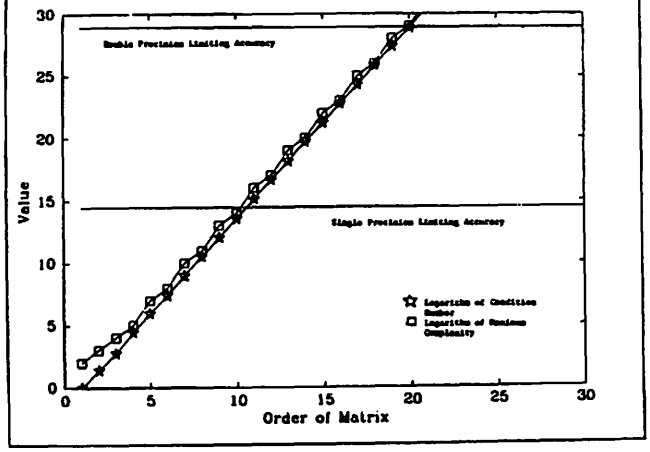
Several comments must be made before we describe the results of our computations on inverting Hilbert or scaled Hilbert matrices. The first comment concerns how these computations were performed and the second comment concerns the errors which are reported. All computations were performed on a CDC 6600 using the University of Minnesota's FORTRAN 77 compiler. This version of the FORTRAN language supports both single and double precision binary floating point computations in which 48 and 96 bits respectively are allocated to the storage of the normalized binary fraction. To make computations in the hyperbolic rational number system of comparable precision we chose to limit the complexity of our fractions to be less than either 2^{48} or 2^{96}. From the statements made in the previous section we recognize that this choice of complexities means that approximately 48 or 96 bits are needed to store both the numerators and denominators of the fractions in the rational number system and that the relative size of the smallest gaps in both number systems are of comparable size. During the computation of the inverse of a Hilbert or scaled Hilbert matrix A we monitored, using the formulas in Table 1, the maximum relative error of the nonzero entries in each of the matrices L, U, L^{-1}, and A^{-1} and found that A^{-1} was always the matrix which had the entry with largest relative error. Therefore, in the figures described below the error we are reporting is always the value of

$$\text{Rerr} = \max_{i,j} \left| \frac{(\text{exact } A^{-1})_{i,j} - (\text{computed } A^{-1})_{i,j}}{(\text{exact } A^{-1})_{i,j}} \right|$$

From Rerr we computed the number of digits of accuracy lost by computing the base 10 logarithm of either 2^{48}Rerr or 2^{96}Rerr depending on which precision was used during the computations.

As stated in the introduction the inversion of a Hilbert matrix is interesting for two reasons. The first reason concerns the fact that the inversion of a Hilbert matrix is an inherently ill-conditioned problem and the second reason concerns the fact that the numbers which arise during the computation of the inverse of a Hilbert matrix can be described using only simple fractions. In Figure 3 we illustrate these facts by plotting two curves. The first curve is a plot of

Figure 3



the base 10 logarithm of the Turing condition number of the Hilbert matrices, note that this curve ascends almost linearly with a slope of about 1.53 [FM67, GK78, To54]. The second curve is a plot of the base 10 logarithm of the maximum complexity of the fractions which arise during the computation of the Hilbert matrices, this maximum complexity was computed using a FORTRAN package which allows one to perform exact rational arithmetic. It is somewhat surprising to observe the rather close correlation between the statistical condition number and the maximum complexity involved in the computations.

Let us now consider the computation of the inverses of Hilbert matrices. In Figure 4 we display the growth of the error when the inverses of the Hilbert matrices are computed in single and double precision in both the binary floating point and simulated hyperbolic rational number systems. We emphasize that the results of the rational computation in Figure 4 are derived from a simulation using floating point approximations and floating point arithmetic as a host, and are not the straightforward exact results of exact rational computation. The methodology of the simulation is described in Section 4.

Several conclusions can be drawn from this figure. The first conclusion is that the base 10 logarithm of the condition number is a reasonable estimate of the number of digits of accuracy that is lost in the floating point computation. The second conclusion is that the number of digits of accuracy that is lost in the floating point computation seems not to depend significantly on the underlying precision of the floating point number system used in the computations. Based on this conclusion our later computations involving the inverse of scaled Hilbert matrices will be done using only double

Figure 4

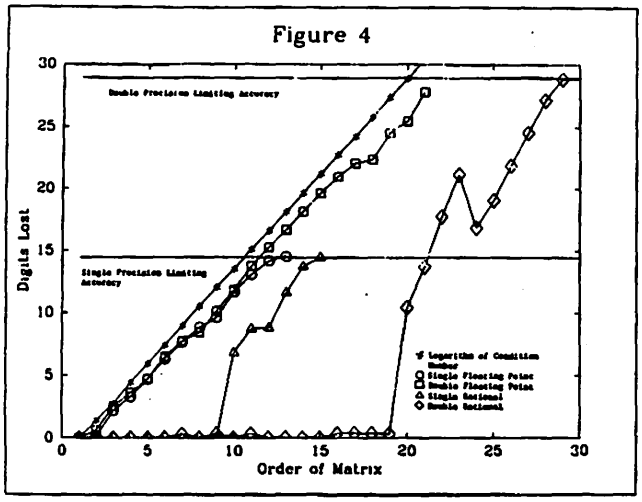
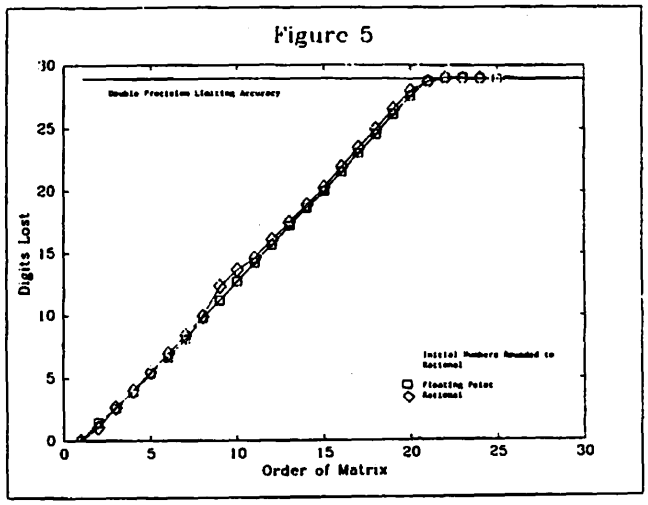


Figure 5



precision arithmetic. The third conclusion is that the plot of the maximum complexity of the fractions which arise during the computation of the inverses of the Hilbert matrices allows one to accurately estimate at what order Hilbert matrix the computations in the rational number system fail to be exact. It is interesting to note that when the computations in the rational number system are first inexact the number of digits of accuracy lost appears to be approximately one-half of the maximum number of digits of accuracy possible. This leads us to believe that the first number that is inaccurately computed in the rational number system is a number which is rounded to a simple fraction since the relative size of the gap which rounds to a simple fraction is proportional to the reciprocal of the square root of maximum complexity allowed in the rational number system. The fourth conclusion is that when the numbers which arise during a computation can be described by simple fractions the rational number system is likely to produce more accurate answers than the usual floating point number system.

For problems in which simple fractions do not play an important role one might believe that computations performed in a hyperbolic rational number system might be significantly less accurate than when performed in a floating point number system. To test this hypothesis we considered the computation of inverses of scaled versions of Hilbert matrices, scaled versions of the form DHD where D is a diagonal matrix and H is a Hilbert matrix. To construct a typical diagonal matrix D we first chose a 108 binary digit random number scaled to lie in the interval (0,1) and then took as the i^{th} diagonal entry of D the i^{th} root of that random number. Note that if L and U are the triangular factors of the Hilbert matrix H then DLD^{-1} and DUD are the corresponding triangular factors of the scaled Hilbert matrix DHD. In Figures 2 and 5 we display the number of digits of accuracy lost in the computation of the inverses of the scaled Hilbert matrices when the entries of the scaled Hilbert matrix were initially rounded so as to be machine numbers in the floating point and rational number systems respectively. Note that for each order matrix the number of digits of accuracy lost represents the worst case loss in accuracy over a sample of 25 different scaled Hilbert matrices. From these figures we conclude that even for problems in which simple fractions do not play an important role a comparable loss in accuracy occurs in both the floating point and rational number systems, with the computations in the floating point number system enjoying a slightly greater accuracy.

4. Hyperbolic Rational Number System Simulation

In our simulation of a hyperbolic rational number system we choose to represent the rational number p/q as the floating point number obtained by dividing p by q, and we choose to implement the rounding rule by writing a function subprogram R(x) which accepts as input a floating point number x and returns as output the appropriately signed value of RATRND as determined by the floating point implementation of the rational rounding algorithm described in Section 2. Thus if the floating point numbers x and x' represent the rational numbers p/q and p'/q' respectively, then we will use the floating point value of R(x+x') as the value of the rational number P(p/q+p'/q').

In Table 3 we illustrate how one can use this function subprogram R(.) to simulate computations in the rational number system by listing the algorithm used to solve the linear system Ax=b as described in Section 3. Note that every floating point number is rounded by the function subprogram R(.) before it is used in the next computation. We emphasize that the floating point arithmetic of the host computer is used to simulate the computations

performed in the hyperbolic rational number system. During a sequence of operations the result of one operation is rounded using R(.) before it is used in the next floating point operation. Thus if some rounding errors have accumulated, but the exact result would have been some simple rational number, then the bias of mediant rounding towards simple rational numbers can allow the computation to produce the correct simple rational number as its result [MK79].

We would like to stress that our simulation of the hyperbolic number system is used only to determine the potential advantages and disadvantages of using this form of rational arithmetic. We do not suggest that our simulation is efficient, estimates of the average running time of R(.) can be obtained from Knuth's [Kn69] analysis of Euclid's algorithm. An efficient arithmetic unit to realize this rational arithmetic has been described in [KM83].

For further details on the properties of several rational number systems we refer the reader to [MK80, MK83, PV84].

```
Input:    A ...  the order n coefficient matrix, and
          b ...  the order n inhomogenous vector.
Output:   b ...  the order n solution vector x of Ax = b
Comment:  The values of A and b specified on input are destroyed
          during the calculation.
Algorithm:
  1. Determine the triangular factorization A = LU of A using the
     Gaussian elimination algorithm (no pivoting).
          1.1  For k = 1 upto n-1 do
          1.2      For i = k upto n do
          1.3          a(i,k) = - R(a(i,k)/a(k,k))
          1.4          For j = k+1 upto n do
          1.5              a(i,j) = R(a(i,j)+R(a(i,k)*a(k,j)))
          1.6          Next j
          1.7      Next i
          1.8  Next k
  2. Solve the lower triangular system Ly = b using the forward
     elimination algorithm.
          2.1  For i = 2 upto n do
          2.2      Sum = 0
          2.3      For j = 1 upto i-1 do
          2.4          Sum = R(Sum+R(a(i,j)*b(j)))
          2.5      Next j
          2.6      b(i) = R(b(i)-Sum)
          2.7  Next i
  3. Solve the upper triangular system Ux = y using the backward
     elimination algorithm.
          3.1  b(n) = R(b(n)/a(n,n))
          3.2  For i = n-1 downto 1 do
          3.3      Sum = 0
          3.4      For j = i+1 upto n do
          3.5          Sum = R(Sum+R(a(i,j)*b(j)))
          3.6      Next j
          3.7      b(i) = R(R(b(i)-Sum)/a(i,i))
          3.8  Next i
```

Table 3: Algorithm for Solving Ax = b Using Rational Arithmetic

References

[Ch83] Man-Duen Choi; "Tricks or Treats with the Hilbert Matrix", Amer. Math. Monthly, 90 (1983), pp. 301-312.

[Co39] A. R. Collar; "On the Reciprocation of Certain Matrices", Proc. Roy. Soc. Edin., 59 (1939), pp. 195-206.

[FM67] George Forsythe and Cleve B. Moler; "Computer Solution of Linear Algebraic Systems", Prentice-Hall (1967).

[GK78] Robert T. Gregory and David L. Karney; "A Collection of Matrices for Testing Computational Algorithms," Krieger (1978).

[HW60] G. H. Hardy and E. M. Wright; "An Introduction to the Theory of Numbers", 4th ed., Clarendon Press (1960).

[Kh63] A. Ya. Khintchin; "Continued Fractions" (Translated from the Russian by P. Wynn), P. Noordhoff Ltd. (1963).

[KM83] P. Kornerup and D. W. Matula, "Finite Precision Rational Arithmetic: An Arithmetic Unit", IEEE TC, C-32 (1983), pp. 378-387.

[Kn69] D. Knuth, "The Art of Computer Programming", vol. 2, "Seminumerical-Algorithms", Addison-Wesley (1969).

[MK79] D. W. Matula and P. Kornerup; "Approximate Rational Arithmetic Systems: Analysis of Recovery of Simple Fractions During Expression Evaluation", Springer-Verlag Lecture Notes in Computer Science, 72 (1979), pp. 383-397.

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

[MK85] D. W. Matula and P. Kornerup; "Finite Precision Rational Arithmetic: Slash Number Systems", IEEE TC, C-34 (1985), pp. 3-18.

[NZ66] Ivan Niven and Herbert S. Zuckerman; "An Introduction to the Theory of Numbers", 2nd ed., Wiley (1966).

[PV84] R. Pyzalski and M. Vala; "Conversion of Decimal Numbers of Irreducible Rational Fractions", submitted , (1984).

[Ri81] Ian Richards; "Continued Fractions Without Tears", Mathematics Magazine, 54 (1981), pp. 163-171.

[To54] John Todd; "The Condition of the Finite Segments of the Hilbert Matrix", National Bureau of Standards Applied Mathematics Series, 39 (1954), pp. 109-116.

[Tu48] A. M. Turing; "Rounding-Off Errors in Matrix Processes", Quarterly J. Mech. Ap. Math., 1 (1948), pp. 287-308.

[Wi65] J. H. Wilkinson; "The Algebraic Eigenvalue Problem", Clarendon (1965).

This work was sponsored by the United States Army under Contract No. DAAG29-80-C-0041, the Department of Energy under Contract No. DE-AL02-82ER12046A000, and the National Science Foundation under grant DCR-8315289.