MATRIX PROCESSORS USING p-ADIC ARITHMETIC FOR EXACT LINEAR COMPUTATIONS

By
E. V. KRISHNAMURTHY
Department of Computer Science
University of Lagos, Lagos, Nigeria

SUMMARY

A unique code (called Hensel's code) is derived for a rational number, by truncating its infinite p-adic expansion. The four basic arithmetic algorithms for these codes are described and their application to rational matrix computations is demonstrated by solving a system of linear equations exactly, using the Gaussian elimination procedure.

A comparative study of the computational complexity involved in this arithmetic and the multiple prime module arithmetic is made with reference to matrix computations. On this basis, a multiple p-adic scheme is suggested for the design of a highly parallel matrix processor.

1. INTRODUCTION

Matrix computations such as solving a system of linear equations, finding the inverse or generalized inverse of a matrix or reduction of a matrix to a specified canonical form or the determination of the characteristic equation are of great importance in a variety of scientific applications. It is well known that when conventional p-ary or floating point arithmetic is used for such computations, the cumulative round-off errors make the results totally unreliable. If the matrix contains only rational entries there are two approaches to obviate such errors:

a. Use of rational arithmetic
b. Use of residue or modulo arithmetic.

The use of rational arithmetic, Knuth[1], though exact, is very expensive and laborious - each rational add/subtract operation requires three multiplications, one add/subtract and a reduction of the resulting fraction to its lowest form (by dividing both the numerator and denominator by the greatest common divisor); rational multiplication/division operations involve two multiplications and a reduction of the result to its lowest form. Therefore residue or modular arithmetic approach has been suggested; young and Gregory[2]; Knuth[1]; Rao et al[3]. In the latter approach, every rational number a/b ($b \neq 0$, $0 \leq a \leq p-1$, $0 < b \leq p-1$) is uniquely identified with $a \cdot b^{-1}$, where the binary operation is the product of modulo, a prime number p and b^{-1} the multiplicative inverse of b in the Galois field formed with elements 0,1,2, ... (p-1) under the two binary operations "add modulo p" and "multiply modulo p". Naturally, the magnitude of p restricts the range of uniquely representable rationals.

It is easily seen that the computational complexity increases with increasing p; for instance, the multiplication complexity is of the order

$$O\left(\frac{k(k+1)}{2}\right)$$

where k is the precision or number of binary digits in p; the computation of multiplicative inverse (and hence the division) over GF(p) requires the use of Euclidean algorithm similar to that for G.C.D. involving on an average $6 \log_2 p (=6k)$ k - precision divisions and $12k^2$ k-precision multiplications, each of complexity $O(k^2)$; see Knuth[1], p.333. To reduce the complexity, therefore, the use of multiple prime modulii has been suggested (see Young and Gregory[2], Rao et al[3]). Here a modulus M (of precision k) which is a product of the distinct primes $p_1, p_2, \ldots, p_s$ (each approximately of precision $\ell_i = \ell$ binary digits such that $k = s\ell$) is chosen or

$$M = \prod_{i=1}^{s} p_i$$

Every integer a is then represented in terms of a set of residues $(r_1, r_2, r_3, \ldots, r_s)$ with respect to each p_i. Every rational number a/b is then identified with $a \cdot b^{-1}$ where b^{-1} is unique and $b \cdot b^{-1}$ has a representation (1, 1, 1, ... 1) and $b \cdot b^{-1} \equiv 1 \bmod M$.

If multiple prime moduli arithmetic is used, the matrix computations are to be performed for each p_i separately and the results are to be later combined (using either the Chinese Remainder theorem or by the simpler and faster symmetric mixed radix procedure). For this approach to be faster (Knuth[1], p.252) each prime p_i should be of the form $2^{\ell_i} - 1$; then the addition and multiplication modulo p_i, as well as the conversion to positional representation turns out to be simpler and faster. (Note that $2^{\ell_i} - 1$ has to be a prime if multiplicative inverse has to exist). In such a case the following advantages result:

a. The arithmetic with respect to modulo p_i is of a lower precision ($\ell_i = \ell$) and s multiplications each of complexity $O(\ell^2)$ are to be performed rather than $O(k^2) = O(s^2\ell^2)$.

b. The computation of multiplicative inverse is less complex; it involves $6s\ell$ divisions each of precision ℓ and complexity $O(\ell^2)$ and $12s\ell$ multiplications each of precision ℓ and complexity $O(\ell^2)$; the division requires one more multiplication by the inverse or s multiplications each of complexity $O(\ell^2)$.

c. Multiple modulii method allows many arithmetic operations to take place simultaneously, so that a substantial savings of execution time can be obtained. This decrease in execution time, as well as the possibility of designing highly parallel

computers, make this method significantly important, particularly for real time calculations.

However, the following disadvantages still exist:

a. When the range of rationals required depends upon the matrix problem in hand, the choice of primes of the form $2^{\ell_i} \pm 1$ becomes difficult; accordingly the choice of M (product of primes) cannot be enlarged systematically to cope up with the hardware design.

b. If the algorithms involve a large number of divisions the complexity increases.

c. The conversion into modular form requires a certain amount of operations of complexity $O(k)$.

d. The conversion back to positional notation require the use of high precision multiplications; hence, if the output consists of a large number of entries, as in matrix inversion, the complexity increases.

These difficulties motivated us to look for an alternative arithmetic system which is also modular in nature, but has the simplicity of arithmetic operations as in the p-ary system, yet free from round-off errors and is simple for hardware realization. The segmented p-adic arithmetic system, which is equivalent to modulo p^r (where p is a prime) arithmetic has all these attractive features; it is surprising that the potentialities of the 2-adic arithmetic system have not been explored. The following are some of the attractive features:

a. The conversion of integers and other rationals into 2-adic representation is simple and realizable by existing hardware.

b. The evaluation of multiplicative inverse is of many orders less in complexity; in fact it is simpler than binary division and has a complexity $O\left(k\left(\frac{k+1}{2}\right)\right)$ for k bit numbers.

c. The fast high-precision multiplication algorithm due to Toom-Cook and Schonhage-Strassen (See Knuth[1], Borodin and Munro[4]) or other high-speed switching schemes could be readily used for speeding up the 2-adic arithmetic, when very large number of bits are involved.

d. The conversion from modulo 2^r to the positional representation is trivial, based on the nature of the algorithm.

e. The choice of r can be made in regular integral steps to meet the requirement of the specific matrix problem in hand. Accordingly the hardware design becomes simple.

A detailed introduction to the segmented p-adic arithmetic system is available elsewhere, Krishnamurthy et al[5]; however, for convenience we will briefly describe in the next section some of the important properties and the basic arithmetic algorithms in this system.

2. PROPERTIES OF H(p,r) CODES AND ARITHMETIC

By the H(p,r) code of the rational number α, we mean that finite segment of the infinite p-adic expansion of α, consisting of the first r-digits, the radix point occupying the same position as in the infinite expansion; (Bachmann[6], Borevich and Shafarevich[7]):

i. Let $a_{-m} \ldots a_0 \, a_1 \ldots a_n \ldots$ be the infinite expansion of α; then the finite segment $a_{-m} \, a_{-n+1} \ldots a_0 \ldots a_n$ where $m+n+1 = r$ is called the Hensel code of α denoted by $H(p,r,\alpha)$.

ii. Given $\alpha = \frac{a}{b} = \frac{c}{d} \, p^m$, $b \neq 0$ and GCD $(p,d) = 1$ and GCD $(p,c) = 1$, $H(p,r,c/d)$ is the p-ary representation of the integer $(c \cdot d^{-1})$ modulo p^r, where d^{-1} is the multiplicative inverse of d modulo p^r (for computing d^{-1} see (v) below); note that the p-ary representation is written for convenience, in the reverse order, with ascending powers from left to right.

For convenience $H(p,r,\alpha)$ is denoted as an ordered pair in the mantissa exponent form thus: (m_α, e_α).

Since we keep the length of $H(p,r,\alpha)$ constant (r digits), we denote only the negative exponents in e_α. When $e_\alpha = -m$, the radix point is placed m digits to the right of the left most digit.

Example:

a. $H(5,4,\frac{7}{15})$ is obtained thus:

$\alpha = 7/15 = 7/3 \cdot 5^{-1}$.

Thus $m_\alpha = (7.3^{-1}) \mod 625 = (7.417) \mod 625 = 419 = .4313$

$e_\alpha = -1$.

b. $H(5,4,15/7) = (m_\alpha, e_\alpha)$; $\alpha = 15/7$ where $m_\alpha = .0402$, $e_\alpha = 0$.

c. $H(5,4,15) = (m_\alpha, e_\alpha)$; $\alpha = 15$ where $m_\alpha = .0300$, $e_\alpha = 0$.

iii. Let $N = \left\lfloor \sqrt{\frac{p^r - 1}{2}} \right\rfloor$ where $\lfloor \; \rfloor$ denotes the lower integral part. Then every rational number $\alpha = a/b$ such that $0 \leq |a| \leq N$ and $0 < b \leq N$ has a unique $H(p,r,\alpha)$.

iv. The $H(p,r,\alpha)$ codes are closed with respect to basic arithmetic operations; or in other words, $H(p,r,\alpha * \beta) = H(p,r,\alpha) * H(p,r,\beta)$: (* denotes ADD/SUBTRACT/MULTIPLY/OR DIVIDE) provided α, β and $\alpha * \beta$ satisfy the range condition (iii).

v. Given $H(p,r,\alpha)$, the code $H(p,r,-\alpha)$ is obtained by a radix complement operation. In view of this, multiplication and division can be performed without corrections that are required in positive p-ary complement number representation. In this sense, it resembles the polarization algorithm in negative base; Krishnamurthy et al[5].

The complementation algorithm is exactly similar to the true or radix-complement of positive p-ary numbers, except that it is taken from the lower index position.

Let $\alpha = a_{-m} \, a_{-m+1} \cdots a_{-1} \cdot a_0 \cdots a_n$

then its complement

$\bar{\alpha} = -\alpha = b_{-m} \, b_{-m+1} \cdots b_{-1} \cdot b_0 \cdots b_n$

is obtained thus:

Rule 1: If $a_i \neq 0$ for $-m \leqslant i \leqslant n$

then $b_i = p - a_i$ for $i = -m$

and $b_i = (p-1)-a_i$ for $-m+1 \leqslant i \leqslant n$.

Rule 2: If $a_i = 0$ for $-m \leqslant i \leqslant j$

then $b_i = 0$ for $-m \leqslant i \leqslant j$

and $b_{j+1} = p - a_{j+1}$

$b_i = (p-1) - a_i$ for $j+2 \leqslant i \leqslant n$.

vi. The arithmetic operations using H(p,r) codes is almost identical with p-ary arithmetic, since it is essentially modulo p^r arithmetic realized as a simple recursion of modulo p operations.

All these arithmetic algorithms, including the division, proceeds from the lower index to the higher index position. In particular, the division (and reciprocal) is deterministic, free from trial-error involved in p-ary arithmetic for quotient determination.

Let

$$\alpha = a_{-m} \cdots a_{-1} \cdot a_0 a_1 \cdots a_n$$

$$\beta = b_{-m} \cdots b_{-1} \cdot b_0 b_1 \cdots b_n$$

be the H(p,r) codes of rationals α and β consisting of $r = m+n+1$ digits.

a. Addition/Subtraction:

The algorithm for addition aligns the p-adic point and finds the sum digit s_i and the carry digit c_{i+1} from a knowledge of a_i, b_i and c_i.

Thus $s_i = (a_i + b_i + c_i) \bmod p$

for $i = -m, -m+1, \ldots n$

$c_{i+1} = 1$ if $a_i+b_i+c_i \geqslant p$

$= 0$ otherwise;

$c_{-m} = 0$ and ignore c_{n+1}.

Subtraction is realized as a complemented addition.

b. Multiplication:

This is similar to the p-ary multiplication, except that the product is developed to only lower r digits (mod p^r) (and hence has a complexity $O\left(\frac{r(r+1)}{2}\right)$). The algorithm consists in forming the cross-products:

$$P_{ij} = b_i a_j \quad \text{for} \quad -m \leqslant i \leqslant n$$

and $j = -m, -n+1, \ldots, n$

Then the partial product

$$P_i = \sum_{j=-m}^{n} P_{ij} \cdot \Delta(m+j)$$

where $\Delta(x)$ denotes a right shift by x digits.

Example: $H(5,4,\frac{1}{4}) \cdot H(5,4,\frac{1}{3}) = (.4333,0) \cdot (.2333,0) = (.3424,0) = H(5,4,\ 1/12)$

c. Multiplicative inverse and division:

Given $0 < b \leqslant p^r-1$ and GCD $(p,b) = 1$, $b^{-1} \bmod p^r$ can be obtained very simply by a recursive solution of the congruences with respect to p.

Let $b = .b_0\ b_1 \ldots\ldots\ b_{r-1}$, $(b_0 \neq 0)$

and $b^{-1} = .q_0\ q_1 \ldots\ldots\ q_{n-1}$

The q_i's can be obtained by solving for q_i in

$$b \sum_{i=0}^{r-1} q_i p^i \equiv 1 \bmod p^r$$

Thus starting with $q_0 = b_0^{-1} \bmod p$, each q_k $(k \geqslant 1)$ is computed by solving for

$$(q_k p^k + \sum_{i=0}^{k-1} q_i p^i) b \equiv 1 \bmod p^{k+1}$$

This leads to the following deterministic trial-error-free division algorithm for obtaining the quotient, digit by digit, proceeding from the lower index to the higher index position.

The following is the algorithm for finding $a.b^{-1}$:

Let R_0 = Zeroth partial remainder or initial numerator a (= 1 for finding b^{-1})

R_i = ith partial remainder

R_{ii} = ith positional digit of R_i

Then $q_i = R_{ii} \cdot b_0^{-1} \bmod p$

for $i = 0,1,2, \ldots (r-1)$

and $R_{i+1} = R_i - q_i\ b\ \Delta(i)$

where $\Delta(i)$ = right shift by i digits.

Note that this algorithm can be applied for any numerator; by setting $R_{00} = 1$, and all other digits of R_0 to zero, one can obtain the multiplicative inverse of b. This algorithm has a complexity $O\left(\frac{r(r+1)}{2}\right)$.

Example:

$H(5,4,\ 8/9) \ / \ H(5,4,\ \frac{1}{2})$

$= (.2243,0) \ / \ (.3222,0) = (.4432,0)$

$= H(5,4,\ 16/9)$.

Remark: If $b_0 = 0$, shift the divisor left keeping count until the first digit is non-zero and suitably adjust the exponent. It is assumed that by shifting the divisor, it does not go outside the range for the given $H(p,r,\alpha)$.

vii. It has been shown in Krishnamurthy et al[5] that every rational number a/b in the range $0 \leqslant a \leqslant N$ and $0 < b \leqslant N$

where $N = \left\lfloor \sqrt{\frac{p^r-1}{2}} \right\rfloor$ (see iii)

has a unique H (p,r,α). This permits us to convert these into rationals by a diophantine solution of two unknowns and amount to a brute-force multiplication by 1,2,.....b to obtain an integer. Since this method is tedious we will describe an alternative method which is applicable to particular algorithms in matrix computations.

3. CONVERSION OF H(p,r) CODES TO RATIONALS

The conversion is based on the following simple principle:

As every rational number a/b ($b \neq 0$, $0 \leq a \leq p^r - 1$, $0 < b \leq p^r - 1$) is represented in the form $(a \cdot b^{-1}) \bmod p^r$ it is possible to determine a as well as b if some common multiple of all the denominators involved in a given algorithm is known. Since any algorithm consists of a predetermined sequence of arithmetic operations, it is possible to derive the arithmetic expression for this common multiple and this facilitates the conversion.

For instance, if we assume that k b is known, then the national number corresponding to $(a \cdot b^{-1}) \bmod p^r$ is

$$\left[1/_{kb} \cdot (a\, b^{-1} \cdot k\, b)\right] \bmod p^r.$$

Therefore as long as $(a\, b^{-1} \cdot k\, b)$ and k b are representable uniquely as H(p,r) codes the conversion is straightforward.

For this purpose, we introduce the following definitions:

i. For each integer a, $0 \leq a \leq p^r - 1$

$$\text{VALUE}(a) = a \text{ if } 0 \leq a \leq (p^r - 1)/2$$
$$= a - p^r, \text{ otherwise.}$$

ii. Let m_α be the mantissa and e_α be the exponent of a p-adic number $\alpha = (a/b)$. Then let

$$I(m_\alpha) = \sum_{i=0}^{r-1} a_i p^i$$

iii. Let $\phi = k \cdot b$ be the integral multiple of b. Since ϕ is an integer its H(p,r) code will have $e_\phi = 0$.

iv. Let $\gamma = \alpha \cdot \phi$ and $H(p,r,\gamma) = (m_\gamma, e_\gamma)$

where $m_\gamma = (m_\alpha \cdot m_\phi) \bmod p^r$ and $e_\gamma = e_\alpha$

Then the rational equivalent of H(p,r, a/b) is got from

$$\frac{1}{p^{e_\alpha}} \quad \frac{\text{VALUE } I(m_\alpha)}{\text{VALUE } I(m_\phi)}$$

Thus given the $H(p,r,\alpha)$ code the rational α can be obtained provided the common multiple of all the denominators is available. Let us call this common denominator as COMDEN.

An expression for COMDEN can always be derived in advance and COMDEN can be computed in parallel for all the algorithms whose inputs are rational numbers. The computation of COMDEN is simplified by using the following procedure:

i. Rescale all the input entries into integers, keeping the scale factor for use at the end.

ii. Convert the entries into p-adic form.

iii. Perform all necessary arithmetic operations of the algorithm.

iv. Compute in parallel COMDEN.

v. Convert the output entries to rationals using:

$$\alpha = \frac{1}{p^{e_\alpha}} \frac{\text{VALUE } (I(m_c \cdot m_\alpha) \bmod p^r)}{\text{VALUE } (I(m_c))}$$

where $H(p,r,\alpha) = (m_\alpha, e_\alpha)$

and COMDEN $= (m_c, e_c)$, $e_c = 0$.

4. DESIGN OF MATRIX PROCESSORS

i. Design considerations:

In using p-adic arithmetic for matrix computations the important decisions to be made are:

(a) The range N of rationals numbers required; accordingly the choice of p and r are to be made.

(b) COMDEN for the algorithm.

Since expressions for these can be obtained, Rao et al[3], these are easily computed.

We already saw that the arithmetic unit for 2-adic arithmetic is identical to that for binary arithmetic, but for a very minor change required for the division operation. Therefore it is quite straightforward to design a matrix processor which would use various algorithms.

Since the arithmetic unit can be made to function for either 2-adic or binary, the final results can be obtained in floating-point binary or decimal form.

ii. Complexity of p-adic computation:

It is now necessary to compare theoretically the complexity involved in using the 2-adic arithmetic with that in using the multiple prime moduli arithmetic, assuming that right choices of primes of the form

$$2^{\ell_i} \pm 1$$

of nearly same precision $\ell_i = \ell$ are available. (For an up to date review of computational complexity see Borodin and Munro[4]);

It is well known that (Berezin and Zhidkov[8]; Faddeev and Faddeva[9]) that most algorithm for solving linear equations or for generalized or conventional matrix inversion involve $O(N^3)$ multiplications, $O(N^3)$ additions and $O(N)$ divisions, where (NxN) is the order of the matrix. However, the computation of characteristic equation by reduction to Frobenius form involves $O(N^3)$ multiplications, $O(N^3)$ additions and $O(N^2)$ divisions. Therefore the $\prod_i p_i$ and 2^r arithmetic will have the following complexities (Table 1) for matrix inversion:

Operations	$\prod_{i=1}^{s} p_i$	2^r
Input Conversion	$N^2 s^2 \ell$	nil
Multiplications	$N^3 s \ell^2$	$N^3 s^2 \ell^2/2$
Divisions	$18Ns\ell^3 + Ns\ell^2$	$N s^2 \ell^2/2$
Output Conversion	$N^2 s^2 \ell^2$	nil

TABLE 1

Since the N^3 term is dominant for larger N, the multiple prime moduli arithmetic seems to be lower in complexity compared to the p-adic arithmetic which requires $s/2$ times more multiplications (s = number of primes chosen). This gives us an impression that as s increases the gain increases. In practice, however, this gain will not result since the choice of many primes of the form

$$2^{\ell_i} \pm 1$$

becomes difficult by keeping $\ell_i = \ell$ (a constant); also other operations and the overhead of the recursions and indexing which have not been taken into account add to the complexity. If the number of divisions increase, as well as the number of output entries are large the conversions might considerably add to the complexity.

In spite of all these, the overwhelming advantage of the multiple prime-moduli arithmetic for realizing a highly parallel computation system (in which all the computations can be done simultaneously thereby reducing the execution time) is to be kept in mind. At the same time the fact that 2-adic arithmetic is simple and economical for hardware design is also to be given due consideration. Therefore a hybrid scheme which uses multiple p-adic arithmetic will have excellent potentialities.

As an example, consider the choice of primes $2, 2^2 - 1, 2^2 + 1, 2^3 - 1$ viz. 2, 3, 5, 7 and the modulo $p_i^{r_i}$ arithmetic with respect to each one of them. It is well known that arithmetic modulo $2^2 - 1$, $2^2 + 1$, $2^3 - 1$, are realized by simple modifications of the binary arithmetic. Since 2^{r_1}, 3^{r_2}, 5^{r_3} and 7^{r_4} are all relatively prime, the chinese remainder theorem or related procedures such as symmetric mixed radix procedure can be applied for conversion, (Note, particularly the advantage when H(2,r) and H(5,r) are used; in this case, the conversion to decimal is straightforward), as shown below:

$$\text{Let } M = \prod_{i=1}^{s} p_i^{r_i};$$

then if we know $a \equiv a_i$ modulo $p_i^{r_i}$ of the integer a (Mod M), it can be reconstructed thus:

$$\text{Let } d_i = M/(p_i^{r_i})$$

$$\text{and } d_i^{-1} = (d_i \bmod p_i^{r_i})^{-1} \bmod p_i^{r_i}$$

$$\text{Then } a = \sum_{i=1}^{s} d_i \left((a_i \cdot d_i^{-1}) \bmod p_i^{r_i}\right) \text{ Mod } M$$

It is possible to optimize such a hybrid scheme for computational complexity, parallelism, economy and simplicity in hardware design[10]. This hybrid scheme will have extensive applications in linear computations, Fourier transforms/convolutions and digital filter design[11].

iii. Example:

Consider the solution of

$$A x = b$$

where A is a non-singular (nxn) matrix with rational entries a_{ij} and b is a (nx1) vector with rational entries b_k. It is easily proved that for Gaussian elimination

$$p^r > 2 \left(\prod_{i=1}^{n} \left(\sum_{j=1}^{n} a_{ij}^2 \right)^{\frac{1}{2}} \prod_{k=1}^{n} |b_k| \right)$$

$$(b_k \neq 0)$$

$$\text{or } r > \frac{1}{\log p} \left(\log 2 \left(\prod_{i=1}^{n} \left(\sum_{j=1}^{n} a_{ij}^2 \right)^{\frac{1}{2}} \prod_{k=1}^{n} |b_k| \right) \right)$$

$$\text{Let } A = \begin{bmatrix} 2 & 2 & -1 \\ -3 & 0 & 2 \\ 4 & -5 & -1 \end{bmatrix}; \quad b = \begin{bmatrix} 5 \\ -5 \\ 0 \end{bmatrix};$$

The choice p = 2, r = 16 would suffice.

The input matrix A, b are converted to H (2, 16) form. Then the augmented matrix (A : b) is reduced to the upper triangular form.

The COMDEN in this case coincides with the determinant of A which equals the product of the pivotal elements used in the elimination.

For convenience, we express the mantissa of H(2, 16) code in hexadecimal form and exponents in decimals. Then,

$$A:b = \begin{bmatrix} .2000,0 & .2000,0 & .ffff,0 & .5000,0 \\ .dfff,0 & .0000,0 & .2000,0 & .bfff,0 \\ .4000,0 & .bfff,0 & .ffff,0 & .0000,0 \end{bmatrix}$$

Elimination:

$$\begin{bmatrix} .1000,0 & .1000,0 & .ffff,-1 & .5000,-1 \\ 0000,0 & .3000,0 & .1000,-1 & .5000,-1 \\ 0000,0 & .7fff,0 & .1000,0 & .6fff,0 \end{bmatrix}$$

$$\begin{bmatrix} .1000,0 & .1000,0 & .ffff,-1 & .5000,-1 \\ 0000,0 & .1000,0 & .baaa,-1 & .7555,-1 \\ 0000,0 & 0000,0 & .5000,-1 & .bfff,-1 \end{bmatrix}$$

$$\begin{bmatrix} .1000,0 & .1000,0 & .ffff,-1 & .5000,-1 \\ 0000,0 & .1000,0 & .baaa,-1 & .7555,-1 \\ 0000,0 & 0000,0 & .1000,0 & .ffff,0 \end{bmatrix}$$

COMDEN = (.2000,0) . (.3000,0) . (.5000,-1) =

(.f000,0)

Conversion to rational form:

$$x = \frac{\text{VALUE (x . COMDEN)}}{\text{VALUE (COMDEN)}} = \frac{1}{15}\begin{bmatrix} 15 \\ 15 \\ -15 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

ACKNOWLEDGEMENT

The author wishes to record his indebtedness to Professor O.J. Fagbemi, Department of Computer Science, University of Lagos, for hospitality during this work.

REFERENCES

1. Knuth, D.E.; The Art of Computer Programming, Vol.2, semi-numerical Algorithms, Reading, Mass., Addison-Wesley, 1971

2. Young, D.M., and Gregory, R.T.; A survey of Numerical Methods, Vol.II, Reading, Mass;, Addison-Wesley, 1973

3. Rao, T.M., Subramanian, K., and Krishnamurthy,E.V., Residue Arithmetic Algorithms for Exact Computation of g-inverses of Matrices, SIAM, J. Numerical Analysis, to appear.

4. Borodin, A., and Munro, I., Computational Complexity of Algebraic and Numerical Problems, Newyork, American Elsevier, 1975.

5. Krishnamurthy, E.V., Mahadeva Rao, T., and Subramanian, K., Proc. Indian. Academy of Sciences, Vol. 81, pp. 58-79, 1975

6. Bachmann, G., Introduction to p-adic Numbers and valuation Theory, Newyork, Academic Press, 1964.

7. Borevich, Z.I., and Shafarevich, I.R., Number Theory, Newyork, Academic Press, 1966.

8. Berezin, I.S., and Zhidkov, N.P., Computational Methods, Volume II, New-york, Pergamon Press, 1965.

9. Faddeev, D.K., and Faddeeva, V.N., Computational Methods of Linear Algebra, London, W.H. Freeman and Company, 1963.

10. Farinmade, J., Parallelism in Matrix Computations, M.Sc. Thesis, Department of Computer Science, University of Lagos, Lagos, 1975.

11. Agarwal, R.C., and Burrus, C.S., Number Theoretic Transforms to Implement Fast Digital Convolution, Proc. IEEE, Vol. 63, pp. 550-560, 1975.