Analysis Of Exact Solution Of Linear Equation Systems Over Rational Numbers By Parallel p-adic Arithmetic

Siddharth Anand
School of Electrical and
Computer Engineering
Oregon State University
Corvallis, Oregon-97330
Email: anand@cs.orst.edu

Shriprakash Sinha
School of Electrical and
Computer Engineering
Oregon State University
Corvallis, Oregon-97330
Email: sinha@cs.orst.edu

***Abstract*— We study and investigate the p-adic arithmetic along with analysis of exact solution of linear equation systems over rational numbers. Initially we study the basic concepts involving the p-adic numbers and why they form a better representation.**

After that we describe a parallel implementation of an algorithm for solving systems of linear equations over the field of rational number based on the Gaussian elimination.The rationals are represented ny truncated p-adic expansion. The approach leads to error free computations directly over the rationals without converting the system to an equivalent one over integers.

The parallelization is based on multiple homomorphic image technique and the result is recovered by parallel version of Chinese Remainder Algorithm.

I. Introduction

Its very illuminating to think about the fact that some at most four hundred years ago, professors at European universities would tell the brilliant students that if they were very diligent, it was not impossible to learn how to do long division. You see, the poor guys had to do it in Roman numerals. Now, here you see in a nutshell what a difference there is in a good and bad notation. (Dijkstra 1977)

We consider that a good scheme for representing numbers, especially for computers, would have the following characteristics. (a) All rational numbers are finitely representable. This requires that the representation be variable-length. (b) The representation is compact. It should require less space, on average, than the fixedlength schemes commonly used in computers. Since the numbers provided by a fixed-length representation are not used equally often, compactness can be achieved by giving frequentlyoccurring numbers short encodings at the expense of longer encodings for less-frequent numbers. (c) The addition, subtraction, and multiplication algorithms are those of the usual integer arithmetic. The division algorithm is as easy as multiplication, and it proceeds in the same direction as the other three algorithms. This property is important for the storage and retrieval of variable-length operands and results.

Although the desired representation is variable-length, there is no implication that operations must be performed serially by digit. Just as data can be retrieved and stored d digits at a time, so the arithmetic unit can be designed to perform operations d digits at a time. By choosing d to be large compared to the average length of operands, we can obtain the speed of a fixed-length design together with the ability to handle operands that are not representable in a fixed length (Wilner 1972).

A. Background

Before we present our proposal, we shall briefly review representations in common use. (For their history, see (Knuth 1969a).) Almost universally, the sequence of digits

$$.....d_i.....d_3d_2d_1d_0$$

is used to represent the nonnegative integer

$$n = \sum_i d_i b^i$$

where b (the base) is an integer greater than one (usually two or ten), and each digit d_i, represents an integer in the range $0 \leq d_i < b$. As a rule, we do not write leading 0s. We must break the rule, however, to represent zero (if we follow it, there is nothing to write).

There is no direct representation of negative integers in common use. Instead we prefix a unary operator to the representation of positive integers. The combination of sign and magnitude is indirect because, to perform arithmetic, we may first have to apply some algebraic transformations. If asked to add two numbers, we first examine the signs to determine whether to use the addition or subtraction algorithm; if we are using the subtraction algorithm, we compare the magnitudes to determine which is to be subtrahend, and which minuend. With a direct representation, if asked to add, we simply add. The radix complement representation is direct in this sense, but it includes only a finite subset of the integers.

Rationals are commonly represented by a pair of integers: a numerator and denominator. In this form, multiplication and division are reasonably easy, but addition and subtraction are relatively hard, and normalization is difficult (Horn 1977). When addition and subtraction are wanted more often than

multiplication and division, a representation that makes the former easier at the expense of the latter would be preferable. For this reason, we usually restrict our numbers to a subset of the rationals known as the fixed-point or floating-point numbers. By inserting a radix point in a sequence of digits (fixed-point), or indicating by means of an exponent where a radix point should be placed (floating-point), we represent those rationals such that, in lowest terms, the denominator divides some power of the base. In this form, addition and subtraction are, after alignment of the radix point, the same as for integers. With a variable-length representation, a major diffculty with the usual division algorithm is that it proceeds from left to right, opposite to the direction of the other three algorithms. To simplify retrieval, processing, and storage, all algorithms should examine their operands and produce their results in the same direction.

The left-to-right division algorithm gives us a way of extending the fixed/floating point representation to include all positive rationals: an infinite but eventually repeating sequence of digits can be finitely denoted by indicating the repeating portion. On paper, the repeating portion is sometimes denoted by overscoring it; for example, $611/495 = 1.2\overline{34}$. A minor annoyance is the fact that representations are not unique; for example, $0.\overline{9} = 1$ and $0.4\overline{9} = .5$. A major annoyance is that further arithmetic is awkward: addition normally begins with the rightmost digit, but a sequence that extends infinitely to the right has no rightmost digit.

The usual representation of nonnegative integers can be extended in various ways. The base may be negative (Songster 1963), or even imaginary (Knuth 1960). In the balanced ternary representation (Avizienis 1971), the base is three, and the digits represent the integers minus-one, zero, and one. Our extension, which we now present, is in quite a different direction.

B. Construction Of Representation

To construct our representation, we shall follow the approach of Hensels p-adic arithmetic (Hensel 1908, 1913). Hensel begins with the usual representation of nonnegative integers, that is, a sequence of digits $.....d_3d_2d_1d_0$. Each digit d_i represents an integer in the range $0 \leq d_i < b$, where the base b is an integer greater than one. He then constructs the representation of other numbers (all rationals, some irrationals and some imaginary numbers (Knuth 1969b)) by means of arithmetic. We shall limit ourselves to rational numbers, and give a finite representation of them that is implementable in computer hardware.

C. Properties Of The Proposed Number System

Excluding the radix point or exponent, the general form of our representation is

$$d_{n+m}d_{n+m-1}.....d_{n+1}d_nd_{n-1}.....d_2d_1d_0$$

The number represented is

$$\sum_{i=0}^{n} d_ib^i - \sum_{i=n+1}^{n+m} d_ib^i/(b^m-1)$$

where b is the base of the representation. To justify this formula, we shall break the digit sequence into two parts: the digits to the left of the quote will be called the negative part, and the digits to its right will be called the positive part. The positive part was our starting point for the construction of the representation.

$$d_n.....d_0 = \sum_{i=0}^{n} d_ib^i$$

The negative part can be found as follows.
$d_{n+m}d_{n+m-1}.....d_{n+1}$'

$$= d_{n+m}d_{n+m-1}.....d_{n+1}0.....0 + d_{n+m}d_{n+m-1}.....d_{n+1}$$

$$= d_{n+m}d_{n+m-1}.....d_{n+1} * b^m + \sum_{n+1}^{n+m} d_ib^{i-n-1}$$

Therefore

$$d_{n+m}d_{n+m-1}.....d_{n+1} = -\sum_{n+1}^{n+m} d_ib^{i-n-1}/(b^m-1)$$

Putting the positive and negative parts together, we find

$$d_{n+m}.....d_{n+1}d_n.....d_0 = d_{n+m}.....d_{n+1} * b^{n+1} + d_n.....d_0$$

and hence we obtain the above formula.

From the formula, we see that sign determination is trivial. If both positive and negative parts are present, we merely compare their leading digits. Assuming the representation to be normalized,$d_{n+m} \neq d_n$.

if $d_{n+m} < d_n$, the number is positive. if $d_{n+m} > d_n$, the number is negative.

If one part is absent, the sign is given by the part that is present. If both parts are absent, the number is zero.

The above paragraphs suggest two comparison algorithms. The first is to subtract the comparands, then determine the sign of the result. The second is to convert the comparands to right-repeating form, then perform the usual digit-by-digit comparison. The first has the advantage that it is a right-to-left algorithm, but the second may have an efficiency advantage.

II. Basics Of p-adic Arithmetic

For any positive integer m, denote $\mathbf{Z_m}$ the ring of integers modulo m and by $|.|$ the canonical ring homomorphism from $\mathbf{Z}$ to $\mathbf{Z_m}$. Let $\mathbf{N}$ be the set of natural numbers.for a given prime p, a rational number α can be represented in a unique way as

$$\alpha = (c/d) * p^e,$$

where c,d and e are integers, c,d and p pairwise relatively prime and d positive.Furthermore α can be uniquely expressed in the following form:

$$\alpha = \sum_{i \geq e} a_i p^i \qquad \text{where } a_i \in \mathbf{Z_p},$$

The infinte sequence ($a_e a_{e+1}....a_{-1}a_0a_{+1}...$) is called the *p-adic representation* of α. We use a *truncated representation*, defined as follows.

Definition1 **(Hensel Codes).** Let p be prime and $r \in \mathbf{N}$. For any rational number $\alpha = (c/d) * p^e$, where *c,d* and p pairwise relatively prime, the *Hensel code* $H_{p,r}(\alpha)$ of length r of is the pair

$$(mant_\alpha, exp_\alpha) = (a_0 a_1 .....a_{r-1}, e),$$

where the r leftmost digits of the p-adic representation of α and e are called the *mantissa* and the *exponent*, respectively.

One easily verifies that we have

$$|c * d^{-1}|_{p^r} = \sum_{=0}^{r-1} a_i \cdot p^i \in \mathbf{Z_{p^r}}.$$

Let $\mathbf{H}_{p,r}$ denote the set of all Hensel codes w.r.t the prime p and the code length r, $\mathbf{H}_{p,r} := \{H_{p,r}(\alpha) \mid \alpha \in \mathbf{Q}\}$. The forward and backwards mapping between $\mathbf{Q}$ and $\mathbf{H}_{p,r}$ are algorithmically computed by the Extended Euclidian Algorithm(EEA), as we state in the following theorems.

Theorem2 **(Forward Mapping).** *Let p be prime and $r \in \mathbf{N}$. Let $\alpha = (c/d) * p^e$ be a rational, such that c,d and p pairwise relatively prime. Then the mantissa $mant_\alpha$ of the code $H_{p,r}(\alpha)$ is computed by EEA applied to p_r and d as*

$$mant_\alpha \equiv c \cdot y \qquad (\text{mod } p^r),$$

where y is the second output of the EEA.

Definition3 **Farey Fraction Set.** let $\mathbf{N}_{(p,r)} = \lfloor \sqrt{(p^r - 1)/2} \rfloor$.The *Farey fraction set* $\mathbf{F}_{p,r}$ of order $N_{(p,r)}$ is the subset of rational number a/b such that:

$$a, b \in \mathbf{N}, 0 \leq a \leq N_{(p,r)}, 0 \leq b \leq N_{(p,r)}.$$

Theorem4 **(Backward Mapping).** *Let p be prime, $r \in \mathbf{N}$ and $c/d \in \mathbf{F}_{p,r}$.If m is a value in the $\mathbf{Z}_{p^r}$ of the Hensel code mantissa related to c/d, then the EEA applied to p^r and m computes a finite sequence pair (x_i, y_i) such that $x_i/y_i = c/d$ for some x.*

Arithmetic operations on Hensel codes are carried out, digit by digit, startting for the leftmost digit, as in the usual base-p arithmetic operations. An addition (or a substraction) can give a result in which some leftmost digits are equal to zero. in this case we that the addition (or substraction) produced a *pseudo-Hensel code*.

Definition5 **(Pseudo-Hensel codes).** A *pseudo-Hensel code* is a code such that $a_0 = ...a_k = 0$, for some k with $0 < k \leq r - 1$.

This loss of significative digits does not permit one to execute division.It can be show that it is possible to overcome this problem by introducing a new approach for both division and for the treatment of pseudo-Hensel codes. These resultsa extend the applicability of p-adic arithmetic to a wide class of computing problems.

In order to reduce the occurences of pseudo Hensel codes by choosing an appropriate base p,we remark that probability of finding a leading zero in a code is $1/(p-1)$. The probability of obtaining a leading zero after addition of two hensel codes given by the probabilty of finding a ($1 \leq a \leq p$) as leading digit of first code and $p - a$ as the leading digit of the second code, that is $1/(p-1)_2$. the same occurs for substraction. from the computational point of view, the best possible choice for p is hence made by taking p to be the greatest prime number less than the maximum integer representable in a memory word. On the other way we want to avoid overflow during computations, hence we will fix the base p on the ground of the word size w of our computer: $p \leq 2^{w/2} + 1$.

The possiblity of performing the exact arithmetic operations on$\mathbf{H}_{p,r}$ is ensured by the following theorem.

Theorem6 *Let p be prime, $r \in \mathbf{N}$, $\alpha_1, \alpha_2 \in \mathbf{Q}$, $\phi \in \{+, -, \cdot, /\}$ an arithmetic operator. If*

$$\alpha_1 \phi \alpha_2 = \alpha_3, \text{with } \alpha_3 \in \mathbf{F}_{p,r}$$

then there exits precisely one $H_{p,r}(\alpha_3)$ such that

$$H_{p,r}(\alpha_3) = H_{p,r}(\alpha_1) \phi' H_{p,r}(\alpha_2)$$

where ϕís the operator in $H_{p,r}$ which corresponds to ϕ in $\mathbf{Q}$

A scheme for a general computation consists of mapping on $H_{p,r}$ the rational input numbers and then performing the computations over the $H_{p,r}$. However by Theorem 4, the inverse mapping can be performed only when the expected result belongs to $F_{p,r}$. This means that we need a bound on the size of the result, in order to make the right choice for p and r.

III. Bounds For The Solutions

Before showing the parallel implementation in the next session, we present the sequential method of Gaussian elimination and an estimate for the size of the result. The problem is stated as follows.

Problem *Given $A \in \mathbf{Q}_{n*n}$ and $b \in \mathbf{Q}_n$, find, if it exits, a vector $x = (x_1, ..., x_n) \in \mathbf{Q}^n$ such that*

Ax = b.

We consider the case $A \in \mathbf{Z}^{n*n}$ and $b \in \mathbf{Z}_n$ first. By Cramer's rule we know that

$$x_i = |A_i|/|\mathbf{A}|,$$

where $|A|$ denotes the determinant of A, and A_i is the matrix obtained from A by substituting the i^{th} column by **b**. Now let a be the maximal entry in a matrix $M \in \mathbf{Z}^{n*n}$, then one easily proves by induction on n that $|M| \leq n! a^n$. From this and $x_i = |A_i|/|\mathbf{A}|$ we obtain that both neumarator and denominator of any x_i are bounded by $n! a^n$, where a is now a maximal entry in A and **b**. from this bound we determine the value of r,

such that the result is in $\mathbf{F}_{p,r}$ for a given prime p. From this definition it suffices that $n!a^n \leq \lfloor\sqrt{(}(p^r-1)/2\rfloor$.

considering the square on both sides of the inequality we obtain $2(n!a^n)^2+1 \leq p^r$. This implies $log_p(2(n!a^n)^2+1) \leq log_p p^r$ or, equivalently,

$$r \geq log_p(2(n!a^n)^2+1).$$

Hadamard's inequality gives another bound for the determinant

$$|A|^2 \leq \prod_{i=1}^{n}(\sum_{j=1}^{n} a_{i,j}^2)_{1/2}.$$

From this bound the following condition is derived

$$p^r \leq \sum_{i=1}^{n}|b_i| \prod_{i=1}^{n}(\sum_{j=1}^{n} a_{i,j}^2)_{1/2}.$$

It should be remarked that both bounds are still quite conservative, since a smaller choice of p and r is often sufficient. in the general case $A \in \mathbf{Q}^{n*n}$ and $\mathbf{b} \in \mathbf{Q}^n$ the bound for the numerator and denominator of the x_i's becomes $n!a^{n(n+1)}$. this follows again from Cramer's rule by considering the equivalent system obtained from A by multiplying each row by the common denominator of all entries in that in that row and of the i^{th} entry in **b**, i.e., multiplying a number of magnitude at most a^{n+1}.

IV. Parallel Implementation

We describe the parallelization on, say, k concurrent processors. We first compute the k prime numbers $p_1, ..., p_k$ at random and the corresponding code length r such that the entries of the solution x are contained in $\mathbf{F}_{g,r}$, where $g = p_1, ..., p_k$.

At this point k parallel tasks are started. Each of them computes the image of the problem w.r.t one prime in p-adic representation of the rational entries, i.e., $H_{p_i,r}(A)$ and $H_{p_i,r}(b)$. By a certain abuse of notation we denote by $H_{p_i,r}(A)$ the matrix $(\widetilde{a}_{i,j})$ with $\widetilde{a}_{i,j} = H_{p_i,r}(a_{i,j})$, and analogously for $H_{p_i,r}(b)$.

Then for each processor a sequential implementation of Gaussian elimination is executed via p-adic arithmetic. The main steps of this algorithm are given below.

Gaussian Elimination for p-adic Computations

Input: n: degree of the linear system; $A = (a_{i,j}) \in \mathbf{Q}^{n*n}$ n-dimensional square matrix; $b = (a_{1,n+1}, ..., a_{n,n+1}) \in \mathbf{Q}^n$:column vector;p : prime base;

Output:$x = (x_1, ..., x_n) \in \mathbf{Q}^n$: solution of $A\mathbf{x} = \mathbf{b}$, if it exits;

Begin

1.1
- Compute the maximum integer number a among the numerators and denominators of the rational entries in A and b.
- Compute the truncation order r, as shown in $r \geq log_p(2(n!a^n)^2+1)$.

1.2 Apply the mapping $H_{p,r}$ to all the entries of A and b.

1.3
- $l := 0$
- for $i := 1$ to n {
  - $l := l+1$
  - for $j := i$ to $n+1$ {
    - Divide $a_{i,j}$ by $a_{i,i}$
  - }
  - for $h := l+1$ to n {
    - Multiply the i^{th} row by $a_{h,l}$
    - Subtract the new i^{th} row from the h^{th} row of the system obtained at the last iteration. This is the new h^{th} row.
  - }
- }

1.4 Recover $H_{p,r}(x_1), ..., H_{p,r}(x_n)$

End

Gaussian Elimination computes solutions $\mathbf{x}^{(i)} \in H_{p_i,r}^n$ for $i = 1, ..., k$. After collecting all of the $\mathbf{x}^{(i)}$ we execute k concurrent calls of CRA. Then the parallel version of CRA is applied on each sequence components $x_j^1, ..., x_j^k$, obtaining the component $x_j \in H_{p_1.....p_k,r}$ of the solution vector **x**.

From the assumptions made on r, the list $\{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(k)}\}$ of results obtained in this way can be mapped back to a vector $\mathbf{x} \in \mathbf{F}_{g,r}^n$ (vector over the Farey fraction set) by the CRA. From Theorem 6 we know that if the solution exits in the $\mathbf{F}_{g,r}$, then it is unique.

After this, the result **x** only need to be converted from the p-adic to the usual representation by the backward mapping, applied in parallel on each component.

A scheme of the parallel computation is given below.

Parallel Computation Scheme

- INPUT — **A,b**
- PRIME COMPUTATION — ↓
- — $p_1, ..., p_k$
- FORWARD MAPPING — ↙ ↓ ↘
- HENSELCODES — $H_{p_1,r}(A), H_{p_1,r}(b)$ .. $H_{p_k,r}(A), H_{p_k,r}(b)$
- — ↓ ↓ ↓
- GAUSS ALGORITHM — ↓ ↓ ↓
- — ↓ ↓ ↓
- SOLUTIONS — $\mathbf{x}^{(1)} = H_{p_1,r}(\mathbf{x})$ .. $\mathbf{x}^{(k)} = H_{p_k,r}(\mathbf{x})$
- SYNCHRONIZATION — ↘ ↓ ↙
- — ↓
- — $\mathbf{x}^1, ..., \mathbf{x}^k$
- — ↘ ↓ ↙
- — ↘ ↓ ↙
- — $\mathbf{x}_1^1, .., \mathbf{x}_1^k$ .. $\mathbf{x}_n^1, .., \mathbf{x}_n^k$
- CRA — ↓ ↓ ↓
- UNIQUE SOLUTION — $\mathbf{x}_1 \in H_{p_1..p_k,r}$ ... $\mathbf{x}_n \in H_{p_1..p_k,r}$
- INVERSE MAPPING — ↘ ↓ ↙
- — ↓
- **OUTPUT** — **x**

V. Conclusion

The above paper was prepared with an aim to throw light on the effectiveness of p-adic numbers and arithmetic along with its application to linear system equations.

A description of the parallel algorithm was given for solving the system of linear equations over rational number. The use of p-adic numbers help in doing error free computations directly over the field of rational numbers, without converting the system to an equivalent problem over integers.

Acknowledgment

We would like to thank Professor Koc for guiding us in the study of p-adic numbers and its application to system of linear equations.

References

[1] C. K. Koc, *A Tutorial on p-adic Arithmetic*, Oregon State University, Corvallis, Oregon, USA, 2002, http://islab.oregonstate.edu/koc/papers/r09padic.pdf

[2] Eric C. R. Hehner and R. Nigel Horspool, *A New Representation of the Rational Numbers for Fast Easy Arithmetic*, SIAM J. Comput., Vol.8, No.2, pages 124-134, 1979, citeseer.nj.nec.com/329968.html

[3] Roberto PIRASTU Carla LIMONGELLI, *Exact Solution of Linear Equation Systems over Rational Numbers by Parallel p-Adic Arithmetic*, No.94-25, Johannes Kepler University, Linz, Austria, 1994, citeseer.nj.nec.com/limongelli94exact.html

[4] G. Kapoulas, *Computable p-adic Numbers*, 1999, citeseer.nj.nec.com/kapoulas99computable.html

[5] Koc Guvenc And, *Exact Solution of Linear Equations on Distributed-Memory Multiprocessors*, citeseer.nj.nec.com/386643.html

[6] Peter Kornerup, *A Systolic, Linear-Array Multiplier for a Class of Right-Shift Algorithms*, IEEE Transactions on Computers, Vol.43, 8, paper 892-898, 1994, citeseer.nj.nec.com/kornerup94systolic.html