p-adic Arithmetic and Parallel Symbolic Computation: An Implementation for Solving Linear Systems

Carla Limongelli, Roberto Pirastu

TECHNICAL REPORT N. RT–INF–1–1995

DIPARTIMENTO DI DISCIPLINE SCIENTIFICHE:
SEZIONE INFORMATICA

CARLA LIMONGELLI [1]
Terza Università degli Studi di Roma
Dipartimento di Discipline Scientifiche: Sezione Informatica
Via Segre, 2 - 00146 Roma - Italy
limongel@inf.uniroma3.it

ROBERTO PIRASTU [2]
Research Institute for Symbolic Computation
J. Kepler University, 4040 Linz, Austria
Roberto.Pirastu@risc.uni-linz.ac.at

[1] Partially supported by Progetto Finalizzato "Sistemi Informatici e Calcolo Parallelo" of CNR under grant n. 93.01625.69.
[2] Supported by the Commission of the European Communities in the framework of the program "Human Capital and Mobility", contract Nr. ERBCHBICT930501

Abstract

In this work we describe the use of truncated p-adic expansion for handling rational numbers by parallel algorithms for symbolic computation. As a case study we propose a parallel implementation for solving linear systems over the rationals.

The parallelization is based on a multiple homomorphic image technique and the result is recovered by a parallel version of the Chinese remainder algorithm. Using a MIMD machine, we compare the proposed implementation with the classical modular arithmetic, showing that truncated p-adic arithmetic is a feasible tool for solving systems of linear equations working directly over rational numbers. A safe algorithm for computing the p-adic division operation is proposed.

The implementation leads to a speedup up to seven by ten processors with respect to the sequential implementation.

KEYWORDS: Parallel algorithms, p-adic arithmetic, scientific computing, computer algebra, shared memory machines.

1 Introduction

In this paper we describe an application of the truncated p-adic representation for rational numbers in parallel symbolic computation.

Although many problems over rational numbers can be reduced to the case of integers, our goal is to show that the p-adic representation is a suitable tool for parallelizing algorithms using homomorphic image techniques working directly over rational numbers. In particular, we compare our implementation with equivalent ones based on modular representation of integers.

p-adic arithmetic has been chosen for two main reasons.

1. p-adic arithmetic representation provides a unified form to treat numbers and functions by means of truncated power series and it constitutes the mathematical background for the definition of basic abstract data structures for a homogeneous computing environment. A unified representation can be obtained when numbers and functions are represented by power series and p-adic analysis offers an appropriate mathematical settlement to handle with power series [12]. In [18] is shown how it is possible to treat numbers by truncated power series, as like as the most general p-adic construction methods in an integrated computing environment.

2. p-adic arithmetic is an exact arithmetic and the algebraic bases on which it is founded overcome the problem of the floating point arithmetic, which is essentially due to a lack of algebraic setting. This last characteristic belongs to modular arithmetic too [10, 8], but the difference is that while modular arithmetic works over the integers, p-adic arithmetic operates on rational numbers. In [15] are shown the advantages due to the possibility of working directly over the rationals. Moreover in [19, 14] is shown that also algebraic numbers are representable in this arithmetic.

Despite to its characteristic of modularity and its powerful algebraic properties (completeness of the p-adic metric space [11]), this arithmetic has not received much attentions because of some computational problems, due to the possible lack of the significant digit of the code.

For the case study we choose the classical linear algebra problem of solving linear systems. For a positive integer n we want to solve a system of n linear equations for the n unknowns $x_1, \ldots, x_n$

$$\left\{\begin{array}{rcl} a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n & = & b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,n}x_n & = & b_2 \\ \ldots & & \ldots \\ a_{n,1}x_1 + a_{n,2}x_2 + \ldots + a_{n,n}x_n & = & b_n \end{array}\right. \tag{1}$$

where $a_{i,j}$ and b_i ($i = 1, \ldots, n$ and $j = 1, \ldots, n$) are rational numbers. We will denote the system (1) by $A\vec{x} = \vec{b}$.

Gaussian elimination is often used in numerical analysis to find approximations to the solutions of such systems. It is well known that for the so-called ill-conditioned systems, small errors in the approximation of the coefficients may lead to large errors in the approximation of the solution. For instance, when a system based on floating point numbers attempts a division of a large dividend by a small divisor, the floating point result could be far from the exact result.

The use of exact arithmetic overcomes this problem. We will apply p-adic arithmetic to perform exact computations and we will compare this approach with the one based on modular arithmetic.

There exist mainly two possibilities to compute exact solutions for (1): either one first transforms the problem to the solution of a system over the integers, or one computes with rational numbers. One can compute a matrix A' and a vector $\vec{b}'$ with integer entries, such that the system $A'\vec{x} = \vec{b}'$ has the same solutions as (1). For instance one can apply Cramer's rule to $A'\vec{x} = \vec{b}'$ and obtain a solution avoiding rational arithmetic. This approach has the disadvantage that the entries in A' are in general considerably larger than the entries in A. On the other hand, working directly with system (1) needs an error-free representation of rational numbers and algorithms for error-free computations with them.

In this work we present a parallel implementation for solving linear systems, based on Gaussian elimination algorithm and the p-adic representation of rational numbers via truncated power series w.r.t. a prime basis p. Our parallelization consists of applying the well known Gaussian elimination method (see for

instance [1]) for several homomorphic images of the problem w.r.t. different prime bases, and recovering the result by the Chinese Remainder Algorithm (CRA). The order of truncation r, as well as the prime bases, is chosen in accordance with an a priori estimation of the magnitude of the solution of the problem. This allows us to do error-free computations directly with rational numbers. For a detailed treatment of p-adic arithmetic in the context of symbolic computation, we refer to [8, 12, 14]. Krishnamurthy in [13] proposes a similar method based on CRA, EEA (Extended Euclidean Algorithm) and HLA (Hensel Lifting Algorithm), for inverting matrices with rational entries. Dixon's approach [7] for solving systems of linear equations has been studied in [27, 26]. In this work we particularly want to stress the usefulness of p-adic representation of rational numbers via Hensel codes, therefore we compare our implementation with an equivalent parallel one which uses modular arithmetic.

In order to show that p-adic arithmetic provides an efficient tool for solving linear systems over the rational numbers, we compared our implementation with one using modular arithmetic and with a sequential implementation in the computer algebra system Maple [3]. Aspects of our parallel implementation are also presented in [17].

The implementation was done in PACLIB, a C-language library for parallel symbolic computation [9], on a Sequent parallel machine with a MIMD architecture.

Next section describes the algebraic bases of the proposed arithmetic. Section 3 shows the algebra of Hensel code set and the treatment of the pseudo-Hensel codes which occurs when a significant digit of the code is missed. The computation of a bound for the solutions is discussed in Section 4. In Section 5 the application of Gaussian elimination algorithm using p-adic arithmetic will be outlined. Section 6 is devoted to the features of the parallel implementation and concluding remarks.

The authors wish to thank the anonymous referees for bringing their attention to some relevant literature and for suggesting a reformulation of Section 5.

2 Basic notions on p-adic arithmetic

A non zero rational number $\alpha = a/b$ can always be uniquely expressed as

$$\alpha = \frac{c}{d} \cdot p^e$$

where e is an integer, p is a fixed prime number, and c, d, and p are pairwise relatively prime integers. This representation is called the *normalized form* of α. Moreover $\hat{\mathbb{Q}}$ will indicate the set of rational numbers c/d such that $GCD(d, p) = 1$. The function

$$\| \cdot \|_p : \mathbb{Q} \to \mathbb{R}$$

from the rational numbers $\mathbb{Q}$ to the real numbers $\mathbb{R}$, defined as

$$\|\alpha\|_p = \begin{cases} p^{-e} & \text{if } \alpha \neq 0 \\ 0 & \text{if } \alpha = 0 \end{cases}$$

is then a norm on $\mathbb{Q}$ (see [11]), called the p-adic norm. On the basis of this p-adic norm, it is possible to define a p-adic metric on $\mathbb{Q}$, such that, given two rational numbers α and β, their distance $d(\alpha, \beta)$ is expressed as:

$$d(\alpha, \beta) = \|\alpha - \beta\|_p.$$

Then $(\mathbb{Q}, d)$ is a metric space. Let $\mathbb{Q}_p$ be the set of equivalence classes of Cauchy sequences in $(\mathbb{Q}, d)$, then the system $(\mathbb{Q}_p, +, \cdot)$ forms a field called the field of p-adic numbers, and $(\mathbb{Q}_p, d)$ is a complete metric space.

The main characteristics of the field of p-adic numbers are:

1. the series

$$\sum_{i=0}^{\infty} p^i$$

converges to 1/(1-p) in $(\mathbb{Q}_p, d)$;

2. every non zero rational number α can be uniquely expressed in the form:

$$\alpha = \sum_{i=e}^{\infty} a_i p^i; \quad a_i \in \mathbb{Z}_p; \quad e \in \mathbb{Z}; \quad \|\alpha\|_p = p^{-e}; \quad a_e \neq 0, \tag{2}$$

where $\mathbb{Z}$ represents the set of integer numbers.

The p-adic representation of a rational number α is then an infinite sequence of digits (the p-adic digits) which are the coefficients of the series given in (2):

$$\alpha = (a_e a_{e-1} \ldots a_{-1} \quad . \quad a_0 a_1 a_2 \ldots).$$

Let us recall that the p-adic expansion of a rational number is periodic. Therefore the p-adic representation can also assume the following form:

$$\alpha = (a_e a_{e-1} \ldots a_{-1} \quad . \quad a_0 \ldots a_{k-m-1}{}' a_{k-m} \ldots a_{k-1} a_k)$$

where the m rightmost digits are the period.

Let us now describe the procedure which computes the p-adic representation of a given rational number α:

p-adic Representation of a Rational Number

Input: p: prime number;
$\alpha \in \mathbb{Q}, \alpha \neq 0$, represented by its normalized form, $\alpha = c/d \cdot p^e$;
Output: the coefficients $a_e, a_{e+1}, a_{e+2}, \ldots$ of the p-adic expansion of α;
Begin

$c_1/d_1 := c/d$;

$i := 0$;

repeat

$a_{e+i} :=\mid c_{i+1}/d_{i+1} \mid_p$;

$c_{i+2}/d_{i+2} := \frac{1}{p}(c_{i+1}/d_{i+1} - a_{e+i})$;

$i := i + 1$;

until the period is detected;

End

Here $\mid c_i/d_i \mid_p = \mid c_i \mid d_i^{-1} \mid_p \mid_p$ is the least non-negative remainder of $c_i/d_i \ mod \ p$.

We note that the hypothesis of primality for p is necessary in order to ensure the existence and the uniqueness of $\mid d_i^{-1} \mid_p$. From now on we will consider p a prime number.

Example 2.1 We compute the p-adic expansion of the rational number $3/4$, with $p = 5$ (in this case $e = 0$):

$$\alpha = \frac{3}{4} \cdot 5^0, \quad \frac{c_1}{d_1} = \frac{3}{4};$$

$$a_0 = \mid \frac{c_1}{d_1} \mid_p = \mid \frac{3}{4} \mid_5 = \mid 3 \cdot \mid 4^{-1} \mid_5 \mid_5 = \mid 12 \mid_5 = 2;$$

$$\frac{c_2}{d_2} = \frac{1}{5}\left(\frac{3}{4} - 2\right) = \frac{1}{5}\left(-\frac{5}{4}\right) = -\frac{1}{4};$$

$$a_1 = \mid \frac{c_2}{d_2} \mid_p = \mid -\frac{1}{4} \mid_5 = \mid \mid -1 \mid_5 \cdot \mid 4^{-1} \mid_5 \mid_5 = 1;$$

$$\frac{c_3}{d_3} = \frac{1}{5}\left(-\frac{1}{4}-1\right) = \frac{1}{5}\left(-\frac{5}{4}\right) = -\frac{1}{4};$$

$$a_2 = |\ \frac{c_3}{d_3}\ |_p = |\ -\frac{1}{4}\ |_5 = 1.$$

In general this process will not terminate, but, since we are assuming that α is a rational number, the p-adic expansion will be periodic and we have to continue the detection of the p-adic coefficients until the period is detected. In this case the p-adic expansion of the number $3/4$ is $.211\ldots = .2\,\overline{1}$. □

Arithmetic operations on p-adic numbers are carried out, digit by digit, starting from the left-most digit a_e, as in usual base p arithmetic operations.

The division operation on p-adic numbers is performed in a different way with respect to usual integer arithmetic. Starting from the left-most digit of both the dividend and the divisor, we obtain the left-most digit of the quotient, and so on, in a way similar to the other three basic p-adic arithmetic operations.

In order to make automatic the p-adic arithmetic computations, the usual and obvious problem is related to the length of p-adic digit sequence. A natural solution is reached by introducing a finite length p-adic arithmetic on the so-called Hensel codes as we will show below.

Definition 1 (Hensel Codes) *Let p be a prime number. Then the* Hensel code *of length r of any number $\alpha = (c/d)\cdot p^e \in \mathbb{Q}$ is the pair*

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

where the left-most r digits and the value e of the related p-adic expansion are called the mantissa and the exponent, respectively. Moreover

$$\sum_{i=0}^{r-1} a_i \cdot p^i \in \mathbb{Z}_{p^r}.$$

□

Let $\mathbb{H}_{p,r}$ indicate the set of Hensel codes with respect to the prime p and the approximation r and let $H_{p,r}(\alpha)$ indicate the Hensel code representation of the rational number $\alpha = (a/b)\cdot p^e$ with respect to the prime p and the approximation r.

The forward mapping is essentially the application of EEA to d and p^r in order to find $|\ d^{-1}\ |_{p^r}$. We solve the Diophantine equation $p^r \cdot x + d \cdot y = 1$, since p^r and d are relatively prime. Then $y = |\ d^{-1}\ |_{p^r}$ because

$$|\ p^r \cdot x + d \cdot y\ |_{p^r} = 1 \pmod{p^r} = |\ d \cdot y\ |_{p^r}\ .$$

Theorem 2.1 (Forward Mapping) *Given a prime p, an integer r and a rational number $\alpha = (c/d)\cdot p^n$, such that $GCD(c,p) = GCD(d,p) = 1$, the mantissa $mant_\alpha$ of the code related to the rational number α, is computed by the Extended Euclidean Algorithm (EEA) applied to p^r and d as:*

$$mant_\alpha \equiv c \cdot y \pmod{p^r}$$

where y is the second output of the EEA **applied to** *d* **and** *p^r.*

Proof. See [23]. □

Let us note that the correspondence between the commutative rings $(\hat{\mathbb{Q}}, +, \cdot)$ and $(\mathbb{H}_{p,r}, +, \cdot)$ is not bijective, since each Hensel code mantissa $.\ a_0 a_1 \cdots a_{r-1}$ $(= \sum_{i=0}^{r-1} a_i \cdot p^i \in \mathbb{Z}_{p^r})$ in $\mathbb{H}_{p,r}$, is the image of an infinite subset of the rational numbers. For this reason we need to define a suitable subset of $\hat{\mathbb{Q}}$, such that the correspondence between this subset and $\mathbb{H}_{p,r}$ is injective.

Definition 2 (Farey Fraction Set) *The Farey fraction set* $\mathbb{F}_{p,r}$ *is the subset of* $\hat{\mathbb{Q}}$ *such that:*

$$a/b \in \hat{\mathbb{Q}} : GCD(a,b) = 1$$

and

$$0 \leq a \leq N, \quad 0 < b \leq N, \quad N = \left\lfloor \sqrt{\frac{p^r - 1}{2}} \right\rfloor,$$

where $\mathbb{N}$ *indicates the set of natural numbers.* □

$\mathbb{F}_{p,r}$ will also be called the Farey fraction set of order N, as $N = N(p,r)$.

Definition 3 *The generalized residue class* $\mathbb{Q}_k^\circ$ *is the subset of* $\hat{\mathbb{Q}}$ *defined as follows:*

$$\mathbb{Q}_k^\circ = \{a/b \in \hat{\mathbb{Q}} \text{ such that } \mid a/b \mid_{p^r} = k\}.$$

□

From the last definition it follows that

$$\hat{\mathbb{Q}} = \bigcup_{k=0}^{p^r-1} \mathbb{Q}_k^\circ.$$

Theorem 2.2 *Let* N *be the largest integer satisfying the inequality*

$$2N^2 + 1 \leq p^r$$

and let $\mathbb{Q}_k^\circ$ *contain the order-*N *Farey fraction* $x = a/b$*. Then* x *is the only order-*N *Farey fraction in* $\mathbb{Q}_k$*.*

Proof. See [8]. □

Also the backward mapping is carried out by EEA. In this case we have to solve the following Diophantine equation: $m \cdot x + p^r \cdot y = 1$ for x and y. This means that

$$\frac{m}{p^r} + \frac{y}{x} = \frac{1}{x \cdot p^r} < \frac{1}{x^2},$$

being by hypothesis $x < p^r$, so that we compute an approximation of $\frac{m}{p^r}$. In the sequence of pairs (x_i, y_i) produced by the EEA the result is then found looking for $y_i \in \mathbb{F}_{p^r}$.

Theorem 2.3 (Backward Mapping) *Given a prime* p*, an integer* r*, a positive integer* $m \leq p^r$ *and a rational number* $c/d \in \mathbb{F}_{p,r} \subset \hat{\mathbb{Q}}$*, let* m *be the value in* $\mathbb{Z}_{p^r}$ *of the Hensel code mantissa related to* c/d*, then the EEA, applied to* p^r *and* m*, computes a finite sequence of pairs* (x_i, y_i) *such that there exists a subscript* j *for which* $x_j/y_j = c/d$*.*

Proof. See [23]. □

From these considerations we can finally state the following theorem.

Theorem 2.4 *Given a prime* p*, an approximation* r*, an arithmetic operator* Φ *in* $\mathbb{Q}$ *and the related arithmetic operator* Φ' *over* $\mathbb{H}_{p,r}$*, then for any* $\alpha_1, \alpha_2 \in \mathbb{Q}$*, if*

$$\alpha_1 \Phi \alpha_2 = \alpha_3, \quad \alpha_3 \in \mathbb{F}_{p,r},$$

then there exists precisely one $\beta \in \mathbb{H}_{p,r}$ *such that*

$$H_{p,r}(\alpha_1) \Phi' H_{p,r}(\alpha_2) = \beta$$

and furthermore $\beta = H_{p,r}(\alpha_3)$*.*

On these bases, every computation over $\mathbb{H}_{p,r}$ gives a code which is exactly the image of the rational number given by the corresponding computation over $\hat{\mathbb{Q}}$.

A general schema of computation may consist in mapping on $\mathbb{H}_{p,r}$ the rational numbers given as input to the computation and then performing the computation over $\mathbb{H}_{p,r}$. However, by THEOREM 2.3, the inverse mapping can be performed only when the expected result belongs to $\mathbb{F}_{p,r}$.

We note that the choice of order of truncation, as well as the choice of the base p, are made in accordance with an a priori estimation of the magnitude of the solution of the problem. In fact we must identify a suitable set of Farey fractions that contains the rational solution; the choices of p and r are a consequence of this identification.

Such an estimate depends in general on the given algorithm/problem one is interested in. The computation of the estimate may turn out to be a difficult problem. Let us mention some examples.

1. *arithmetic over the rationals:* Let us consider the computation of a^b, where $a \in \mathbb{Q}$ and $b \in \mathbb{Z}$. The number of bits which are necessary to represent the rational result is: $b \cdot log_2 a$.

2. *algebra of polynomials:* For example, it is easy to compute in advance the maximum coefficient which can be obtained by a polynomial multiplication. In fact: given the polynomials $\sum_{i=0}^{n} a_i \cdot x^i$ e $\sum_{j=0}^{m} b_j \cdot x^j$, if $a = max\{|\ a_i\ |\}_{1 \le i \le n}$, $b = max\{|\ b_j\ |\}_{1 \le j \le m}$ and $c = max\{a, b\}$, then the greatest coefficient of the polynomial result is smaller than $max\{n, m\} \cdot c^2$.

3. *linear algebra:* For example, it is well known that the determinant $D(A)$ of an n-dimensional square matrix A, is bounded by $n! \cdot a^n$, where $a = max\{|\ a_{i,j}\ |\}$, $1 \le i, j \le n$.

There is also a class of mathematical problems which are particularly well-suited for being solved by p-adic arithmetic: these are problems which are affected either by overflow during the computations or by ill-condition as we will see with the case study that we are going to analize and implement.

In Section 4 we will discuss in more detail a bound for the solutions of linear equation systems over rational numbers.

3 Operations with Hensel codes

In this section we treat Hensel codes arithmetic and we face the problem of pseudo-Hensel codes manipulation, which essentially consists in a loss of significant digits (i.e.: the left-most digits) in an Hensel code. This loss of significant digits does not permit one to execute the division as well as literature has been stated in [8]. In this section we will briefly describe the main arithmetic operations, to show that it is possible to overcome this problem, by presenting a new approach both for division computation and for the treatment of the pseudo-Hensel codes. These results, together with the parallelization of p-adic arithmetic, proposed by the author, extend its use in a wide class of computing problems.

Let us consider now the arithmetic operations in $\mathbb{H}_{p,r}$.

- Addition

Given two Hensel codes

$$H_{p,r}(\alpha) = (mant_\alpha, exp_\alpha) \text{ and } H_{p,r}(\beta) = (mant_\beta, exp_\beta),$$

we first manipulate them to have $exp_\alpha = exp_\beta$, then perform the addition taking into account that all the operations are carried out from left to right.

Example 3.1 We want to compute the following addition:

$$\frac{3}{10} + \frac{1}{2}, \text{ in } \mathbb{Z}_{5^4}$$

by choosing $p = 5$ and $r = 4$, we obtain $\mathbb{F}_{5,4} = 17$. The Hensel codes related to 3/10 and to 1/2 are respectively:

$$H_{4,5}(3/10) = (.4\ 2\ 2\ 2\ ,\ -1) \text{ and } H_{4,5}(1/2) = (.3\ 2\ 2\ 2\ ,\ 0)$$

Since the exponents are different, we must normalize the code which has the greater exponent:

$$(.3\ 2\ 2\ 2\ ,\ 0) \longrightarrow (.0\ 3\ 2\ 2\ ,\ -1)$$

Now we can carry out the addition between the mantissas:

+	.	4	2	2	2	,	−1
=	.	0	3	2	2	,	−1
	.	4	0	0	0	,	−1

The code result (.4 0 0 0, -1) represents the rational number 4/5. □

Addition behaves as described in the following operational table in which $\$\mathbb{H}_{p,r}$ represents the complement of the set $\mathbb{PH}_{p,r}$ with respect to $\mathbb{H}_{p,r}$ (i.e. $\mathbb{H}_{p,r} = \mathbb{PH}_{p,r} \cup \$\mathbb{H}_{p,r}$), being $\mathbb{PH}_{p,r}$ the set of the pseudo-Hensel codes, given by the following definition:

Definition 4 (Pseudo-Hensel codes)
A pseudo-Hensel code ($\mathbb{PH}_{p,r}$) *is a code such that* $a_0 = \cdots = a_k = 0 \neq a_{k+1}$, *with* $0 < k < r$-1. *The* order *of a pseudo-Hensel code is then* k.

+	$\$\mathbb{H}_{p,r}$	$\mathbb{PH}_{p,r}$
$\$\mathbb{H}_{p,r}$	$\mathbb{H}_{p,r}$	$\$\mathbb{H}_{p,r}$
$\mathbb{PH}_{p,r}$	$\$\mathbb{H}_{p,r}$	$\mathbb{PH}_{p,r}$

- Subtraction
In order to perform a subtraction it is sufficient first to compute the complement *mod* p^r of the minuend and then to carry out the addition. If the minuend is a pseudo-Hensel code, then the subtraction can be carried out in the usual way, without using the complement of the minuend (except in the case when the subtrahend is the Hensel code which represents zero).

Example 3.2 We compute the following subtraction:

$$\frac{3}{4} - \frac{3}{2}, \text{ in } \mathbb{Z}_{5^4}$$

By choosing $p = 5$ and $r = 4$, we have $\mathbb{F}_{5,4} = 17$. The Hensel codes related to 3/4 and to 3/2 are respectively:

$$H_{4,5}(3/4) = (.2\ 1\ 1\ 1\ ,\ 0) \text{ and } H_{4,5}(3/2) = (.4\ 2\ 2\ 2\ ,\ 0)$$

In order to carry out the subtraction we must get a p-adic unit from the right digit (instead of from the left digit, as usually happens in subtraction between two integer numbers).

−	.	2	1	1	1	,	0
=	.	4	2	2	2	,	0
	.	3	3	3	3	,	0

the code result (.3 3 3 3, 0) represents the rational number $-3/4$. □

The following table shows the operational behaviour of subtraction.

−	$\$\mathbb{H}_{p,r}$	$\mathbb{PH}_{p,r}$
$\$\mathbb{H}_{p,r}$	$\mathbb{H}_{p,r}$	$\$\mathbb{H}_{p,r}$
$\mathbb{PH}_{p,r}$	$\$\mathbb{H}_{p,r}$	$\mathbb{PH}_{p,r}$

- Multiplication

In order to perform multiplication we must operate by multiplying the respective mantissas of the codes, and then we must add their exponents. Also in this case the code result is truncated to r digits.

Example 3.3 We carry out the following operation:

$$\frac{4}{15} * \frac{5}{2}, \text{ in } \mathbb{Z}_{5^4}$$

By choosing $p = 5$ and $r = 4$, we have $\mathbb{F}_{5,4} = 17$. The Hensel codes related to 4/15 and to 5/2 are respectively:

$$H_{4,5}(4/15) = (.3\ 3\ 1\ 3\ ,\ -1) \text{ and } H_{4,5}(5/2) = (.3\ 2\ 2\ 2\ ,\ 1).$$

$*$	.	3	3	1	3	,	-1
$=$	.	3	2	2	2	,	1
		4	0	0	0		
			1	2	3		
				1	2		
					1		
$=$	.	4	1	3	1	,	0

The code result (.4 1 3 1, 0) represents the rational number 2/3. □

The following operational table shows the behaviour of the multiplication and indicates that the multiplication of the elements of $\mathbb{H}_{p,r}$ is always possible.

$*$	$\$\mathbb{H}_{p,r}$	$\mathbb{P}\mathbb{H}_{p,r}$
$\$\mathbb{H}_{p,r}$	$\$\mathbb{H}_{p,r}$	$\mathbb{P}\mathbb{H}_{p,r}$
$\mathbb{P}\mathbb{H}_{p,r}$	$\mathbb{P}\mathbb{H}_{p,r}$	$\mathbb{P}\mathbb{H}_{p,r}$

- Division

In order to perform division we must operate by dividing the respective mantissas of the codes, and then we must subtract the respective exponents.

Example 3.4 We want to carry out the following division:

$$\frac{3}{4} / \frac{6}{5}, \text{ in } \mathbb{Z}_{5^4}$$

By choosing $p = 5$ and $r = 4$, we have $\mathbb{F}_{5,4} = 17$. The Hensel codes related to 3/4 and to 6/5 are respectively:

$$H_{4,5}(3/4) = (.2\ 1\ 1\ 1\ ,\ 0) \text{ and } H_{4,5}(6/5) = (.1\ 1\ 0\ 0\ ,\ -1).$$

In order to carry out the division we must compute the inverse $mod\ p(= 5)$ of the least significant digit of the divisor and then we must multiply it by the most left hand digit of the dividend (or of the partial dividend). In this way we will obtain all the digits of the quotient. In this case we must compute $|1^{-1}|_5 = 1$.

2	1	1	1	1	0	0	0
3	2	4	4	2	4	1	3
	4	0	0				
	1	0	4				
		1	4				
		4	3				
			3				

The code result (.2 4 1 3, 1) represents the rational number 5/8. □

We must pay particular attention when we operate with elements of $\mathbb{PH}_{p,r}$. In fact if the first digit of the divisor is zero, we cannot compute the modular inverse as stated in the classical algorithm described in [8].
The following operational table indicates when a division results in a pseudo-Hensel code.

/	$\$\mathbb{H}_{p,r}$	$\mathbb{PH}_{p,r}$
$\$\mathbb{H}_{p,r}$	$\$\mathbb{H}_{p,r}$	$\mathbb{PH}_{p,r}$
$\mathbb{PH}_{p,r}$	$\mathbb{PH}_{p,r}$	$\mathbb{PH}_{p,r}$

Looking at the examples presented above, we note that an addition or subtraction, may give a result in which some left-most digits are equal to zero. In this case we will say that the addition (or subtraction) has generated a pseudo-Hensel code (see DEFINITION 4).

We define an algorithm which allows us to overcome the problem of pseudo-Hensel code manipulation and also to decrease the frequency with which they occur during computations, in particular at the end of the computation, when the rational result must be detected. In the following we see when a pseudo-Hensel code can occur.

Example 3.5 We want to compute the following addition:

$$\frac{13}{15} + \frac{13}{10}, \quad \text{in} \quad \mathbb{Z}_{5^4}$$

By choosing: $p = 5$, $r = 4$, we obtain $\mathbb{F}_{5,4} = 17$.

The Hensel codes related to 13/15 and to 13/10 are respectively:

$$H_{4,5}(13/15) = (.1\ 4\ 1\ 3\ ,\ -1) \text{ and } H_{4,5}(13/10) = (.4\ 3\ 2\ 2\ ,\ -1)$$

The addition follows:

+	.	1	4	1	3	,	-1
=	.	4	3	2	2	,	-1
	.	0	3	4	0	,	-1

In this case one significant digit has been lost. Now, if we apply backward mapping to detect the rational result, we have an error, because the Farey fraction set is decreased: $\mathbb{F}_{5,3} = 7$. In fact the rational result of this computation is 13/10, which does not belong to $\mathbb{F}_{5,3}$. □

Nevertheless we can carry on with the computation, because the code approximation has not been decreased, but if we want to compute a division in which the dividend belongs to $\mathbb{H}_{p,r}$ and the divisor is a pseudo-Hensel code of order k (with $k < r$), as we have previously observed, we can appropriately manipulate these codes, in order to apply the division algorithm. In the following we will show two algorithms proposed in literature for the manipulation of the pseudo-Hensel codes. The first one was proposed by Dittemberger in 1987 (see [6]) and can be summarized by the following steps:

ALGORITHM DIT87

Input: $q_1 \in \mathbb{PH}_{p,r}$:
a pseudo-Hensel code of order k $(1 \leq k \leq r-1)$;
Output: $\overline{q}_1 \in \$\mathbb{H}_{p,r-k}$;

1. eliminate the k left-most zeros of the mantissa of q_1;
2. increase the exponent of q_1 by k. □

We note that if we want to carry on with the computations, all the codes involved in the computation must be modified, that is their approximation must be decreased by a factor of k. In this way we will eliminate the pseudo-Hensel code, but the approximation of all the codes will be decreased.

The algorithm proposed in [4] tries to rebuild the approximation lost by the pseudo-Hensel code occurrence:

ALGORITHM CM87

Input: $a, b \in \$\mathbb{H}_{p,r}$; $q_1 \in \mathbb{P}\mathbb{H}_{p,r}$, pseudo-Hensel code of order k $(1 \leq k \leq r-1)$, such that $q_1 = a + b$;
Output: $\overline{q}_1 \in \$\mathbb{H}_{p,r}$;

1. apply the inverse mapping to a and to b, by obtaining the rational numbers related to the codes which has produced the pseudo-Hensel code;
2. apply the direct mapping to these rational numbers, with an approximation equal to $r + k$;
3. carry out the addition: a pseudo-Hensel code of order k is obtained in $\mathbb{P}\mathbb{H}_{p,r+k}$;
4. eliminate the first k zeros and update the exponent; now the code will belong to $\$\mathbb{H}_{p,r}$ □

The latter algorithm presents some inconveniences. While in the first algorithm the approximation of all the codes involved in the computation decreases and could also become zero (in this latter case it is necessary to begin the computation again by doubling at least the code approximations), in the latter algorithm the computation with the code stops in order to obtain the rational numbers which have generated the pseudo code. But, in this case, we do not know anything about the order of this intermediate result, hence we can not be certain that backward mapping will give the exact result. Furthermore this last algorithm is time consuming.

Moreover let us note that the recovery of the code approximation is necessary only when a division operation with a pseudo code as divisor occurs.

Hence the new proposal consists in proceeding without manipulating the pseudo-Hensel codes, but to manipulate the division algorithm, only when a pseudo-Hensel code occurs as a divisor. In this case the following algorithm is proposed:

ALGORITHM LIM92

Input: p, r;
$q_1 \in \$\mathbb{H}_{p,r} = (a_0\ a_1\ \ldots a_{r-1},\ 0)$: dividend;
$q_2 \in \mathbb{P}\mathbb{H}_{p,r} = (0\ \ldots 0\ b_k \ldots b_{r-1},\ 0)$: divisor (pseudo-Hensel code of order k);
Output: $q_3 = q_1/q_2 \in \mathbb{P}\mathbb{H}_{p,r} = (0\ \ldots 0\ c_0 \ldots c_{r-1-k},\ -2k)$;

1. represent the dividend in $\mathbb{H}_{p,r-k}$: $(a_0\ a_1\ \ldots a_{r-k-1},\ 0)$;
2. represent the divisor in $\mathbb{H}_{p,r-k}$: $(b_k\ \ldots b_{r-1},\ k)$;
3. carry out the division [8] in $\mathbb{H}_{p,r-k}$: $(a_0\ a_1\ \ldots a_{r-k-1},\ 0)/(b_k\ \ldots b_{r-1},\ k)$;
4. represent the code result $(c_0 \ldots c_{r-1-k},\ -2k)$ in $\mathbb{P}\mathbb{H}_{p,r}$: $(0\ \ldots 0\ c_0 \ldots c_{r-1-k},\ -2k)$. The result will be a pseudo-Hensel code of order k. □

Example 3.6 We want to carry out the following computation:

$$\frac{1}{4}/(\frac{1}{2} + \frac{1}{3}) + \frac{1}{25}$$

We choose, as usual, $p = 5$ and $r = 4$.

$$H_{5,4}(1/4) = (.4\ 3\ 3\ 3,\ 0);\ H_{5,4}(1/2) = (.3\ 2\ 2\ 2,\ 0);$$

$$H_{5,4}(1/3) = (.2\ 3\ 1\ 3,\ 0);\ H_{5,4}(1/25) = (.1\ 0\ 0\ 0,\ -2);$$

$$\frac{1}{2}+\frac{1}{3} = (.3\ 2\ 2\ 2,\ 0)+(.2\ 3\ 1\ 3,\ 0) = (.0\ 1\ 4\ 0,\ 0)$$

$$\frac{1}{4}/(\frac{1}{2}+\frac{1}{3}) = (.4\ 3\ 3\ 3,\ 0)/(.0\ 1\ 4\ 0,\ 0)$$

By applying the above described algorithm we obtain:

Input: $q_1 = (.4\ 3\ 3\ 3,\ 0)$;
$q_2 = (.0\ 1\ 4\ 0,\ 0)$; (pseudo code of order 1);
Output: $q_3 = (.4\ 3\ 3\ 3,\ 0)/(.0\ 1\ 4\ 0,\ 0) = (.0\ 4\ 2\ 2,\ -2)$;

- we represent the dividend in $\mathbb{H}_{5,3}$, by truncating the right-most digit of the mantissa:

$$(.4\ 3\ 3\ 3,\ 0) \longrightarrow (.4\ 3\ 3,\ 0)$$

- we represent the divisor in $\mathbb{H}_{5,3}$, by truncating the left-most digit of the mantissa. In this case we must update the exponent because the position of the code digits is changed with respect to the power of p:

$$(.0\ 1\ 4\ 0,\ 0) \longrightarrow (.1\ 4\ 0,\ 1)$$

- now we carry out the division [8] in $\mathbb{H}_{5,3}$:

$$(.4\ 3\ 3,\ 0)/(.1\ 4\ 0,\ 1) = (.4\ 2\ 2,\ -1)$$

- we represent the code result in $\mathbb{PH}_{5,4}$:

$$(.4\ 2\ 2,\ -1) \longrightarrow (.0\ 4\ 2\ 2,\ -2)$$

Furthermore $H_{5,4}(1/25) = (.1\ 0\ 0\ 0,\ -2)$; hence:

$$(.0\ 4\ 2\ 2,\ -2)+(.1\ 0\ 0\ 0,\ -2) = (.1\ 4\ 2\ 2,\ -2)$$

We note that $H_{5,4}(17/50) = (.1\ 4\ 2\ 2,\ -2)$. □

Computing in this way we can avoid the loss of significant digits (on the contrary, in the algorithm DIT87 the approximation order of all the codes involved in the computation decreases) and allows the manipulation of the pseudo-Hensel codes in the same way as the Hensel codes. We do not need to recover the rational numbers related to the Hensel codes which have generated the pseudo code (as like as the algorithm CM87 does). Moreover, the approximation lost can be recovered during the rest of the computation, because, with our algorithm, the code length is not decreased.

The occurrences of the pseudo-Hensel codes can be sensibly reduced, by taking an appropriate base p and, especially, by parallelizing the arithmetic.

In order to reduce the occurrences of pseudo-Hensel codes choosing an appropriate base p, we can note that the probability of finding a leading zero in a code is equal to $1/(p-1)$. The probability of obtaining a leading zero after an addition between two Hensel code is given by the probability of finding $p-k$ (with $1 \leq k \leq p$) as leading digit of the first code and k as the leading digit of the second code, that is $1/(p-1)^2$. The same occurs for subtraction. From a 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.

The parallelization of p-adic arithmetic is not only suitable to reduce the occurrences of the pseudo-Hensel codes, but it speeds up rational number arithmetic especially when big number computations occur. The idea of the parallel p-adic approach [15] is to represent each rational number by a Hensel code for several values of p, each representing one homomorphic image. The computations are then performed independently in each image. The unique result has to be constructed out of the results in the images in a recovery step by using the Chinese Remainder Algorithm. Finally a backward mapping has to be performed that retrieves the rational number from the Hensel code.

We saw in Section 2. that an *a priori* estimation of the result is necessary in order to fix the base p and the approximation r. In the next section we are going to analyze the apriori estimation of the result related to the solution of a linear siystem over the rationals.

4 Bounds for the Solutions

As we saw in the previous sections, the computation of a suitable bound for the size of the solutions is a fundamental step of any p-adic algorithm. In our case we consider systems of linear equations over rational numbers. This means that, for a given matrix $A \in \mathbb{Q}^{n\times n}$ and $\vec{b} \in \mathbb{Q}^n$, we need an integer m such that, if a solution vector $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{Q}^n$ of $A\vec{x} = \vec{b}$, exists, then the denominator $\mathbf{den}_i$ and the numerator $\mathbf{num}_i$ of each entry x_i is bounded by m, viz.

$$|\mathbf{den}_i| \leq m, \quad |\mathbf{num}_i| \leq m \tag{3}$$

For the case $A \in \mathbb{Z}^{n\times n}$ and $\vec{b} \in \mathbb{Z}^n$ such a bound m can be easily computed, for instance, by Cramer's rule. We have in fact

$$x_i = \frac{|A_i|}{|A|} \tag{4}$$

where $|A|$ denotes the determinant of A, and A_i is the matrix obtained from A by substituting the ith column by $\vec{b}$. Now let a be a maximal entry in a matrix $M \in \mathbb{Z}^{n\times n}$, then by induction on n $|M| \leq n!a^n$. From this and (4) we obtain that both numerator and denominator of any x_i are bounded by $m := n!a^n$, where a is now a maximal entry in A and $\vec{b}$. From this bound we determine a value for r, such that the result is in $\mathbb{F}_{p,r}$ for a given prime p. From the definition it suffices that

$$n!a^n \leq \left\lfloor \sqrt{\frac{p^r - 1}{2}} \right\rfloor \tag{5}$$

Considering the square of 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,

$$r \geq \log_p(2(n!a^n)^2 + 1) \tag{6}$$

Hadamard's inequality (see for instance [22] or [21]) gives another bound for the determinant

$$|A|^2 \leq \prod_{i=1}^{n} \left(\sum_{j=1}^{n} a_{i,j}^2 \right)^{1/2} \tag{7}$$

From this bound the following condition is derived in [8]

$$p^r \geq \sum_{i=1}^{n} |b_i| \prod_{i=1}^{n} \left(\sum_{j=1}^{n} a_{i,j}^2 \right) \tag{8}$$

In practice both bounds are still conservative, since a smaller choice of p and r is often sufficient. In the general case $A \in \mathbb{Q}^{n\times n}$ and $\vec{b} \in \mathbb{Q}^n$ 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 row and of the ith entry in $\vec{b}$, i.e., multiplying by a number of magnitude at most a^{n+1}.

5 The Parallel Algorithm

The parallelization is based on the concurrent application of Gauss' algorithm on several homomorphic p-adic images of the problem.

The multiple homomorphic images technique [12, 20] presents the following characteristics:

1. the image domains are simpler than the original domain so that the image problem can be solved more efficiently;

2. the forward mapping preserves the operations in every image domain as stated in Theorem 2.4;

3. the transformation leads to several independent homomorphic image problems each of which can be solved exactly, independently and in parallel as Figure 1 shows.

4. the correctness of the recovery step is assured by the Chinese Remainder Algorithm that has been parallelized in [15, 16] on the basis of the following theorem taken from [12] (the computation of each summand in (9) is done in parallel).

Theorem 5.1 (Chinese Remainder Theorem) *Let $p_1, \ldots, p_k$ be k relatively prime integers > 1. Then for any $s_1, \ldots, s_k (s_i < m_i)$ there is a unique integer s satisfying*

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

and $s_i \equiv s \mod p_i$; the integer s can be computed using

$$s = \sum_{i=1}^{k} (\frac{M}{p_i}) s_i T_i \mod M, \tag{9}$$

where T_i is the solution of

$$(\frac{M}{p_i}) T_i \equiv 1 \mod p_i.$$

Many computer algebra algorithms use the modular approach. There the input is mapped into a homomorphic image, the computation is done in this image and the CRA is performed in one iteration of a big sequential loop. This step is repeated until enough image results have been computed to reconstruct the result in the original domain[3].

Since in the sequential approach the CRA is interleaved with the remaining algorithm it is not possible to use a parallelized CRA for computing all necessary chinese remainderings in one step.

In the following we describe the parallelizzation algorithm on, say, k concurrent processors, like in Fig. 1. Let us note that we have arbitrarily many processors available that will be automatically mapped on the real processors.

We first compute k prime numbers $p_1, \ldots, p_k$ at random and the corresponding code length r according to the computed bound, such that the entries of the solution $\vec{x}$ are expected to be in $\mathbb{F}_{g,r}$, where g is the smallest prime number such that $g \geq p_1 \cdots p_k$ (Step **1.1**, Fig. 2).

At this point k parallel tasks are started. Each of them computes the image of the problem with respect to one prime in p-adic representation of the rational entries, i.e., $H_{p_i,r}(A)$ and $H_{p_i,r}(\vec{b})$ (Step **1.2**, Fig. 2). By a certain abuse of notation we denote by $H_{p_i,r}(A)$ the matrix $(\tilde{a}_{i,j})$ with $\tilde{a}_{i,j} = H_{p_i,r}(a_{i,j})$, and analogously for $H_{p_i,r}(\vec{b})$.

Then for each processor a sequential implementation of Gaussian elimination is executed via p-adic arithmetic (Step **1.3**, Fig. 2).

Note that an homomorphic image of the problem may not allow a solution, since, for instance, the determinant of the matrix might be zero modulo the particular prime. In this case the program detects that a prime cannot be used and computes, at random, another prime. Then it applies the same algorithm on the new homomorphic image. Although such a situation implies a considerably longer execution time for the Gauss' algorithm, it turned out that the case did not arise often during our tests.

Gaussian elimination computes solutions $\vec{x}^{(i)} \in \mathbb{H}^n_{p_i,r}$ for $i = 1, \ldots, k$. After collecting all of the $\vec{x}^{(i)}$ we execute k concurrent calls of CRA that have to be applied to codes with null exponent. In order to do it, we must multiply and shift the codes (Step **1.4**, Fig. 2). We apply to each sequence of components $x_j^{(1)}, \ldots, x_j^{(k)}$, obtaining the component $x_j \in \mathbb{H}_{g,r}$ of the solution vector $\vec{x}$ (Step **1.5**).

[3] One example for such an algorithm is the computation of the gcd over polynomials with the IPGCDC algorithm in SACLIB (see [2]).

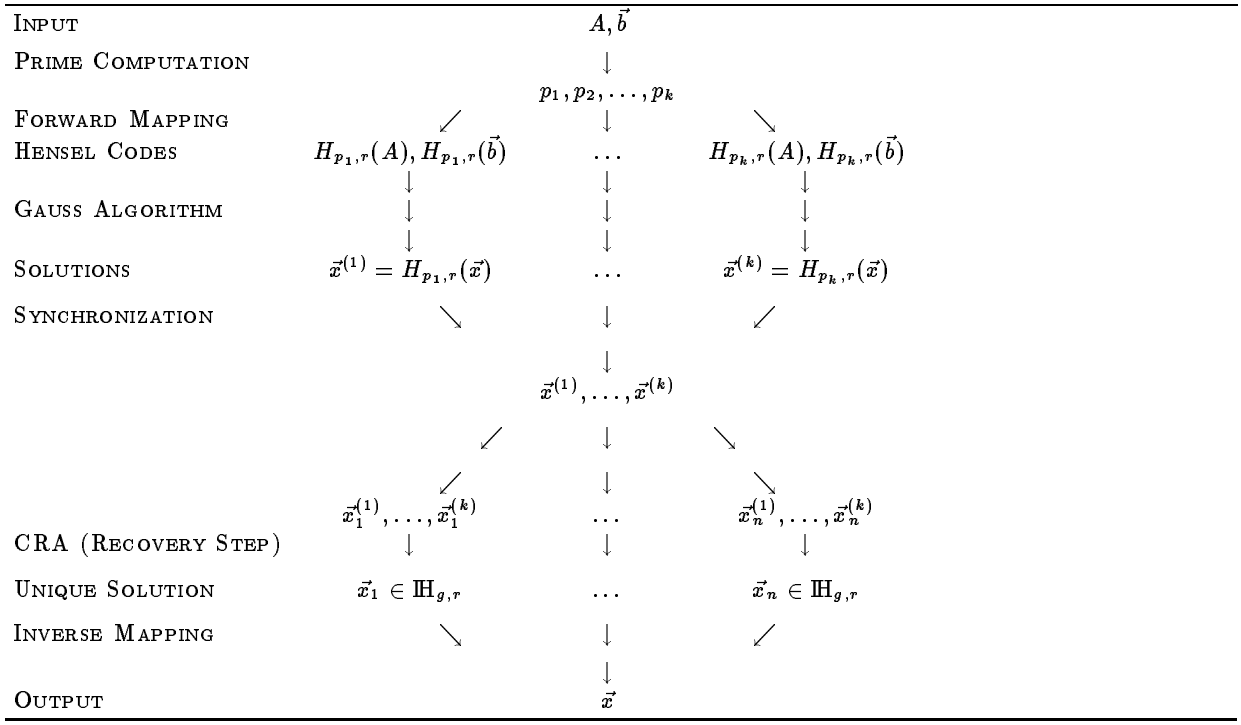

Figure 1: Parallel Computation Scheme

From the assumptions made on the bound and on r, the list $\{\vec{x}^{(1)}, \ldots, \vec{x}^{(k)}\}$ of results obtained in this way can be mapped back to a vector over the Farey fraction set $\vec{x} \in \mathbb{F}_{g,r}^n$ by the EEA. From Theorem 2.4 we know that if the solution exists in $\mathbb{F}_{g,r}$, then it is unique (Step **1.6**).

After this, the result $\vec{x}$ only needs to be converted from the p-adic to the usual representation by the backward mapping, applied in parallel on each component.

In the case of dense matrices with large dimension and size with respect to the number of processors, also standard parallelization techniques for dense linear systems could be applied.

6 Implementation and Experimental Results

As a parallel environment for our implementation we used PACLIB (see [9]), a system developed at RISC-Linz for parallel computer algebra. PACLIB is based on the SACLIB library (see [5]), which provides several computer algebra algorithms written in C. On the other hand, several other symbolic computation systems provide a parallel implementation of a linear system solver. For instance the parallel computer algebra system ||MAPLE|| (see [25]), a parallel version of Maple developed at RISC-Linz, provides a solver based on the Gauss-Jordan algorithm that can also handle symbolic entries [24], and in particular rational numbers.

We performed several tests of our implementation on randomly generated linear systems on a Sequent Symmetry machine with 20 processors, a MIMD computer with shared memory.

The parallel implementation is compared with the corresponding sequential implementation, where we apply sequentially the same mapping onto $\mathbb{H}_{p_i,r}$ for the same primes p_i as in the equivalent parallel execution.

In Table 1 the execution times of both the sequential and the parallel implementation are reported in milliseconds. The input size is the maximum bit length of the numbers. If the entries are rational numbers the numerator and the denominator have a bit length bounded by this input size. Parallelizing the algorithm over 10 processors we achieve a speedup up to 7.5, with respect to the sequential algorithm.

We compared our implementation with an efficient modular parallel implementation for systems over integers which makes use of a mixed method (Gauss and Kramer), implemented in PACLIB. We consider two cases:

Input: n: degree of the linear system;
$A = (a_{i,j}) \in \mathbb{Q}^{n \times n}$: n-dimensional square matrix;
$\vec{b} = (a_{1,n+1}, \ldots, a_{n,n+1}) \in \mathbb{Q}^n$: column vector;
$p_1, \ldots, p_k$, k prime numbers;

Output: $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{Q}^n$: solution of $A\vec{x} = \vec{b}$, if it exists;

Begin

1.1. (Prime Computation)
Compute the maximum integer number a among the numerators and denominators of the rational entries in A and $\vec{b}$; compute the truncation order r, as shown in (6); compute the number k of necessary processors; compute g;

start k parallel tasks

1.2. (Forward Mapping)
In each task apply the parallel mapping $H_{p_i,r}$ to all the entries of A and $\vec{b}$ in order to obtain the Hensel Codes;

1.3. (Gauss' Algorithm)
Compute Gauss' algorithm in each domain $\mathbb{H}_{p_i,r}$ in parallel in order to obtain the k vectors solutions: $\vec{x}^{(1)} = H_{p_1,r}(\vec{x}), \ldots, \vec{x}^{(k)} = H_{p_k,r}(\vec{x})$

end k parallel tasks

1.4. (Synchronization)
For each of the k domains, the exponents of the related codes must become zero (in order to apply CRA);

start n parallel tasks

1.5. (CRA)
Apply parallel CRA to each k-tuple $\vec{x}_i^{(1)}, \ldots, \vec{x}_i^{(k)}$

start k parallel tasks
(Unique Solution $\vec{x}_i$ for CRA)
end k parallel tasks

and find n solutions $\vec{x}_1 \in \mathbb{H}_{p_1 \cdots p_k,r}, \ldots, \vec{x}_n \in \mathbb{H}_{p_1 \cdots p_k,r}$;

1.6. (Inverse Mapping)
Apply the backward mapping to each of the solutions, obtaining $\vec{x} = (x_1, \ldots, x_n)$;

end n parallel tasks

End

Figure 2: Parallel Algorithm

Table 1: Comparison of sequential and parallel algorithm

Dimension	Input Size	Sequential	Parallel	Speedup
10	10	3064	692	4.4
15	10	9388	1792	5.2
20	10	27707	4421	6.2
20	20	57014	7563	7.5
20	30	69481	11196	6.2
25	10	61528	8751	7.0
25	20	108695	15813	6.8
30	10	119890	16893	7.0

Table 2: Comparison of modular, p-adic, and rational p-adic when length=10.

Dimension	Modular	p-adic	Rational
5	321	218	278
10	692	619	718
15	1422	1719	1315
20	3210	3315	2995
25	5972	5756	5579
30	14309	12299	11107

Table 3: Comparison of modular, p-adic, and rational p-adic when dim. =20.

Input length	Modular	p-adic	Rational
5	1908	2024	1722
10	3007	2966	2903
15	3519	3957	3416
20	5760	5441	4421
25	7502	8785	7067
30	9845	11151	9037
35	11222	11734	10023
40	22406	21372	11023

1. The input data are integers;
2. The input data are rationals. We present the case where only the vector $\vec{b}$ has rational entries.

In the first case, p-adic arithmetic is essentially reduced to modular arithmetic and the backward mapping becomes almost trivial, so that no improvement is achieved. Since the denominator of any component x_i of the solution $\vec{x}$ divides $|A|$. So $H_{g,r}(x_i) \cdot H_{g,r}(|A|)$ is an integer and no backward mapping is needed, since we have

$$x_i = \frac{/H_{g,r}(x_i)H_{g,r}(|A|)/_g}{|A|} \ , \tag{10}$$

where $/\cdot/_g$ is the *signed* modulo mapping to $\{-\frac{g-1}{2}, \ldots, \frac{g-1}{2}\}$. Remark that the determinant $|A|$ is computed as a direct by-product of Gaussian elimination.

In the second, more interesting case, in order to do a fair comparison with the modular algorithm, which accepts only integers entries, we have to study the size of equivalent inputs for both algorithms. Let $A \in \mathbb{Q}^{n \times n}$ be again the matrix describing a system over the rational numbers and let s be the maximal size among all denominators of the entries in A and $\vec{b}$. We must transform $A\vec{x} = \vec{b}$ to an equivalent system $A'\vec{x} = \vec{b'}$ with integer entries. This means to multiply each row/equation by an appropriate integer. It is easy to see that the smallest integer m_j such that $m_j\vec{a}_j, m_j b_j$ are all integers, is equal to the l.c.m. over all denominators arising in the jth row $\vec{a}_j$ (resp. b_j) of A (resp. $\vec{b}$). In the worst case, m_j will be the product of all denominators (i.e., $m \leq (n+1)s$). In the comparison, one must take into account this fact, although the average size of m will be usually smaller than this.

Here we are considering rational entries only in $\vec{b}$, so if s is the size of the entries for the p-adic algorithm, the entries of the modular algorithm will be of size $2s$.

In Table 2 we show the behaviour of the algorithms for fixed input length. For the considerations stated above a length of 10 for integer entries means a length of 5 for the rational case.
In Table 3 the timings of some executions are shown, where the dimension of the system is 20.

We also compare our sequential algorithm with the implementation available in MapleV (see [3]). MapleV implements a fraction free Gaussian elimination, so the equations are converted to have integer coefficients

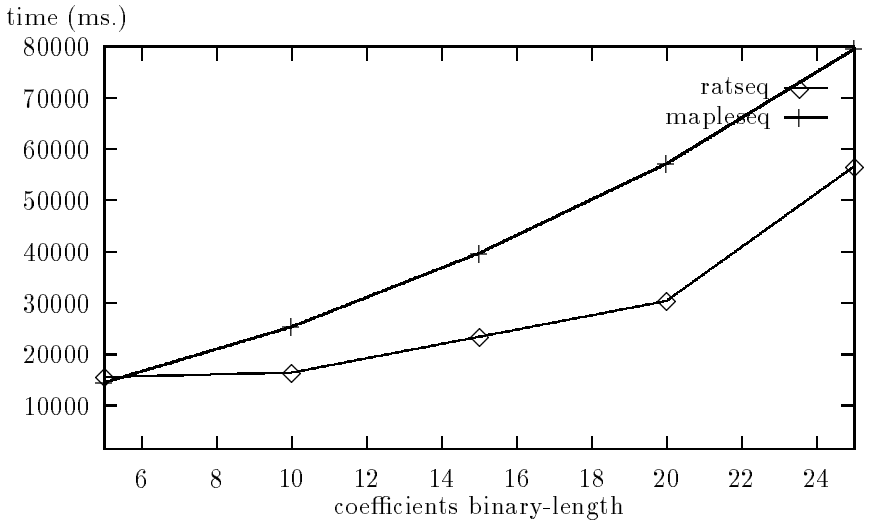

Figure 3: Comparison of MapleV and the sequential p-adic algorithm with dim. =20.

and after each elimination step, the greatest common divisor of the coefficients is divided out to minimize growth. The timings in Fig. 3 show the behaviour of the algorithms.

7 Conclusions

In this article we presented a case study for the suitability of truncated p-adic arithmetic in parallel symbolic computation algorithms over rational numbers.

We presented a parallel implementation for solving systems of linear equations over the rationals. For the computations, the rational entries are represented by Hensel codes, i.e., by a truncated p-adic representation. This approach permits us to do error-free computations directly over the field of rational numbers, without converting the system to an equivalent problem over the integers. The parallelization is done by applying the sequential Gaussian elimination to different p-adic images of the problem with respect to several prime bases, using the p-adic arithmetic described in [14]. The result is recovered by a parallel Chinese remainder algorithm. Using 10 processors the parallel implementation achieves a speedup up to 7.5 with respect to the sequential one.

We compared our implementation with a parallel modular approach over the integers and with the built-in sequential implementation in MapleV. As expected, the p-adic representation is at least as efficient as the modular one for the case of integer coefficients and more efficient for the case of rational coefficients.

These experimental data confirm the expected behaviour of linear algebra algorithms implemented via p-adic arithmetic as regards the heavy computational complexity of CRA. In [15] it is shown that for problems with many large input data the asymptotic running time of the p-adic algorithm is never dominated by the cost of the recovering step.

This case study strongly encourages us to do some further work in order to show the suitability of p-adic arithmetic for the parallel solution of other problems over rational numbers. In particular, a more detailed comparison with the p-adic techniques described in [26] has to be done.

References

[1] Aho, A. V., Hopcroft, J. and Ullman, J. D.: *The Design and Analysis of Computer Algorithms*. Addison

Wesley Publishing Company, 1975.

[2] Buchberger, B., Collins, G. E. et al.: A SACLIB primer. Technical report, RISC-Linz, Johannes Kepler University, Austria, 1992.

[3] Char, B. W., et al.: A tutorial introduction to MAPLE. *Journal of Symbolic Computation*, Vol.2, n.2, 1986.

[4] Colagrossi, A. and Miola, A: A normalization algorithm for truncated p-adic arithmetic. In *Proc. IEEE 8-th Symposium on Computation Arithmetic*, Como, May 1987.

[5] Collins, G. E. et al. A SACLIB 1.1 user's guide. Technical report, RISC-Linz, Johannes Kepler University, Austria, 1993. RISC-Linz Report Series n. 93-19.

[6] Dittenberger, K.: An efficient method for exact numerical computation. Diploma Thesis, Johannes Kepler University, RISC-Linz Institute Austria, 1987.

[7] Dixon, J. D.: Exact solution of linear equations using p-adic expansions. *Numer. Math.*, 40:137–141, 1982.

[8] Gregory, R. T. and Krishnamurthy, E. V. : *Methods and Applications of Error-Free Computation.* Springer Verlag, 1984.

[9] Hong, H. et al.: A PACLIB user manual. Technical report, RISC-Linz, Johannes Kepler University, Austria, 1992. RISC-Linz Report Series n. 92-32.

[10] Knuth, D.: *The Art of Computer Programming*, volume 2. Addison Wesley Publishing Company, 1981.

[11] Koblitz, N.: *p-adic Numbers, p-adic Analysis and Zeta Functions.* Springer Verlag, 1977.

[12] Krishnamurthy, E. V.: *Error-Free Polynomial Matrix Computations.* Springer Verlag, 1985.

[13] Krishnamurthy, E. V.: Algebraic transformation approach for parallelism. In L. Kronsjo, editor, *Advances in Parallel Algorithms*, pp. 151–178. Blackwell, 1993.

[14] Limongelli, C.: *The Integration of Symbolic and Numeric Computation by p-adic Construction Methods.* PhD thesis, University of Rome "La Sapienza" Italy, 1993. Technical Report, Collection of theses, V-93-4.

[15] Limongelli, C.: On an efficient algorithm for big rational number computations by parallel p-adics. *Journal of Symbolic Computation*, Vol.15, n.2, 1993.

[16] Limongelli, C. and Loidl, H. W.: Rational number arithmetic by parallel p-adic algorithms. In Springer Verlag, editor, *Proc. of Second International Conference of the Austrian Center for Parallel Computation (ACPC)*, *LNCS*, Vol. 734, Springer Verlag, 1993.

[17] Limongelli, C. and Pirastu, R.: Exact solution of linear equation systems over rational number by parallel p-adic arithmetic. In B. Buchberger and J. Volkert, editors, *Parallel Processing: CONPAR 94 - VAPP VI*, *LNCS*, Vol. 854, Springer Verlag, 1994.

[18] Limongelli, C. and Temperini, M.: Abstract specification of structures and methods in symbolic mathematical computation. *Theoretical Computer Science*, 104:89–107, Elsevier Science Publisher, October 1992.

[19] Limongelli, C. and Temperini, M.: On the uniform representation of mathematical data structures. In A. Miola, editor, *Design and Implementation of Symbolic Computation Systems, International (Symposium Disco '93)*, *LNCS*, Vol. 722, Springer Verlag, 1993.

[20] Lipson, J. D.: *Elements of Algebra and Algebraic Computing.* Addison Wesley Publishing Company, 1988.

[21] Mignotte, M.: Some useful bounds. In B. Buchberger, G. E. Collins, and R. Loos, editors, *Computer Algebra Symbolic and Algebraic Computation*, pp. 259–263. Springer Verlag, 1983.

[22] Minc, H. and Marcus, M.: *Introduction to Linear Algebra*. Macmillan, New York, 1965.

[23] Miola, A.: Algebraic approach to p-adic conversion of rational numbers. *Information Processing Letters*, 18:167–171, 1984.

[24] Pirastu, R. and Siegl, K.: Parallel computation and indefinite summation: A ||MAPLE|| application for the rational case. to appear in Journal of Symbolic Computation, special issue on Symbolic Computation in Combinatorics Δ_1, Eds. P. Paule and V. Strehl, 1995.

[25] Siegl, K.: Parallelizing algorithms for symbolic computation using ||MAPLE||. In *Fourth ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming, San Diego*, pp. 179–186, 1993.

[26] Villard, G.: Parallel general solution of rational linear systems using p-adic expansions. In Barton Cosnard and Vanneschi, editors, *Procs of the IFIP WG 10.3 Working Conference on Parallel Processing*. Elsevier Sc.P., 1988.

[27] Villard, G.: *Symbolic computation and parallelism: solution of linear systems (in French)*. PhD thesis, Institut National Polytechnique de Grenoble, December 1988.