Computations with $p-$adic numbers in Maxima

JA Vallejo[a,*,1]

[a] Facultad de Ciencias, Universidad Autónoma de San Luis Potosí,
Av Chapultepec 1570 Col. Lomas del Pedregal
CP 78295 San Luis Potosí (SLP) México

ARTICLE INFO

Keywords:
$p-$adic numbers
Hensel codes
Finite-segment arithmetic
Maxima CAS
Symbolic computations

ABSTRACT

This is just a first attempt to create a Maxima package for working with $p-$adic numbers. It is extremely ugly and slow, lacking anything remotely resembling elegance or optimization, but at least it gives correct answers (for all those examples that I have been able to find in the literature). The current version is suitable for basic courses on the subject of $p-$adic analysis, and maybe for constructing examples of some simple theoretical ideas.

1. Loading the package

The most simple way to do it consists in putting a copy of `padics.mac` in your working directory. If a Maxima session is launched from that directory, you can load the package with

```
(%i1) load("padics.mac");
(%o1) "padics.mac"
```

The other option is to do a global installation, putting a copy of `padics.mac` in the system directory (you will need root permissions for that) `/usr/share/maxima/5.42.1/share/contrib` or its Windows equivalent. Then, load the package with the same command above.

2. The $p-$adic norm

We can compute the p-adic order of a rational number with `padic_order`. The syntax of all the commands in the package is quite intuitive; for example, to compute the order of a rational with respect to the prime p, we use `padic_order(rational,prime)`:

```
(%i2) padic_order(144,3);
(%o2) 2
(%i3) padic_order(17,3);
(%o3) 0
```

We follow the convention that the order of 0 is always (real) infinity:

```
(%i4) makelist(padic_order(0,i),i,[2,3,5,7,11,13]);
(%o4) [inf,inf,inf,inf,inf,inf]
(%i5) padic_order(3/10,5);
(%o5) -1
(%i6) padic_order(36015/88,7);
(%o6) 4
```

Notice that the $p-$adic order is an even function:

*Corresponding author
`jvallejo@fc.uaslp.mx` (J. Vallejo)
http://galia.fc.uaslp.mx/~jvallejo (J. Vallejo)
ORCID(s): 0000-0002-9508-1549 (J. Vallejo)
[1] Supported by a research grant from the Consejo Nacional de Ciencia y Tecnología (México) code A1-S-19428.

```
(%i7) padic_order(-3/10,5);
(%o7) -1
```

The reduction to the canonical form of a rational number

$$r = \frac{a}{b} = p^{\text{porder}(r)} \frac{a'}{b'}$$

can be achieved with `padic_canonical`. The result has the form of a list `[p^porder(r),a'/b']`:

```
(%i8) padic_canonical(0.234,2);
(%o8) [1/4,117/125]
(%i9) padic_canonical(0,3);
(%o9) [1,0]
```

The $p-$adic norm of a rational number is computed by `padic_norm`:

```
(%i10) makelist(padic_norm(0,j),j,[2,3,5,7,11,13,17]);
(%o10) [0,0,0,0,0,0,0]
(%i11) padic_norm(17,17);
(%o11) 1/17
(%i12) padic_norm(144,3);
(%o12) 1/9
(%i13) padic_norm(12,5);
(%o13) 1
(%i14) makelist(padic_norm(162/13,k),k,[3,13]);
(%o14) [1/81,13]
```

The next example comes from `http://mathworld.wolfram.com/p-adicNorm.html`:

```
(%i15) makelist(padic_norm(140/297,k),k,[2,3,5,7,11]);
(%o15) [1/4,27,1/5,1/7,11]
```

Another example, this one from `www.asiapacific-mathnews.com/03/0304/0001_0006.pdf`:

```
(%i16) makelist(padic_norm(63/550,k),k,[2,3,5,7,11,13]);
(%o16) [2,1/9,25,1/7,11,1]
(%i17) padic_norm(0.234,2);
(%o17) 4
```

Let us check the triangle *equality*:

```
(%i18) padic_norm(3/10,5);
(%o18) 5
(%i19) padic_norm(40,5);
(%o19) 1/5
(%i20) padic_norm(3/10-40,5);
(%o20) 5
```

The next examples come from `https://www.sangakoo.com/en/unit/p-adic-distance`:

```
(%i21) padic_norm(10/12,2);
(%o21) 2
(%i22) padic_norm(10/12,5);
(%o22) 1/5
(%i23) padic_norm(10/12,7);
(%o23) 1
```

Of course, once we have a norm available, we can define the associated $p-$adic distance, here denoted `padic_distance`. The first example computes the distance between 2 and 28814 in $\mathbb{Q}_7$ (so you can easily guess the syntax):

```
(%i24) padic_distance(2,28814,7);
(%o24) 1/2401
(%i25) padic_distance(2,3,7);
(%o25) 1
(%i26) padic_distance(2166^2,2,7);
(%o26) 1/2401
(%i27) padic_distance(3,3+29^4,29);
(%o27) 1/707281
```

The next example comes from https://www.sangakoo.com/en/unit/p-adic-distance:

```
(%i28) padic_distance(82,1,3);
(%o28) 1/81
```

3. p–adic Expansions (Hensel codes)

As in the case of real numbers, the only explicit representation of p–adic numbers we can get are those based on rational numbers (periodic p–adic expansions). Here we consider Hensel (pseudo)codes, which basically are truncations of the p–adic expansions to a given order r. The syntax is `hensel(rational,p,r)`, the output has the form `[[exponent],mantissa]`, and the algorithm used here is based on the one proposed by G. Bachman in [1].

```
(%i29) hensel(5/7,7,7);
(%o29) [[-1],5,0,0,0,0,0,0]
(%i30) hensel(-84,7,9);
(%o30) [[1],2,5,6,6,6,6,6,6,6]
(%i31) hensel(8/3,5,9);
(%o31) [[0],1,2,3,1,3,1,3,1,3]
(%i32) hensel(3/4,5,4);
(%o32) [[0],2,1,1,1]
(%i33) hensel(2/15,5,7);
(%o33) [[-1],4,1,3,1,3,1,3]
(%i34) hensel(7/6,5,4);
(%o34) [[0],2,4,0,4]
(%i35) hensel(2/7,5,4);
(%o35) [[0],1,2,1,4]
(%i36) hensel(1/12,5,7);
(%o36) [[0],3,4,2,4,2,4,2]
(%i37) hensel(5/8,5,4);
(%o37) [[1],2,4,1,4]
(%i38) hensel(1/2,5,4);
(%o38) [[0],3,2,2,2]
(%i39) hensel(1/3,5,4);
(%o39) [[0],2,3,1,3]
(%i40) hensel(1/4,5,4);
(%o40) [[0],4,3,3,3]
(%i41) hensel(1/4,5,7);
(%o41) [[0],4,3,3,3,3,3,3]
(%i42) hensel(1/25,5,4);
(%o42) [[-2],1,0,0,0]
(%i43) hensel(-7/8,3,5);
(%o43) [[0],1,2,1,2,1]
```

The command `nicehensel` displays the result in the form commonly found in textbooks and expository works (this form has the drawback that, when $p > 7$, is is impossible to distinguish between the number 11 and two consecutive

1's, so it will not be used in what follows), that is, something like

$$r = a_{-e}...a_{-1}.a_0 a_1 a_2...$$

where e is the order of r.

```
(%i44) nicehensel(8/3,5,9);
(%o44) .123131313
(%i45) nicehensel(8/75,5,9);
(%o45) 12.3131313
(%i46) nicehensel(3/4,5,4);
(%o46) .2111
(%i47) nicehensel(2/15,5,7);
(%o47) 4.131313
(%i48) nicehensel(1/3,5,7);
(%o48) .2313131
(%i49) nicehensel(-1/3,5,7);
(%o49) .3131313
(%i50) nicehensel(5/1,5,4);
(%o50) .0100
(%i51) nicehensel(25,5,4);
(%o51) .0010
(%i52) nicehensel(2/7,5,4);
(%o52) .1214
```

Let us compare this with the table presented in [5]:

```
(%i53) h[i,j]:=nicehensel(i/j,5,4)$
(%i54) genmatrix(h,17,17);
```

$$\begin{pmatrix}
.1000 & .3222 & .2313 & .4333 & 1.000 & .1404 & .3302 & .2414 & .4201 & 3.222 & .1332 & .3424 & .2034 & .4101 & 2.313 & .1234 & .3043 \\
.2000 & .1000 & .4131 & .3222 & 2.000 & .2313 & .1214 & .4333 & .3012 & 1.000 & .2120 & .1404 & .4014 & .3302 & 4.131 & .2414 & .1132 \\
.3000 & .4222 & .1000 & .2111 & 3.000 & .3222 & .4021 & .1303 & .2313 & 4.222 & .3403 & .4333 & .1143 & .2013 & 1.000 & .3104 & .4121 \\
.4000 & .2000 & .3313 & .1000 & 4.000 & .4131 & .2423 & .3222 & .1124 & 2.000 & .4240 & .2313 & .3123 & .1214 & 3.313 & .4333 & .2210 \\
.0100 & .0322 & .0231 & .0433 & .1000 & .0140 & .0330 & .0241 & .0420 & .3222 & .0133 & .0342 & .0203 & .0410 & .2313 & .0123 & .0304 \\
.1100 & .3000 & .2000 & .4222 & 1.100 & .1000 & .3142 & .2111 & .4131 & 3.000 & .1411 & .3222 & .2232 & .4021 & 2.000 & .1303 & .3342 \\
.2100 & .1322 & .4313 & .3111 & 2.100 & .2404 & .1000 & .4030 & .3432 & 1.322 & .2204 & .1202 & .4212 & .3222 & 4.313 & .2042 & .1431 \\
.3100 & .4000 & .1231 & .2000 & 3.100 & .3313 & .4302 & .1000 & .2243 & 4.000 & .3041 & .4131 & .1341 & .2423 & 1.231 & .3222 & .4420 \\
.4100 & .2322 & .3000 & .1433 & 4.100 & .4222 & .2214 & .3414 & .1000 & 2.322 & .4324 & .2111 & .3321 & .1134 & 3.000 & .4402 & .2024 \\
.0200 & .0100 & .0413 & .0322 & .2000 & .0231 & .0121 & .0433 & .0301 & .1000 & .0212 & .0140 & .0401 & .0330 & .4131 & .0241 & .0113 \\
.1200 & .3322 & .2231 & .4111 & 1.200 & .1140 & .3423 & .2303 & .4012 & 3.322 & .1000 & .3020 & .2430 & .4431 & 2.231 & .1421 & .3102 \\
.2200 & .1100 & .4000 & .3000 & 2.200 & .2000 & .1330 & .4222 & .3313 & 1.100 & .2332 & .1000 & .4410 & .3142 & 4.000 & .2111 & .1240 \\
.3200 & .4322 & .1413 & .2433 & 3.200 & .3404 & .4142 & .1241 & .2124 & 4.322 & .3120 & .4424 & .1000 & .2343 & 1.413 & .3340 & .4234 \\
.4200 & .2100 & .3231 & .1322 & 4.200 & .4313 & .2000 & .3111 & .1420 & 2.100 & .4403 & .2404 & .3034 & .1000 & 3.231 & .4030 & .2323 \\
.0300 & .0422 & .0100 & .0211 & .3000 & .0322 & .0402 & .0130 & .0231 & .4222 & .0340 & .0433 & .0114 & .0201 & .1000 & .0310 & .0412 \\
.1300 & .3100 & .2413 & .4000 & 1.300 & .1231 & .3214 & .2000 & .4432 & 3.100 & .1133 & .3313 & .2143 & .4302 & 2.413 & .1000 & .3401 \\
.2300 & .1422 & .4231 & .3433 & 2.300 & .2140 & .1121 & .4414 & .3243 & 1.422 & .2411 & .1342 & .4123 & .3013 & 4.231 & .2234 & .1000
\end{pmatrix}$$

4. Arithmetic with $p-$adics

The basic arithmetic functions are implemented as:

- `padic_sum` (for sum, addition)
- `padic_subtract` (for difference, subtraction)
- `padic_multiply` (for product, multiplication)
- `padic_divide` (for division, quotient)

The syntax is quite evident: each function takes the arguments `(operand1,operand2,p)`. Some test numbers:

```
(%i55) l1:hensel(3/10,5,4);
(l1) [[-1],4,2,2,2]
(%i56) l2:hensel(1/2,5,4);
(l2) [[0],3,2,2,2]
(%i57) padic_sum(l1,l2,5);
(%o57) [[-1],4,0,0,0]
```

Notice that the Hensel code for the sum $3/10 + 1/2$ corresponds to $4/5$:

```
(%i58) hensel(4/5,5,4);
(%o58) [[-1],4,0,0,0]
```

Also, notice that a consequence of using a finite segment representation is that adding up a really small number with a really big one just gives the bigger:

```
(%i59) padic_sum([[2],2,5,1,5],[[-3],3,3,3,2],7);
(%o59) [[-3],3,3,3,2]
```

Another test (this is an example in [5]) with $p = 5$ and $r = 9$

```
(%i60) h1:hensel(2/3,5,9);
(h1) [[0],4,1,3,1,3,1,3,1,3]
(%i61) h2:hensel(5/6,5,9);
(h2) [[1],1,4,0,4,0,4,0,4,0]
(%i62) padic_sum(h1,h2,5);
(%o62) [[0],4,2,2,2,2,2,2,2,2]
```

Let us check the following example in [5]:

```
(%i63) padic_subtract(h1,h2,5);
(%o63) [[0],4,0,4,0,4,0,4,0,4]
```

And those of [8]:

```
(%i64) padic_subtract(hensel(3/4,5,4),hensel(3/2,5,4),5);
(%o64) [[0],3,3,3,3]
```

An example on multiplication, also from [8]:

```
(%i65) t1:hensel(4/15,5,4);
(t1) [[-1],3,3,1,3]
(%i66) t2:hensel(5/2,5,4);
(t2) [[1],3,2,2,2]
(%i67) padic_multiply(t1,t2,5);
(%o67) [[0],4,1,3,1]
```

Another example from [5]:

```
(%i68) padic_multiply(h1,h2,5);
(%o68) [[1],4,2,0,1,2,4,3,2,0]
```

An example from [7]:

```
(%i69) al:hensel(1/4,5,4);
(al) [[0],4,3,3,3]
(%i70) be:hensel(1/3,5,4);
(be) [[0],2,3,1,3]
(%i71) padic_multiply(al,be,5);
(%o71) [[0],3,4,2,4]
```

The function `normalize_hensel` normalizes the Hensel code so that the first digit after the dot is not zero:

```
(%i72) normalize_hensel([[-1],0,0,1,2,3]);
(%o72) [[1],1,2,3]
```

It is internally used by the function for computing divisions. Our first example is a trivial one:

```
(%i73) padic_divide([[0],4,0,0,0,0,0,0],[[0],2,0,0,0,0,0,0],7);
(%o73) [[0],2,0,0,0,0,0,0]
```

The next example is from [5]:

```
(%i74) dividend:[[0],4,1,3,1,3,1,3];
(dividend) [[0],4,1,3,1,3,1,3]
(%i75) divisor:[[0],3,4,2,4,2,4,2];
(divisor) [[0],3,4,2,4,2,4,2]
(%i76) padic_dividei(dividend,divisor,5);
(%o76) [[0],3,1,0,0,0,0,0]
(%i77) d1:[[0],2,1,1,1];
(d1) [[0],2,1,1,1]
(%i78) d2:[[-1],1,1,0,0];
(d2) [[-1],1,1,0,0]
(%i79) padic_divide(d1,d2,5);
(%o79) [[1],2,4,1,4]
(%i80) padic_divide([[0],4,3,3,3],[[0],0,1,4,0],5);
(%o80) [[-2],0,4,2,2]
```

This is the example presented in [8]: $1/4/(1/2 + 1/3) + 1/25$

```
(%i81) padic_sum(padic_divide([[0],4,3,3,3],
                              padic_sum([[0],3,2,2,2],[[0],2,3,1,3],5)
                              ,5),
                 [[-2],1,0,0,0],
                 5);
(%o81) [[-2],1,4,2,2]
```

A quick check that the answer is correct:

```
(%i82) 1/4/(1/2+1/3)+1/25;
(%o82) 17/50
(%i83) hensel(17/50,5,4);
(%o83) [[-2],1,4,2,2]
```

Another example, from [7]:

```
(%i84) alf:hensel(8/9,5,4);
(alf) [[0],2,2,4,3]
(%i85) bet:hensel(1/2,5,4);
(bet) [[0],3,2,2,2]
(%i86) padic_divide(alf,bet,5);
(%o86) [[0],4,4,3,2]
```

We reproduce here one more example, from [10]:

```
(%i87) hensel(1/333333,5,27);
(%o87) [[0],2,4,4,4,4,4,2,1,2,3,4,4,1,2,3,1,0,1,2,3,2,3,1,3,1,0,2]
(%i88) time(%);
(%o88) [0.001]
```

Reference [10] states that 3 seconds were required for this computation, using a C++ library, in 2005.

5. From Hensel codes to rationals

Passing from Hensel codes to equivalent rational numbers in $\mathbb{Q}_p$ requires the use of the appropriate Farey fractions. A function for generating the Farey fractions $\mathbb{I}_n$ is `farey(n)`:

```
(%i89) farey(17);
(%o89) [0,1/17,1/16,1/15,1/14,1/13,1/12,1/11,1/10,1/9,2/17,1/8,2/15,1/7,
        2/13,1/6,3/17,2/11,3/16,1/5,3/14,2/9,3/13,4/17,1/4,4/15,3/11,2/7,
        5/17,3/10,4/13,5/16,1/3,6/17,5/14,4/11,3/8,5/13,2/5,7/17,5/12,3/7,
        7/16,4/9,5/11,6/13,7/15,8/17,1/2,9/17,8/15,7/13,6/11,5/9,9/16,4/7,
        7/12,10/17,3/5,8/13,5/8,7/11,9/14,11/17,2/3,11/16,9/13,7/10,12/17,
        5/7,8/11,11/15,3/4,13/17,10/13,7/9,11/14,4/5,13/16,9/11,14/17,5/6,
        11/13,6/7,13/15,7/8,15/17,8/9,9/10,10/11,11/12,12/13,13/14,14/15,
        15/16,16/17,1]
(%i90) time(%);
(%o90) [0.0]
```

The package implements the algorithm by Gregory and Krishnamurthy described in the book [4]. We have added a heuristic routine to detect a few particular cases in order to give cleaner results. For instance, cases such as `[[m],a0,0,0,0,...]`,`[[m],a0,p-1,p-1,p-1,...]` or `[[m],a0,a1,...,ak,0,0,...,0s]` with $s > k$.

The syntax is `hensel_to_farey(list,p)`, where `list` is a Hensel code, and p is the prime we are considering. Some trivial examples:

```
(%i91) hensel_to_farey([[2],3,0,0,0,0,0,0,0],7);
(%o91) 147
(%i92) hensel_to_farey([[0],4,4,4,4,4,4,4],5);
(%o92) -1
(%i93) hensel_to_farey([[0],3,3,3,3,3,3,3],5);
(%o93) -3/4
(%i94) hensel_to_farey([[0],0,0,0,0,0,0],5);
(%o94) 0
```

From [10] (pg 12):

```
(%i95) hensel_to_farey([[0],2,3,1,5],7);
(%o95) 9/43
```

From [4] (pp. 100 and ff):

```
(%i96) hensel_to_farey([[0],0,2,4,1],5);
(%o96) 5/8
(%i97) hensel_to_farey([[1],2,4,1,4],5);
(%o97) 5/8
(%i98) hensel_to_farey([[-1],3,2,2,2],5);
(%o98) 1/10
(%i99) hensel_to_farey([[0],3,2,2,2],5);
(%o99) 1/2
(%i100) hensel_to_farey([[0],2,3,1,3],5);
(%o100) 1/3
(%i101) hensel_to_farey([[0],0,6,0,6],7);
(%o101) -7/8
```

A quick check:

```
(%i102) hensel(1/3,5,4);
(%o102) [[0],2,3,1,3]
(%i103) hensel(-7/8,7,4);
(%o103) [[1],6,0,6,0]
```

Example from [11]:

```
(%i104) hensel_to_farey([[0],2,2,1,0],5);
(%o104) 4/17
(%i105) hensel_to_farey([[0],1,2,1,4],5);
(%o105) 2/7
```

Examples from the paper [6]:

```
(%i106) hensel_to_farey([[0],2,2,1,0],5);
(%o106) 4/17
(%i107) hensel_to_farey([[0],2,2,3,4],5);
(%o107) 17/16
(%i108) hensel_to_farey([[0],4,2,3,4],5);
(%o108) 13/17
(%i109) hensel_to_farey([[0],1,1,0,0,1,0],3);
(%o109) -13/17
(%i110) hensel_to_farey([[0],1,2,1,4],5);
(%o110) 2/7
(%i111) hensel_to_farey([[0],3,4,2,3],5);
(%o111) 11/7
```

6. Hensel codes of square roots

The computation of square roots in $\mathbb{Q}_p$ is a very interesting topic that depends on a lot of number-theoretical notions, among which that of a quadratic residue is the most important. An integer q is called a quadratic residue mod p if there exists an integer x such that $x^2 = q \mod p$.

Remark: If $p = 2$, every integer is a quadratic residue. If p is a prime different from 2, there are $(p-1)/2$ residues and $(p-1)/2$ non-residues in $\mathbb{F}_p - \{0\}$ (that is, the multiplicative group of $\mathbb{F}_p$ or $\mathbb{F}_p^*$).

As a consequence of the reciprocity law in Number Theory, we get the following criterion:

(a) If $p = 1 \mod 4$, then -1 is a quadratic residue mod p.
(b) If $p = 3 \mod 4$, then -1 is a non residue mod p. Notice that every prime is equivalent to 1 or 3 mod 4.

Euler's criterion for quadratic reciprocity states that (given $a \in \mathbb{Z}$ and p an odd prime) the Legendre symbol evaluates to $(a|p) = a^{(p-1)/2}$, and this equals $1 \mod p$ if a is a quadratic residue and $-1 \mod p$ if it is not.

First, we introduce a function `sqrtmod` to determine whether a given integer is a quadratic residue modulo p or not. For example, $a = 2$ is not a quadratic residue modulo $p = 5$:

```
(%i112) sqrtmod(2,5);
(%o112) "Not a quadratic residue"
```

But it is modulo $p = 7$:

```
(%i113) sqrtmod(2,7);
(%o113) [3,4]
```

If a is a quadratic residue modulo p with root x, then another root is given by the y such that $y = -x \mod p$. We compute the p-adic roots numerically, with the aid of Newton's method, using the command `padic_sqrt`, whose syntax is `padic_sqrt(number,p)`. The command admits an optional argument fixing the number of iterations to be done, as in `padic_sqrt(number,p,iterations)`.

```
(%i114) padic_sqrt(2,7);
(%o114) [215912063945802350977/152672884556058511392,
         2267891697076964737/1603641597827614272]
```

We can get the corresponding Hensel of the roots codes through

```
(%i115) map(lambda([u],hensel(u,7,9)),%);
(%o115) [[[0],3,1,2,6,1,2,1,2,4],[[0],4,5,4,0,5,4,5,4,2]]
```

Another example: the square root of 7 in $\mathbb{Q}_3$ (from [2]):

```
(%i116) padic_sqrt(7,3,3);
(%o116) [977/368,108497/41008]
```

The first fraction contains $3 \times 2 = 6$ exact digits to determine the square root of 7 in $\mathbb{Q}_3$. To get its corresponding Farey sequence it does not make sense to take more than 6 digits. Hence we do the following:

```
(%i117) map(lambda([u],hensel(u,3,6)),%);
(%o117) [[[0],1,1,1,0,2,0],[[0],2,1,1,2,0,2]]
(%i118) hensel_to_farey(%[1],3);
(%o118) 1/25
```

The result is not exactly the rational we started with, but it is very close in $\mathbb{Q}_3$:

```
(%i119) padic_distance(977/368,1/25,3);
(%o119) 1/2187
```

More examples:

```
(%i120) padic_sqrt(6,5)[1];
(%o120) 80746825394092993/32964753427463648
(%i121) hensel(%,5,4);
(%o121) [[0],1,3,0,4]
```

The results can be compared against those given in the on-line calculator http://www.numbertheory.org/php/p-adic.html:

```
(%i122) padic_sqrt(25,7);
(%o122) [552213837122886833247075521/110442767424206762611644736,5]
(%i123) map(lambda([u],hensel(u,7,8)),%);
(%o123) [[[0],2,6,6,6,6,6,6,6],[[0],5,0,0,0,0,0,0,0]]
```

Example from [3]:

```
(%i124) padic_sqrt(-2,3);
(%o124) [-28545857/22783264,28545857/22783264]
(%i125) hensel(%[1],3,8);
(%o125) [[0],1,1,2,0,0,2,0,1]
```

Again an example from [2], to see the influence of the choice of initial condition in Newton's iteration (notice how we fix the number of iterations as 3 here):

```
(%i126) padic_sqrt(1,3,3);
(%o126) [1,3281/3280]
(%i127) map(lambda([u],hensel(u,3,8)),%);
(%o127) [[[0],1,0,0,0,0,0,0,0],[[0],2,2,2,2,2,2,2,2]]
(%i128) map(lambda([u],hensel_to_farey(u,3)),%);
(%o128) [1,-1]
```

7. Solving p−adic systems of linear equations

This is a topic of fundamental importance in applications. The algorithm implemented here is Gaussian reduction, and in order to solve the problem $Ax = b$ we have the commands `padic_gauss` and `padic_backsub`. The first one triangularizes the system, and its syntax is `padic_gauss(B,p)`, where $B = A|b$ is the augmented matrix of the system (that is, the coefficient matrix A augmented with the column b of non homogeneous terms). The resulting triangular matrix is processed by `padic_backsub` to obtain the Hensel codes of the solution.

The following examples are from [7]:

```
(%i129) D:matrix([3,1,3,16],[1,3,1,8],[1,1,3,12])$
(%i130) padic_gauss(D,11);
(%o130) matrix(
[[[0],3,0,0,0],[[0],1,0,0,0],[[0],3,0,0,0],[[0],5,1,0,0]],
[[[0],0,0,0,0],[[0],10,3,7,3],[[0],0,0,0,0],[[0],10,3,7,3]],
[[[0],0,0,0,0],[[0],0,0,0,0],[[0],2,0,0,0],[[0],6,0,0,0]]
)
(%i131) padic_backsub(%,11);
(%o131) [[[0],2,0,0,0],[[0],1,0,0,0],[[0],3,0,0,0]]
```

By converting to Farey fractions, we get the rational form of the solutions:

```
(%i132) map(lambda([x],hensel_to_farey(x,11)),%);
(%o132) [2,1,3]
```

Another example:

```
(%i133) C:matrix([2,2,-1,5],[-3,0,2,-5],[4,-5,-1,0]);
(C) matrix(
[2,2,-1,5],
[-3,0,2,-5],
[4,-5,-1,0]
)
(%i134) padic_gauss(C,5);
(%o134) matrix(
[[[0],2,0,0,0,0,0],[[0],2,0,0,0,0,0],[[0],4,4,4,4,4,4],[[1],1,0,0,0,0,0]],
[[[0],0,0,0,0,0,0],[[0],3,0,0,0,0,0],[[0],3,2,2,2,2,2],[[1],3,2,2,2,2,2]],
[[[0],0,0,0,0,0,0],[[0],0,0,0,0,0,0],[[0],0,3,2,2,2,2],[[0],0,2,2,2,2,2]]
)
(%i135) padic_backsub(%,5);
(%o135) [[[-2],0,0,1,0,0,0],[[-2],0,0,1,0,0,0],[[-2],0,0,4,4,4,4]]
```

We convert the solution to rational (Farey) expressions:

```
(%i136) map(lambda([x],hensel_to_farey(x,5)),%);
(%o136) [1,1,-1]
```

The examples above were trivial, the only intention was to show how the usual (rational) results are recovered within p−adic algebra. Now we consider more advanced examples, with some matrices that are highly unstable in rational arithmetic. As the output are really big expressions, we suppress them adding a dollar symbol at the end of the commands:

```
(%i137) E:matrix(
[10,9,8,7,6,5,4,3,2,1,1],
[9,9,8,7,6,5,4,3,2,1,2],
[8,8,8,7,6,5,4,3,2,1,-5],
[7,7,7,7,6,5,4,3,2,1,9],
[6,6,6,6,6,5,4,3,2,1,15],
[5,5,5,5,5,5,4,3,2,1,1],
[4,4,4,4,4,4,4,3,2,1,6],
[3,3,3,3,3,3,3,3,2,1,14],
[2,2,2,2,2,2,2,2,2,1,3],
[1,1,1,1,1,1,1,1,1,1,1]
)$
(%i138) padic_gauss(E,8209)$
(%i139) time(%);
(%o139) [1.038]
(%i140) padic_backsub(%th(2),8209)$
(%i141) map(lambda([x],hensel_to_farey(x,8209)),%);
(%o141) [-1,8,-21,8,20,-19,-3,19,-9,-1]
```

Of course, the solution we have obtained is exactly the same that results when using a built-in command such as `solve` or `linsolve`. This is so because the coefficients of E are rational. However, even restricting ourselves to computations with rationals, there are cases where the use of p−adic arithmetic can be convenient. A paramount example of this is provided by the Hilbert matrices, known for their bad condition number. The main consequence of this fact is that the *numerical* solutions to linear systems of the form $Hx = b$ (where H is a matrix with high condition number) do not depend continuously on the coefficients of H: If x_0 is a solution to $Hx = b_0$ and x_1 is a solution to $Hx = b_1$, where b_0 and b_1 are close in some vector norm, x_0 and x_1 are *not* necessarily close in that norm. In what follows, we focus our attention on camputations with Hilbert matrices. For example:

```
(%i142) F:addcol(hilbert_matrix(5),[137/60,87/60,459/420,743/840,1875/2520]);
(F) matrix(
[1,1/2,1/3,1/4,1/5,137/60],
[1/2,1/3,1/4,1/5,1/6,29/20],
[1/3,1/4,1/5,1/6,1/7,153/140],
[1/4,1/5,1/6,1/7,1/8,743/840],
[1/5,1/6,1/7,1/8,1/9,125/168]
)
(%i143) padic_gauss(F,8209)$
(%i144) time(%);
(%o144) [0.161]
(%i145) padic_backsub(%th(2),8209);
(%o145) [[[0],0,0,0,0,0,0,0,0],[[0],21,0,0,0,0,0,0,0],
         [[0],8120,8208,8208,8208,8208,8208,8208,8208],
         [[0],141,0,0,0,0,0,0,0],[[0],8140,8208,8208,8208,8208,8208,8208,8208]]
(%i146) map(lambda([x],hensel_to_farey(x,8209)),%);
(%o146) [0,21,-89,141,-69]
```

Observe that `padic_gauss` automatically chooses the number t of digits in the Hensel codes of the solution. There will be a lot of situatons where our simple heuristics for choosing t will lead to bad values. For those cases, one can manually choose t as another argument, using the alternative function `padic_gauss2`. Below we choose 8 digits to solve the system defined by the matrix E defined above, getting the same result as before:

```
(%i147) padic_gauss2(E,8209,8)$
(%i148) padic_backsub(%,8209)$
(%i149) map(lambda([x],hensel_to_farey(x,8209)),%);
(%o149) [-1,8,-21,8,20,-19,-3,19,-9,-1]
```

Let us see how p-adic arithmetics deal with the high condition number of Hilbert matrices and the associated instabilities:

```
(%i150) M1:addcol(hilbert_matrix(5),makelist(j,j,1,5));
(M1) matrix(
[1,1/2,1/3,1/4,1/5,1],
[1/2,1/3,1/4,1/5,1/6,2],
[1/3,1/4,1/5,1/6,1/7,3],
[1/4,1/5,1/6,1/7,1/8,4],
[1/5,1/6,1/7,1/8,1/9,5]
)
(%i151) M2:addcol(hilbert_matrix(5),makelist(j+29^(3+random(6)),j,1,5));
(M2) matrix(
[1,1/2,1/3,1/4,1/5,20511150],
[1/2,1/3,1/4,1/5,1/6,24391],
[1/3,1/4,1/5,1/6,1/7,20511152],
[1/4,1/5,1/6,1/7,1/8,500246412965],
[1/5,1/6,1/7,1/8,1/9,17249876314]
)
```

Notice that the non-homogeneous coefficients in M_2 are really close to the initial ones in M_1:

```
(%i152) makelist(padic_distance(M1[j][6],M2[j][6],29),j,1,length(M1));
(%o152) [1/20511149,1/24389,1/20511149,1/500246412961,1/17249876309]
```

In this case, the output has a size small enough to be displayed:

```
(%i153) padic_gauss2(M1,29,6);
```

$$\begin{pmatrix}
[[0],1,0,0,0,0,0] & [[0],15,14,14,14,14,14] & [[0],10,19,9,19,9,19] & [[0],22,21,21,21,21,21] & [[0],6,23,5,23,5,23] & [[0],1,0,0,0,0,1] \\
[[0],0,0,0,0,0,0] & [[0],17,26,16,26,16,26] & [[0],17,26,16,26,16,26] & [[0],24,26,23,26,23,26] & [[0],2,27,1,27,1,27] & [[0],16,14,14,15,14,28] \\
[[0],0,0,0,0,0,0] & [[0],0,0,0,0,0,0] & [[0],5,19,14,9,24,28] & [[0],22,28,21,28,21,28] & [[0],21,28,20,28,20,28] & [[0],6,24,4,23,4,1] \\
[[0],0,0,0,0,0,0] & [[0],0,0,0,0,0,0] & [[0],0,0,0,0,0,0] & [[0],20,8,11,17,2,26] & [[0],11,17,22,5,5,23] & [[0],5,10,4,28,15,24] \\
[[0],0,0,0,0,0,0] & [[0],0,0,0,0,0,0] & [[0],0,0,0,0,0,0] & [[0],0,0,0,0,0,0] & [[0],16,0,3,1,19,1] & [[0],9,20,8,28,16,9]
\end{pmatrix}$$

```
(%i157) padic_backsub(%,29);
(%o157) [[[0],9,4,0,19,18,1],[[0],20,16,25,14,20,3],
         [[0],19,6,17,8,15,19],[[0],10,20,28,24,27,18],
         [[0],6,21,15,15,0,11]]
(%i158) map(lambda([x],hensel_to_farey(x,29)),%);
(%o158) [1/73774969,2/122364169,-4725/9013,1/83362185,2/80199829]
```

The solutions of the original system and the perturbed one, are quite close (in the $p-$adic distance, of course):

```
(%i159) makelist(padic_distance(%[j],%th(4)[j],29),j,1,length(%));
(%o159) [1/24389,1/24389,1/24389,1/24389,1/24389]
```

For comparison, let us see how things work in the purely rational case:

```
(%i160) A:hilbert_matrix(5)$
(%i161) X:makelist(x[k],k,1,5)$
(%i162) B1:makelist(k,k,1,5)$
(%i163) B2:makelist(rat(k+random(0.01)),k,1,5)$
(%i164) for j:1 thru 5 do eq1[j]:sum(A[j,k]*X[k],k,1,5)=B1[j]$
(%i165) for j:1 thru 5 do eq2[j]:sum(A[j,k]*X[k],k,1,5)=B2[j]$
(%i166) solve(makelist(eq1[j],j,1,5),X);
(%o166) [[x[1]=125,x[2]=-2880,x[3]=14490,x[4]=-24640,x[5]=13230]]
(%i167) solve(makelist(eq2[j],j,1,5),X),numer;
(%o167) [[x[1]=120.1003883272046,x[2]=-2780.48089521768,
          x[3]=14040.16509590324,x[4]=-23939.74158066582,
          x[5]=12880.11148768633]]
```

The solutions (%o166) and (%o167) are far away from each other in the usual absolute value distance. This example is a striking evidence for the better stability properties of $p-$adic arithmetic over the traditional rational one (see [4, 7, 8, 9, 10, 11]).

References

[1] G. Bachman: *Introduction to p−adic Numbers and Valuation Theory*, Academic Press, New York, NY, 1964.

[2] K. Conrad: *Hensel's Lemma*. Available at https://kconrad.math.uconn.edu/blurbs/gradnumthy/hensel.pdf.

[3] J. E. Cremona: *Introduction to Number Theory Lecture Notes*, 2018. Available at http://homepages.warwick.ac.uk/staff/J.E.Cremona/courses/MA257/ma257.pdf.

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

[5] C. K. Koc: *A Tutorial on p−adic Arithmetic*. Oregon State University. Technical Report, April 2002. Available at http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.81.2931&rep=rep1&type=pdf.

[6] V. E. Krishnamurthy : *On the Conversion of Hensel Codes to Farey Rationals*. IEEE Transactions on computers, **C32** 4 (1983).

[7] E. V. Krishnamurty, T. Mahadeva Rao and K. Subramanian: *p−Adic arithmetic procedures for exact matrix computations*. Proc, Indian Acad. Sci., Vol. **82A** 5 (1975) 165–175.

[8] C. Limongelli and R. Pirastu: *p−adic Arithmetic and Parallel Symbolic Computation: An Implementation for Solving Linear Systems*. Computers and Artificial Intelligence **14** 1 (1996) 35–62.

[9] C. Limongelli: *On an efficient algorithm for big rational number computations by parallel p−adics*. J. Symbolic Computation **15** (1993) 181–197.

[10] C. Lu: *A Computational Library Using p-adic Arithmetic for Exact Computation With Rational Numbers in Quantum Computing*. Final Report (Grant #FA9550-05-1-0363) Towson University, 2005. Available at https://archive.org/details/DTIC_ADA456488/page/n5.

[11] A. Vimawala: *p−Adic Arithmetic Methods for Exact Computations of Rational Numbers*. School of Electrical Engineering and Computer Science, Oregon State University. Available at: http://cs.ucsb.edu/koc/cs290g/project/2003/vimawala.pdf.