Exact p-adic computation

Tim Dokchitser (Bristol)

23 September 2020

p-adic numbers

$\mathbb{Q} \hookrightarrow \quad \mathbb{R} =$ completion of $\mathbb{Q}$ under the usual absolute value,

$$|x|_\infty = \begin{cases} x, & x \geq 0 \\ -x, & x < 0 \end{cases}.$$

For every prime p,

$\mathbb{Q} \hookrightarrow \quad \mathbb{Q}_p =$ completion of $\mathbb{Q}$ under the p-adic absolute value,

$$|p^n \tfrac{a}{b}|_p = p^{-n} \text{ if } p \nmid ab.$$

These are the only absolute values on $\mathbb{Q}$ (non-trivial, up to equvalence).
$\mathbb{R}, \mathbb{Q}_2, \mathbb{Q}_3, \mathbb{Q}_5, \ldots$ are all complete uncountable fields, good for analysis.

Example

In every field complete with respect to an absolute value,

$$1 + x + x^2 + x^3 + \ldots = \frac{1}{1-x}, \qquad |x| < 1$$

$$\begin{array}{lcll} 1 + 1/2 + 1/4 + 1/8 + \ldots & = & \frac{1}{1-\frac{1}{2}} & = \quad 2 \qquad \text{in } \mathbb{R} \\ 1 + 2 + 4 + 8 + \ldots & = & \frac{1}{1-2} & = \quad -1 \qquad \text{in } \mathbb{Q}_2 \end{array}$$

Applications & requirements

Replacing real analysis by p-adic analysis proved fruitful, especially in number theory. And easier because of the strong triangle inequality

$$|a_1 + ... + a_n| \leqslant |a_1| + ... + |a_n|.$$

Applications in number theory:

- Prime decomposition in number fields
- Factoring polynomials over the integers (LLL)
- Computing Galois groups over $\mathbb{Q}$
- Chabauty-Coleman method for finding rational points on curves
- Point-counting algorithms based on p-adic cohomology
- Galois representations, Conductors (with V. Dokchitser, C. Doris)

Tools required:

$\mathbb{Q}_p$ and their finite extensions	($\leftrightarrow \mathbb{C}$)
Hensel's lemma	($\leftrightarrow$ Newton-Raphson)
Factorisation & root finding	

Current implementations (see Caruso's notes on p-adics)

Implemented in Maple, Pari-GP, Flint, Nemo, Sage, Magma, ... ↰
+ Mathemagix (lazy) essentially floating point

Example ($\sqrt{2}$)

In $\mathbb{R}$

$$\sqrt{2} \approx 1.4142135624 \qquad (\times 10^0)$$

In $\mathbb{Q}_7$

$$\sqrt{2} \approx 4+5\cdot 7+4\cdot 7^2+5\cdot 7^4+4\cdot 7^5+5\cdot 7^6+4\cdot 7^7+2\cdot 7^8+O(7^{10})$$

Example (Typical precision loss in $\mathbb{R}$)

$$e^{-20} \approx \sum_{n=0}^{100} \frac{(-20)^n}{n!} \begin{array}{lll} = & -0.22001502 & \text{(8 digits)} \\ = & 4.10277101181578\ \text{E-9} & \text{(16 digits)} \\ = & 2.061153622438...\ \text{E-9} & \text{(32 digits)} \\ = & 2.061153622438...\ \text{E-9} & \text{(64 digits)} \end{array}$$

See e.g. Platt, *Isolating some non-trivial zeros of zeta*, 2017.

Precision loss in p-adics

```
R<x>:=PolynomialRing(pAdicField(2,20));
Factorisation((x-8)*(x-16)*(x-32)*(x-64));
  Factorization: Insufficient precision in factorization of a
  non-squarefree polynomial.

R<x>:=PolynomialRing(pAdicField(2,23));
Factorisation((x-8)*(x-16)*(x-32)*(x-64));
  (x^2 + (9531*2^8 + O(2^23))*x - 4039*2^9 + O(2^23))  *
  (x - 305009*2^3 + O(2^23))^2

R<x>:=PolynomialRing(pAdicField(2,26));
Factorisation((x-8)*(x-16)*(x-32)*(x-64));
  (x - 92161*2^6 + O(2^26))  *
  (x + 77823*2^5 + O(2^26))  *
  (x + 16383*2^4 + O(2^26))  *
  (x + 393215*2^3 + O(2^26))
```

New: lazy exact p-adic arithmetic

Christopher Doris: package for exact p-adics in Magma

arxiv: 2008.11063

- Standalone

https://github.com/cjdoris/ExactpAdics2

- Now implemented in C, to be released soon as part of Magma.

```
R<x>:=PolynomialRing(pAdicField(2: Exact));
Factorisation((x-8)*(x-16)*(x-32)*(x-64));
  1 + O(2^20))*x + -2^6 + O(2^26)  *
  1 + O(2^20))*x + -2^5 + O(2^25)  *
  1 + O(2^20))*x + -2^4 + O(2^24)  *
  1 + O(2^20))*x + -2^3 + O(2^23)  *
```

Functionality in Magma

Basic arithmetic, root finding, factorisation, arbitrary extensions of $\mathbb{Q}_p$.

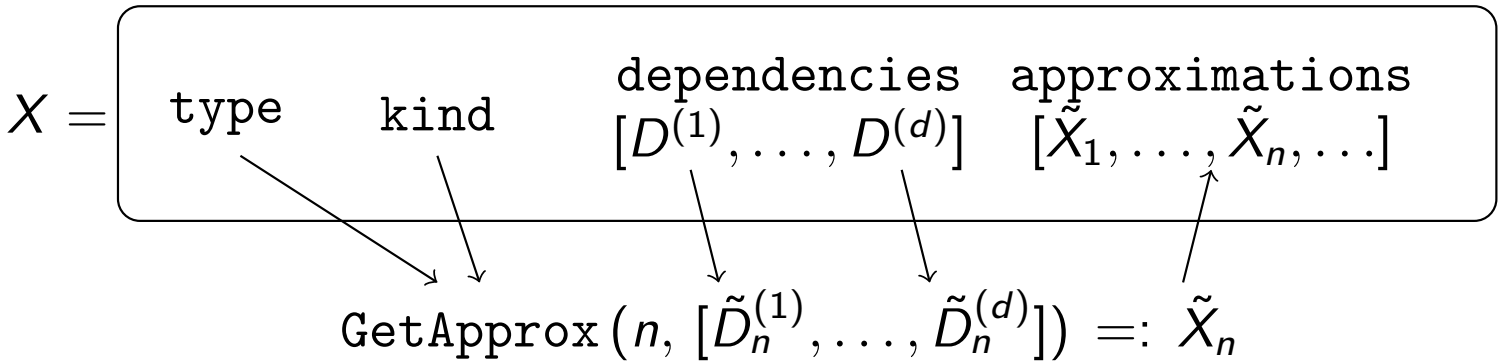
```
R<x>:=PolynomialRing(pAdicField(2,26));
r:=Roots(x^24-33)[1,1];r;
  532917 + O(2^23)
r-532917;
  O(2^23)

R<x>:=PolynomialRing(pAdicField(2: Exact));
r:=Roots(x^24-33)[1,1];r;
  532917 + O(2^20)
r-532917;
  -298987*2^23 + O(2^43)

Factorisation(x^4+2*x+r-532917: Extensions:=true);
... (one trivial extension, and one degree 3 totally ramified)
```

Implementation

- Epoch $n \in \mathbb{N}$ controls precision (2^n p-adic digits of $\mathbb{Q}_p$).
- All p-adic objects keep track of dependencies.



- Type: Magma type (FldXPad, FldXPadElt, ...)
- Kind: "p-adic number coerced from $\mathbb{Z}$",
  "sum of two elements from the same field",
  "unramified field extension", ...
- Type + Kind $\Rightarrow$ `GetApprox` implementation
- Every element X keeps a cache $[\tilde{X}_1, ..., \tilde{X}_n]$ of approximations computed so far.