p-Adics in SAGE

David Roe

Department of Mathematics
Harvard University

Sage Days 4

Outline

1. Definitions
   - The Mathematical Objects
   - Computer Representations
2. Implementation
   - Classes
   - Lessons
   - Demo
3. Future
   - Current Status
   - p-Adic Matrices and Polynomials

Outline

1. Definitions
   - The Mathematical Objects
   - Computer Representations
2. Implementation
   - Classes
   - Lessons
   - Demo
3. Future
   - Current Status
   - p-Adic Matrices and Polynomials

Outline

1. Definitions
   - The Mathematical Objects
   - Computer Representations
2. Implementation
   - Classes
   - Lessons
   - Demo
3. Future
   - Current Status
   - p-Adic Matrices and Polynomials

$\mathbb{Z}_p$ and $\mathbb{Q}_p$

$\mathbb{Z}_p$ is a ring, whose elements can be thought of as

- A power series in p, i.e. $a_0 + a_1 p + \cdots + a_n p^n + \cdots$, or
- A sequence of elements of $\mathbb{Z}/p^n\mathbb{Z}$, chosen consistently.

$\mathbb{Q}_p$ is the fraction field of $\mathbb{Z}_p$. A p-adic rational is then:

- A Laurent series in p, i.e. $a_k p^k + a_{k+1} p^{k+1} + \cdots$.
- A power of p times a p-adic integer.

Basic operations on p-adics

Say $x \in \mathbb{Q}_p$

- The valuation of x, $v_p(x)$ is the largest power of p dividing x.
  - $x \in \mathbb{Z}_p$ if and only if $v_p(x) \geq 0$.
  - If $x \in \mathbb{Z}_p$, $v_p(x)$ is the largest power of $p\mathbb{Z}_p$ containing x.
- The unit part of x is a $u \in \mathbb{Z}_p^\times$ with $x = p^{v_p(x)}u$.
- There is a map $\mathbb{Z}_p \to \mathbb{Z}/p^n\mathbb{Z}$ for all n defined by reduction modulo $p^n\mathbb{Z}_p$.
- $\mathbb{Z}$ and $\mathbb{Q}$ sit inside $\mathbb{Z}_p$ and $\mathbb{Q}_p$ respectively.

Precision

Sage distinguishes two types of precision:

- The absolute precision, $a(x)$, is the power of p modulo which that element is defined.
- The relative precision, $r(x)$, is the precision of the unit part.

If $x, y \in \mathbb{Q}_p$

- $a(x) = r(x) + v_p(x)$
- $a(x+y) = a(x-y) = \min(a(x), a(y))$
- $r(xy) = r(x/y) = \min(r(x), r(y))$
- $v_p(xy) = v_p(x) + v_p(y)$
- $v_p(x \pm y) \geq \min(v_p(x), v_p(y))$
- $v_p(x \pm y) = \min(v_p(x), v_p(y))$ if $v_p(x) \neq v_p(y)$

Types

There are four basic types of p-adic rings R in SAGE.

- Capped Relative: relative precision bounded by $c(R)$.
- Capped Absolute: absolute precision bounded by $c(R)$.
- Fixed Modulus: absolute precision bounded by $c(R)$, no tracking.
- Lazy: no precision bounds; elements can raise their precision.

Types

There are two types of p-adic fields.

- Capped Relative.
- Lazy.

Extensions

One can create field extensions of $\mathbb{Q}_p$, defined by some monic polynomial f.
We classify the extension based on f.

- f is unramified if it remains irreducible passing to $\mathbb{F}_p$
- $f = x^N + \cdots + a_0$ is eisenstein if $p \mid a_i$ for $0 \leq i \leq N - 1$ and $p^2 \nmid a_0$.
- Otherwise we factor our extension.

The Class Heriarchy

This is what the class heirarchy looks like. . .

- Each file contains a unique class.
- Each parent should have at most two superclasses. Each element should have one.
- The class for elements depends only on the p-adic type and the kind of extension (not ring versus field).
- I tried to keep the heirarchy in a state where I can generalize the work done on p-adics to power series.

The Class Heriarchy

This is what the class heirarchy looks like. . .

- Each file contains a unique class.
- Each parent should have at most two superclasses. Each element should have one.
- The class for elements depends only on the p-adic type and the kind of extension (not ring versus field).
- I tried to keep the heirarchy in a state where I can generalize the work done on p-adics to power series.

Improvement Process

- Write down a framework: the classes and the interface.
- Implement functionality in Python, starting with the basic classes.
- Have other people write doctests.
- Change base classes over to SageX.
- Work on polynomials and matrices

Some Internals

It turns out that one wants to avoid power series as much as possible. For the base rings, we store an integer (actually an `mpz_t`), a precision and sometimes a valuation.
For extensions, elements are stored as polynomials with p-adic coefficients. We restrict the precisions of the coefficients to optimize the arithmetic for extensions.

What I Learned

- Design the class heirarchy and interface first, consulting other systems. Fix them early.
- Do a better job dividing up the work, and making the design sufficiently modular to do this.
- One class per file. Good file/classs names.
- Python first, then SageX is amazing.
- I need to do a better job with doctests.
- Only use one reversion control system.

Live Demo!

Current Status and Next Steps

- Polynomials over *p*-adics need work
- Currently extensions are disabled because the move to SageX in the base classes broke them.
- Want to implement round4 natively
- Unify number field functionality and *p*-adic functionality
- Take ideas for *p*-adics and port them to power series.

Matrices and Polynomials

- Represent matrices and polynomials as a common valuation, an integer matrix or polynomial, and a set of precision data.
- Find algorithms to determine the precision information of the answer separately from finding the answer.