Precision models for arithmetic in local fields

David Roe
(joint with Xavier Caruso)

Department of Mathematics
University of Calgary

March 1, 2012

Outline

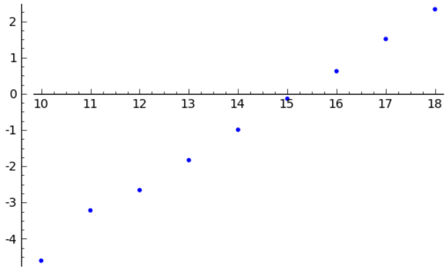
Characteristic polynomials in Magma

Disclaimer: I develop p-adics in Sage, an open source competitor to Magma. Moreover, I haven't run either of the following experiments personally, but rely on the experience of Kiran Kedlaya and Justin Walker. I give this example not to denigrate Magma but rather to motivate the solutions I propose in the remainder of the talk.

Suppose you create a random 50×50 matrix A over $\mathbb{Z}_p$ in Magma 2.15 and ask for its characteristic polynomial. Even if the entries of A have precision 100, the resulting polynomial will have very few digits of precision remaining.

Characteristic polynomials in Magma

In order to resolve this precision loss, Magma's characteristic polynomial algorithm appears to have changed in the most recent version (2.18). While the precision behavior has improved dramatically, the runtime has regressed. In the following timing graph, the horizontal axis gives the size of the matrix and the vertical axis gives the log of the time to compute the characteristic polynomial.



Karatsuba

Consider the Karatsuba algorithm for multiplying polynomials. Given input polynomials of degree $2n$

$$\begin{aligned} F &= F_0 + x^n F_1 \\ G &= G_0 + x^n G_1, \end{aligned}$$

we set

$$\begin{aligned} H_0 &= F_0 \cdot G_0 \\ H_2 &= F_1 \cdot G_1 \\ H_1 &= (F_0 + F_1) \cdot (G_0 + G_1) - H_0 - H_2. \end{aligned}$$

Then

$$F \cdot G = H_0 + H_1 x^n + H_2 x^{2n}$$

but we have only used three polynomial multiplications of degree n rather than four, at the cost of two additions.

Numerical stability

But suppose that the coefficients of F and G are p-adic and known to finite precision. For example, if

$$F = G = (1 + O(p^4)) + (p^4 + O(p^8)) \cdot x,$$

we would set

$$\begin{aligned} H_1 &= \left(1 + O(p^4) + p^4 + O(p^8)\right)^2 - \left(1 + O(p^4)\right)^2 - \left(p^4 + O(p^8)\right)^2 \\ &= \left(1 + O(p^4)\right) - \left(1 + O(p^4)\right) - \left(p^8 + O(p^{12})\right) \\ &= O(p^4). \end{aligned}$$

But the actual coefficient of x in the product has more precision:

$$2 \cdot \left(1 + O(p^4)\right) \cdot \left(p^4 + O(p^8)\right) = 2p^4 + O(p^8).$$

Precision versus speed

We're thus presented with a choice:

- Use faster algorithms, but sacrifice precision on some (or all) inputs,
- Use naive algorithms in order to retain as much precision as possible, but sacrifice speed, making computations with large inputs infeasible.

Of course, some non-naive algorithms will have better precision behavior than others. Over archimedean fields, a lot of effort has been devoted to finding fast algorithms that are numerically stable; many of these approaches can be adapted to the non-archimedean context. But even numerically stable algorithms frequently involve more precision loss than the naive algorithms.

Why we track precision

Recording the precision at each step of an algorithm imposes an overhead to computations with p-adics. We track precision for two main reasons:

- As a convenience to the user, so that they know the accuracy of the output of the algorithm.
- So that the algorithm can perform inexact operations (such as inversion, p-adic logarithms and exponentials) to an appropriate working precision so that the final answer has sufficient accuracy.

Separation

Separate the approximation from the precision

By separating the precision of compound *p*-adic data types from the approximation, algorithms can break the dichotomy of precision vs speed by using fast algorithms to compute an approximation and computing the precision of the result separately.

This separation is possible because precision in the non-archimedian world behaves far better than in archimedian computations: the ultrametric gives us far better precision control than the triangle inequality.

Notation

- $\mathcal{O}_K$ – complete discrete valuation ring. We will occasionally suppose that $\mathcal{O}_K$ has finite residue field.
- π – a uniformizer of $\mathcal{O}_K$.
- K – the field of fractions of $\mathcal{O}_K$.
- v – the valuation on K, normalized so that $v(\pi) = 1$.

In order to store elements of K in a finite amount of space, we require that K contain a dense subring k whose elements are simpler to represent.

- If $K = \mathbb{Q}_p$ then we may choose $k = \mathbb{Q}$.
- If $K = \mathbb{F}_q((t))$ then $k = \mathbb{F}_q(t)$ is dense in K.

Precision

Suppose that $a \in k$ is an approximation to $x \in K$. We say that a has *absolute precision* n if

$$v(x - a) \geq n.$$

We say that a has *relative precision* n if

$$v(a/x - 1) \geq n.$$

- If a is an approximation of x with absolute precision n and b of y with absolute precision m then $a + b$ is an approximation of $x + y$ with absolute precision $\min(n, m)$.
- If a is an approximation of x with relative precision n and b of y with relative precision m then ab is an approximation of xy with relative precision $\min(n, m)$.

Imprecise elements

An *approximate element* of K is a pair (a, m), where $a \in k$ and $m \in \mathbb{Z}$. We think of (a, m) as representing all possible elements $x \in K$ with $v(x - a) \geq m$, and write

$$a + O(\pi^m).$$

Geometrically, this approximate element is a ball of radius p^{-m} around a. While there is no distinguished center of such a ball, in practice we can fix for each precision m a set of distinguished elements of k that are inequivalent modulo π^m. For example, if $K = \mathbb{Q}_p$ and $k = \mathbb{Q}$, then we may choose

$$\{a/p^n : n \geq 0 \text{ and } 0 \leq a < p^{m+n}\}.$$

Precision loss in basic arithmetic

Precision loss occurs when addition and multiplication are mixed for elements with different valuations. For example, for odd p

$$\frac{\left(\left(1 + p^{99} + O(p^{100})\right) + \left(-1 + p^{99} + O(p^{100})\right)\right)}{p^{99} + O(p^{1000})} = 2 + O(p).$$

Unfortunately, mixing addition and multiplication is necessary for solving most problems.

Different precision models

- For $\mathbb{Z}_p$ and $\mathbb{Q}_p$, Sage currently supports three different precision models (capped relative, capped absolute and fixed modulus). But in reality these different models correspond more closely to the underlying data structure rather than the precision tracking, since there's only one shape of disc in $\mathbb{Q}_p$.
- For more complicated structures such as vectors, matrices, polynomials and power series, there are many different precision shapes possible. Different precision shapes have different tradeoffs between speed and accuracy.

Vector Precision

An element of K^n could be represented as

- a list of n approximate elements of K (together with the specification of a distinguished basis),
- a ball of radius p^{-m} around an element of k^n,
- an element of k^n together with an $\mathcal{O}_K$-lattice $P \subset K^n$.

Note that the third option generalizes the first two, and that we can always choose a basis for P consisting of vectors in k^n. In fact, P is determined exactly, without need for approximation.

Polynomial Precision

If we fix the degree n then $\{1, x, \ldots, x^n\}$ provides a distinguished basis for the space of polynomials of degree n as a K vector space. Some additional precision shapes have particular utility for polynomials:

- Newton polygons. If we consider a polynomial as a function from K to K, then having lots of extra precision in an "interior" coefficient does not add to the precision of any evaluation. Moreover, one can determine the Newton polygon of a product easily from the slopes of the input polygons, simplifying computation of the precision.
- Lagrange precision. We can give the precision of $f(a_i)$ for some fixed set of $a_i \in K$.
- A mixture, involving the specification of various derivatives at various points.

Matrix Precision

Similarly, the space of $m \times n$ matrices is isomorphic to K^{mn}, so precision types for vectors apply to matrices as well. We have extra precision shapes as well.

- If we consider a matrix as representing a linear map $K^n \to K^m$ then the image vectors will be defined with some precision lattice in K^m. This yields a “column precision” on our matrix, where each column has the same precision.
- Similarly, we can consider a “row precision,” where the rows of a matrix all have the same precision.

Power series

- The space $K[[x]]$ is no longer finite dimensional over K, so additional complications arise: you need to truncate series both in the x and p "variables." Precision types similar to those for finite dimensional vector spaces make sense however.
- Some precision structures for power series may specify an infinite Newton polygon symbolically. When this Newton polygon has positive slope in the limit then it allows a rigorous computation of the precision of power series evaluation. Information about all coefficients is necessary in order to ensure convergence, and needs to be stored separately from the finite list of approximate coefficients. A precision structure gives a natural place to reason with this information.

Precision on varieties

If V is a variety defined over K then points on V are another type of inexact object we might want to work with. The Grassmanian $G(m, n)$ of m-dimensional subspaces of K^n and an elliptic curve E defined over K provide illustrative examples.

- The Grassmanian $G(m, n)$ is covered by affine charts, each isomorphic to $K^{m(n-m)}$. To specify a point we need to specify a chart, and then a point in the relevant vector space.
- An elliptic curve, on the other hand, is usually given as a subvariety of projective space, and we specify points in projective space that are supposed to lie on the curve. We can give the precision of such an approximate point as a lattice in the tangent space.

Applying Functions to Precisions

Theorem

Suppose $x \in K^n$ and $f\colon K^n \to K^m$ is differentiable at x with surjective differential df_x. For any $\mathcal{O}_K$-submodule $P \subset K^n$ there exists $r \in \mathbb{Z}$ with the following property. If $a \in K$ with $v(a) > r$ then

$$f(x + aP) = f(x) + a\, df_x(P).$$

Practical approach

Since all of the precisions we've considered can be described as $\mathcal{O}_K$-modules, this theorem gives us an approach to computing with p-adics.

1. For a given function $f\colon K^n \to K^m$ find a method for determining r from $P \subset K^n$.
2. Given an approximate element $(x, P) \in K^n$, compute $df_x(P)$ (possibly "rounding" back to a lattice of the same precision type).
3. Compute $f(x)$, (mostly) ignoring precision.

Working exactly

In the description on the previous slide, the computation of the approximation and of the precision of the result are completely separate. For some problems this approach is reasonable. For example, in the multiplication of polynomials over $\mathbb{Z}_p$ you can indeed just multiply over $\mathbb{Z}$ and then reduce modulo the precision of the result.

Even if the evaluation of f involves division you can work over $\mathbb{Q}$ (for example), and then apply only a single reduction at the end.

Intermediate rounding

On the other hand, if the computation of f involves approximate functions such as exponential and logarithm maps then you need to determine a working precision in computing f. Moreover, even if the computation of f only involves basic arithmetic, it can be valuable to reduce modulo a power of p in order to avoid intermediate coefficient blowup. For this reason it's still valuable to understand the precision behavior of the algorithms, and to use p-adically numerically stable ones.

Determinants

As an application, we determine the precision of the determinant of a matrix. Suppose A is an approximation matrix with precision lattice dA. The differential of the determinant at the identity is the trace function, so as long as dA is small enough, the precision of $\det(A)$ is given by

$$\mathrm{trace}(\det(A)A^{-1}dA).$$

We can therefore compute the determinant of a p-adic matrix by computing the determinant of an appropriate approximation, and then computing the precision separately.

LU Decomposition

We can compute precisions for the LU decomposition similarly. Let $f\colon M_{n\times m}(K) \to M_n(K) \times M_m(K)$ map A to its LU decomposition (L, U). Since

$$A + dA = (L + dL)(U + dU),$$

we have

$$dA = dL \cdot U + L \cdot dU + \text{higher order terms.}$$

Since dL is lower triangular and dU is upper triangular, we can solve for them from dA, L and U.

Further applications

One can apply similar reasoning to determine precisions for evaluation of polynomials, Euclidean division, root finding and factorization, images and kernels, inverse matrices and characteristic polynomials.

Precision Tradeoffs

There are a spectrum of precision types to choose from for polynomials and matrices.

- On one end lies the “flat precision,” specified by a single integer. Computations with flat precision tend to be quite simple, but one may have to sacrifice precision in a long computation.
- Conversely, working directly with O_K-lattices offers the greatest flexibility and preservation of precision, but at the cost of expensive Hermite form computations, with running times on the order of $O(n^{2.3})$.
- In between lie other precision types such as Newton polygons for polynomials, which offer a compromise between speed and flexibility.

Questions?