Local Computations for Global Problems: Intro and 1st Problems

Claus Fieker

MPI Leipzig, August 30, 2021

Intro/ Basics

Local computations are going to refer to computations using p-adic rings/ fields or their extensions.
$\mathbb{Q}_p = \{\sum_{i \geq v} x_i p^i \mid 0 \leq x_i < p\}$ for some prime p.
$\mathbb{Z}_p = \{\sum_{i \geq 0} x_i p^i \mid 0 \leq x_i < p\}$ for some prime p.
Here all sums converge in the p-adic topology. In fact, $\mathbb{Z}_p$ is the completion of $\mathbb{Z}$ and $\mathbb{Q}_p$ of $\mathbb{Q}$. Elements of $\mathbb{Z}_p$ are called p-adic integers, of $\mathbb{Q}_p$ p-adic numbers.

Problem

In general, we cannot represent neither p-adic integers nor p-adic numbers exactly.

Why?

Two key advantages:

- we can *choose* p - there are infinitely many
- we can use *algebra* - approximation and rounding is mainly "mod"

There are also disadvantages:
p-adics do not (yet) play well with real-life (physics, biology, ...)
...thus we need to "move back" to reality at the end.

Rounding and Errors

p-adic numbers are approximated, same as reals: the infinite sum is truncated:

$$x = \sum_{i \geq v} x_i p^i \approx \sum_{i=v}^{n} x_i p^i$$

n is the precision (of x).
Compare to reals

$$x = \sum_{i \leq b} x_i 10^i \approx \sum_{i=n}^{b} x_i 10^i$$

But, since in addition, the carry always propagates upwards, we can never *know* the last digit after some operations. For p-adics, the same carry just vanishes into the unknown at the other end. We don't care.

Rounding and Errors

Given x and y with precision n, we can compute $x + y$, $x \cdot y$ with precision n as well. The error does not increase!
Similar statements hold for more complicated operations such as $\log$, $\exp$.
Careful: division by units is also fine (ie. does not increase the error) while division by p will increase the (absolute) error. (Side remark: multiplication by p thus decreases the error)

Algebra

The more algebraic way to phrase this fundamental result on precision: $\sum_{i=v}^{n} x_i p^i$ is a representative for the coset modulo $p^{n+1}\mathbb{Z}_p$.
Since for rings, the cosets also form a ring, we get the same result.
As long as we are only doing ring operations, we will never loose precision. (However, in some weird situation we can gain precision.)
This simplifies the analysis of complex algorithms significantly.
Precision is only dictated by the size of the result - independent of the length of the computation.

Example

Suppose we want to solve

$$Ax = b$$

for $A \in \mathrm{Gl}(k, \mathbb{Q})$ and $b \in \mathbb{Q}^k$. Since x is unique, we can

- solve directly (Gauss, LU, ...) in $\mathbb{Q}$
- move to $\mathbb{R}$ and use numerical tools
- move to $\mathbb{Q}_p$ or $\mathbb{Z}_p$

Why?

- In $\mathbb{Q}$: coefficient explosion: depending on the strategy intermediate results can be exponentially large
- In $\mathbb{R}$: numerical instability (Hilbert matrix) can easily produce totally useless results
- In $\mathbb{Z}_p$ we're essentially just computing $x \bmod p^n$: all stays small, no errors.

Choose precision

So: what precision n do we need? This is one of the core problems in almost all p-adic computations. To simplify assume A and b are integral, $\|A\|_\infty$, $\|b\|_\infty \leq B$
Here: Cramer's rule writes x as a quotient of 2 determinants, Hadamard gives upper bounds for them. Result:

$$x_i = \frac{n_i}{d_i}$$

for $|n_i|$, $|d_i| \leq (\sqrt{k}B)^k$.
So if $p^n > 2(\sqrt{k}B)^k$ or

$$n \geq k(\log_p B + \frac{1}{2}\log_p k)$$

we're in business. Generically, this is sharp.

Rational Reconstruction

Given $x \bmod p^n$, find a, $b \in \mathbb{Z}$ s.th.

$$a \equiv xb \bmod p^n$$

(then we say $x \equiv a/b \bmod p^n$)
Easiest method:

$$M = \begin{pmatrix} 1 & x \\ 0 & p^n \end{pmatrix}$$

and consider the lattice spanned by the rows of M. If a, $b \leq B$ and $p^n > 2B^2$, then any shortest vector (u, v) in this lattice satisfies $v/u = a/b$.
(u, v) can be found using Gauss-reduction or LLL. (Or using continued fractions without involving lattices or with a variant of just the $\gcd$ algorithm.)
Either way: this can be done *quasi-linear* in n.

$\mathbb{Z}_p$

For $\mathbb{Z}_p$ there are not too many possibilities, usually we can

- work in $\mathbb{Z}/p^n\mathbb{Z}$, but think p-adically (absolute precision)
- write $x = p^v y$ for a unit $y \in (\mathbb{Z}/p^n\mathbb{Z})^*$ (floating model, relative precision)

Note: in the 1st: $p^n = 0$, but not in the second.

- $\mathbb{Z}/p^n\mathbb{Z}$ is well understood: a finite Artinian ring, euclidean, PIR, ...
- the floating model is more like reals/ interval arithmetic
- algorithms can easily be analysed in the 1st, the runtime is predictable
- (some) (pathological) examples work better in the 2nd
- the 2nd can easily be extended to $\mathbb{Q}_p$ (allow negative v)

More Complex Ideas

Consider x in precision n, so $x = \sum_{i=0}^{n-1} x_i p^i + p^n y$ for some unknown y.
Then x^p can be guaranteed with precision $n+1$ as all binomial coefficients for the error term are divisible by p.
However, $x^{p-1} \cdot x$ will usually not have precision $n+1$.
X. Caruso, D. Roe, T. Vaccon gave a framework where lattices are used to trace the error terms (and part of the history) more carefully.
They can assure that $x^{p-1} \cdot x$ will be in precision $n+1$, but the cost scales at least quadratically in the number of variables to be traced.

Lazy Approach

There is also the *lazy approach* where noting is actually computed - until you ask for it explicitly.
In short: instead of computing $x + y$ a straight line program is assembled to record the fact that $z = x + y$ *without actually computing anything*. When the user asks for z up to some precision, the program is analysed and with hopefully minimal effort the desired digits are produced.
C. Doris implemented this in Magma, for (multi-variate) power series this is classic.

Comparison

There is no best, most correct choice.
The absolute one is mathematically easy, not not really p-adic.
The relative one is close to the doubles, but as difficult to use as the doubles (best stick to integral computations)
The lattice framework is great if precision is expensive - and the number of variables small
The lazy approach is great if no a-priory bounds are available

Basics

We're going to need extensions of $\mathbb{Q}_p$ as well. Here is a summary of the basic results we need:
Any finite extension $E/\mathbb{Q}_p$ can uniquely be written as a totally ramified extension over an unramified one: There is U s.th. E/U is totally ramified and $U/\mathbb{Q}_p$ is unramified.
For each degree there is exactly one unramified extension. The Galois group is cyclic, generated by the Frobenius. The residue field is the finite field of the same degree (over $\mathbb{F}_p$).
For us: given that I can choose p (mainly) I will mainly not use ramified extensions. They are interesting and complicated. I want fast and easy. I also try to not have denominators.

Slightly more

Given $E/\mathbb{Q}_p$ a finite extension, there are 2 canonical equivalent valuations/ absolute values on E:
One extending the valuation/ absolute value from $\mathbb{Q}_p$
One coming from the maximal ideal.
Both are used and useful. Just be careful to understand which is implemented.
Finally: extensions occur as completions of number fields at prime ideals.

...even more

Computing the canonical split into ramified/ unramified is *hard*.
Not computing the split makes computation of valuations *hard*.
The precise notion of precision of elements is also tricky in general:
for unramified: choose an exact polynomial and precision is the minimal precision of the coefficients.
For ramified: precision is usually measured in terms of a uniformizer, so does not immediately translate into coefficients. Be *careful*
The discussion about the different models is the same.
Thus: avoiding this, when possible is good.

Completions

Let K be a number field, $\mathcal{O}$ some order and $\mathfrak{p}$ be some inevitable unramified prime ideal. Then the completion of $\mathcal{O}$ at $\mathfrak{p}$ is the valuation ring of the unramified extension of $\mathbb{Q}_p$ of the degree the inertia index of $\mathfrak{p}$.
It can easily be constructed by using the same polynomial as for the residue field.
Note: unramified implies that we can choose *exact* defining polynomials that can be used at any precision.
By using primes not dividing the discriminant (of the polynomial defining the field), we do not need any p-maximal order.
In general, computing a general completion is "the same" as computing a p-maximal order, the primes in there and then some linear algebra.

Splitting Fields

$K = \mathbb{Q}[t]/f$ and p be an easy prime: coprime to the discriminant of f, avoiding a finite number of additional pitfalls.
Then, $f = \prod f_i \bmod p$ and, setting $d = \text{lcm}_i \deg(f_i)$ we see that f splits completely in $\mathbb{F}_{p^d}$. In fact, this is *the* splitting field.
Let E be the unramified extension of $\mathbb{Q}_p$ of degree d. Using Newton lifting we see that f splits into linears over E as well, so E is a p-adic splitting field.
(Lifting is easy since disc $f \neq 0 \bmod p$ implies all roots are single roots.)

Lifting

Given

$$f \equiv \prod_{i=1}^{d}(t - \alpha_i) \bmod p$$

we want

$$f \equiv \prod_{i=1}^{d}(t - \beta_i) \bmod p^n$$

for some possibly large n.

...

Using classical Newton iteration

$$x \mapsto x - \frac{f(x)}{f'(x)}$$

which will converge quadratically (so need $\log_2 n$ steps). Each iteration would need (using Horner) $O(d)$ many multiplication. We have d roots, so a total of $\tilde{O}(d^2)$
We can also use a product tree to lift the entire factorisation! This will result a $\tilde{O}(d)$ total operations (using fast arithmetic). For non-trivial examples this wins.

Lifting

Of course, for small examples, using the double iteration is not too bad. It is also possible to first lift a partial factorisation over $\mathbb{Q}_p$ before lifting the factorisations of the factors in the extension.
J. Klüners even chose a tower of extensions to step-by-step refine the roots.
In the evil case (ramified) one needs to also understand that a factorisation up to precision k will not imply that all factors are known to precision k. In general, an input precision of k means the individual factors *cannot* be guaranteed with the same precision.
For the easy, unramified case, this is no problem.

Task

Given $\alpha_i \in K$, for some number field K, find the lattice of multiplicative relations, i.e. all $\lambda_i \in \mathbb{Z}$ s.th. $\prod \alpha_i^{\lambda_i} = 1$.
By looking at valuations, this reduces via $\mathbb{Z}$-linear algebra to a problem of units:
Given α_i units in some number field K, find the lattice of relations.
Using logarithms: $\prod \alpha_i^{\lambda_i} = 1$ is almost the same as $\sum \lambda_i \log \alpha_i = 0$ (Almost: there is a problem with torsion, but that can be fixed in a third step)

Classical...

Using classical logarithms: K embeds into $\mathbb{C}$ exactly n different ways. Sorting the usual way (taking only one of conjugate pairs), we get a map

$$L : K \to \mathbb{R}^{r_1+r_2} : \alpha \mapsto (\log |\alpha^{(i)}|)$$

Under our assumptions, the image of the units under this map forms a lattice and, essentially, linear algebra will find the answer. However: the result should be in $\mathbb{Z}$, but this way it will be in $\mathbb{R}$. Instead of exact, it will be full of rounding errors.

...

Partial solution (1): normalise the last component to be 1, then the others should be in $\mathbb{Q}$. Use continued fractions to recover plausible results - and verify.
Partial solution (2): this is a lattice problem. Use LLL. Problem: this lattice is in $\mathbb{R}^?$, there is no implementation available for lattices that are not in $\mathbb{Q}^?$. So will have to start with an approximated lattice.

... using p-adics

Or, we could use p-adics. Let p be an easy prime (unramified, not an index divisor, ...) and E an extension of $\mathbb{Q}_p$ that is a splitting field. Maybe choose p s.th. the degree of E is not too large.
Then K embeds into E also n times and we can use the "same" map as above:

$$L_p : K \to E^n : \alpha \mapsto (\log \alpha^{(i)})_i$$

where $\log : E \to E$ is the p-adic logarithm.
Caveat: the p-adic log is only defined (via power series) on 1-units, ie. elements such that $v(1-x) > 0$. But there is works. We can extend this as α is a unit, hence $v(\alpha^{(i)}) = 0$, so a (large) power of α will be a 1-unit, so $\log \alpha^{(i)} := 1/\mu \log(\alpha^{(i)})^\mu$.

Leopoldt

Believing Leopoldts conjecture, the $L_p(\alpha)$ are p-adically dependent iff the $L(\alpha)$ are $\mathbb{R}$-dependent. Normalising, as above, the last coefficient in the dependency to 1, we now get

$$\prod \alpha_i^{\lambda_i} = \zeta$$

for a torsion unit ζ and $\lambda_n = 1$, $\lambda_i \in \mathbb{Q}$ is given $\mathrm{mod} p^n$, so rational reconstruction can find λ_i exactly. If we have bounds....

Bounds

Tricky: let α_i for $1 \leq i \leq n-1$ be independent and α_n an additional unit.

We have 2 tasks:

- decide if α_n is independent as well (false if n is too large)
- if not, find the dependency.

p-adically, if they are independent, they will be independent in almost any precision.

We need more:

Bound, Dobrowski

If α is a non-torsion unit, then there is at least one conjugate

$$\geq 1 + \frac{1}{6}\frac{\log n}{n^2}$$

If the α_i are dependent, they generate a $\mathbb{R}$-lattice of dimension $n-1$. Let β_i be a basis for this lattice. Then $\alpha_i = \prod \beta_j^{b_{i,j}}$ for $b_{i,j} \in \mathbb{Z}$.
If the α_i are dependent, then $\alpha_n^{a_n} = \prod_{i=1}^{n-1} \alpha_i^{a_i}$. Using the β_i

$$\prod \beta_j^{a_n b_{n,j}} = \prod_{i=1}^{n-1} \prod \beta_j^{a_i b_{i,j}} = \prod_j \beta_j^{\sum a_i b_{i,j}}$$

We get a system of linear equations:

$$a_n b_{n,j} = \sum a_i b_{i,j}$$

Which we can normalise (the 1st $n-1$ are independent!):

$$b_{n,j} = \sum \frac{a_i}{a_n} b_{i,j}$$

Cramer's rule gives a_i/a_n as a quotient of determinants of the $b_{i,j}$, so we need to bound the determinants.

As we do not know the $b_{i,j}$ (they depend on the β which we do not have), this is fun.
The $\det(b_{i,j})$ is however the *index* of

$$\langle \beta_i | i \rangle : \langle \alpha_i | 1 \leq i \leq n-1 \rangle$$

. The index can be computed using (real) logarithms, hence estimated:

$$|\det(b_{i,j})_{1 \leq i,j \leq n-1}| = \frac{\sqrt{\det L(\alpha_i) L^t(\alpha_j)}}{\sqrt{\det L(\beta_i) L^t(\beta_j)}}$$

The numerator determinant can be estimated using Hadamard (we have the α_i explicitly. Here precision is not important, any upper bound will do)

The denominator: we need a lower bound. The elements form a lattice with a lower bound for the Minkowski-minimum, so we can use this.
Note: if $n-1$ is the unit rank, we can replace Minkowski by any lower regulator bound.
Note: this estimate is pessimistic. Using a lower precision is fine - we need to verify the relation anyway.

Being more "complex"

So far we've used embeddings into a single splitting field and used all "conjugates". However, comparing to the classical case: for each *pair* of complex embeddings, we take only *one* datum and, in general, we split into real and imaginary part to get n real values. So Let $\mathfrak{p}$ be an easy prime ideal above p, $K_{\mathfrak{p}}$ be the completion of $\mathfrak{p}$. Then

- $K_{\mathfrak{p}} : \mathbb{Q}_p = f(\mathfrak{p}|p)$
- $\mathrm{Gal}(K_{\mathfrak{p}})$ is cyclic, generated by σ
- if $\alpha_i \in K_{\mathfrak{p}}$, then so is $\alpha_j = \sigma\alpha_i$
- $\log$ and σ commute
- thus any relation between the i-th conjugate implies one for the j-th
- we do not want $K_{\mathfrak{p}}$-relations, but $\mathbb{Q}_p$-relations

Better:

So:

- for each prime ideal $\mathfrak{p}_i$ over p, we pick *one* conjugate in $K_{\mathfrak{p}_i}$
- take the logarithms L_p
- any $\mathbb{Q}_p$-relation implies a coefficient wise relation

So we replace the linear system from above by an $n \times r$ system over $\mathbb{Q}_p$ - a clear win.

Why????

So far, given that we need to verify any p-adic relation, and/or need to have bounds, both coming from real computations, why do the p-adic one at all?

- fine print: the units are usually given factored: $\alpha = \prod \beta_i^{n_i}$ where n_i can easily have 1000 bits and the product several hundred terms. Thus to get precise real information requires a precision logarithmic in the exponents and linear in the number of terms.
  Upper bounds are easier.

...

- during the computation, the representations grow, badly.
  Thus even the p-adic precision needs to be increased.
  However: instead of increasing the precision, we can just use a different prime and do the same. Given that the solution is unique in $\mathbb{Q}$, we can combine information using CRT, in parallel.
  (I don't know how to increase the $\log$-precision - and they tend to dominate the time.)
- no hassle with numerical problems in the linear equations.
  There are hardly any implementations of sophisticated linear algebra in high real precision, and neither any literature.
  Floats are useless here.

Verification

We need to compute image under the real-$\log$-map L to an absolute precision of 10^{-3} or so (from the bound) and see if we get close enough to 0.
The difference to the direct solver is that the verification needs only *low* absolute precision, while the "finding" would need absolute precision proportional to the size of the relation.

Postprocessing

At the end of each run we have either

- a relation
- a proof that the new unit is independent.

Having a relation allows to compute a basis for the new group: Let $U = \langle \alpha_1, \ldots, \alpha_{n-1} \rangle$ with independent α_i and α_n dependent with relation $\alpha_n^{a_n} \prod_{i=1}^{n-1} \alpha_i^{a_i} = 1$ for $a_i \in \mathbb{Z}$. Assume $\gcd(a_i | i) = 1$. Then, using HNF with transformation, we find $T \in \mathrm{Gl}(n, \mathbb{Z})$ s.th.

$$(a_1, \ldots, a_n)T = (1, 0, \ldots, 0)$$

Then

$$\beta_j = \prod_{i=1}^{n} (T^{-1})_{i,j} \alpha_i$$

for $2 \leq j \leq n$ is the new basis.