

Computations with p -adic polylogarithms in Sage

– Global Virtual SageDays 109

Alex J. Best

27/5/2020

Boston University

These slides are available online (in handout form) at
[https://alexjbest.github.io/talks/
sage-computations-polylogs/slides_h.pdf](https://alexjbest.github.io/talks/sage-computations-polylogs/slides_h.pdf)



These slides are available online (in handout form) at

https://alexjbest.github.io/talks/sage-computations-polylogs/slides_h.pdf

Goal: Introduce you to (p -adic polylogarithms) in Sage and explain some applications of these computations to solving S -unit equations.



What are polylogarithms?

Polylogarithms are special functions of a complex variable z , obtained by iteratively dividing by z and taking antiderivatives, starting with $\text{Li}_0 = \frac{z}{1-z}$:

$$\text{Li}_1(z) = \int_0^z \frac{t}{(1-t)t} dt = -\log(1-z),$$

$$\text{Li}_2(z) = \int_0^z \frac{-\log(1-t)}{t} dt \quad \text{the } \textit{dilogarithm},$$

$$\text{Li}_3(z) = \int_0^z \frac{\text{Li}_2(t)}{t} dt$$

$$\vdots$$

What are polylogarithms?

Their power series expansions around zero are rather nice:

$$\operatorname{Li}_n(z) = \sum_{k=1}^{\infty} \frac{z^k}{k^n} = z + \frac{z^2}{2^n} + \frac{z^3}{3^n} + \cdots$$

(for $|z| < 1$) and we can see that

$$\operatorname{Li}_n(1) = \zeta(n)$$

What are polylogarithms?

Their power series expansions around zero are rather nice:

$$\operatorname{Li}_n(z) = \sum_{k=1}^{\infty} \frac{z^k}{k^n} = z + \frac{z^2}{2^n} + \frac{z^3}{3^n} + \cdots$$

(for $|z| < 1$) and we can see that

$$\operatorname{Li}_n(1) = \zeta(n)$$

indeed Sage knows this symbolically

```
sage: polylog(3, 1)  
zeta(3)
```

```
sage: polylog(2, 1)  
1/6*pi^2
```

```
sage: polylog(2, 1/2)  
1/12*pi^2 - 1/2*log(2)^2  
sage: polylog(2, 7.0)  
1.24827318209942 -  
6.11325702881799*I
```

Properties of polylogarithms

These functions satisfy many interesting functional equations:

$$\operatorname{Li}_2(x) + \operatorname{Li}_2(1 - x) = \operatorname{Li}_2(1) - \log(x) \log(1 - x)$$



Properties of polylogarithms

These functions satisfy many interesting functional equations:

$$\operatorname{Li}_2(x) + \operatorname{Li}_2(1 - x) = \operatorname{Li}_2(1) - \log(x) \log(1 - x)$$

Powering:

$$\operatorname{Li}_n(z^k) = \frac{\sum_{m=0}^{k-1} \operatorname{Li}_n(\zeta_k^m z)}{k^{n-1}}$$



Properties of polylogarithms

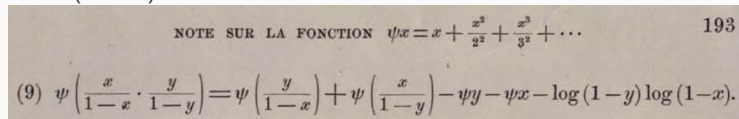
These functions satisfy many interesting functional equations:

$$\operatorname{Li}_2(x) + \operatorname{Li}_2(1-x) = \operatorname{Li}_2(1) - \log(x) \log(1-x)$$

Powering:

$$\operatorname{Li}_n(z^k) = \frac{\sum_{m=0}^{k-1} \operatorname{Li}_n(\zeta_k^m z)}{k^{n-1}}$$

Abel (1826):



NOTE SUR LA FONCTION $\psi x = x + \frac{x^2}{2^2} + \frac{x^3}{3^2} + \dots$ 193

(9) $\psi\left(\frac{x}{1-x} \cdot \frac{y}{1-y}\right) = \psi\left(\frac{y}{1-x}\right) + \psi\left(\frac{x}{1-y}\right) - \psi y - \psi x - \log(1-y) \log(1-x).$

What are the p -adics?



Parallel with the real/complex numbers. They are defined by:

1. Fixing a norm on \mathbf{Q} , defined by

$$|x|_p = p^{-\overbrace{\max\{i \in \mathbf{Z} : p^i | x\}}^{=\nu_p(x)}}$$

2. Completing \mathbf{Q} with respect to $|\cdot|_p$, to get a complete normed field.

What are the p -adics?



Parallel with the real/complex numbers. They are defined by:

1. Fixing a norm on \mathbf{Q} , defined by

$$|x|_p = p^{-\overbrace{\max\{i \in \mathbf{Z} : p^i | x\}}^{= \nu_p(x)}}$$

2. Completing \mathbf{Q} with respect to $|\cdot|_p$, to get a complete normed field.

Upshot: p is now *small* ($|p|_p = p^{-1}$), so instead of decimal expansions for elements of \mathbf{R} :

$$\frac{1}{3} = 3 \cdot \frac{1}{10} + 3 \cdot \frac{1}{10^2} + 3 \cdot \frac{1}{10^3} + 3 \cdot \frac{1}{10^4} + \text{smaller terms}$$

we have p -adic expansions for elements of \mathbf{Q}_p :

$$\frac{1}{3} = 5 + 4 \cdot 7 + 4 \cdot 7^2 + 4 \cdot 7^3 + 4 \cdot 7^4 + \text{smaller terms}$$

p -adics in Sage

There is now good support for p -adics in Sage, thanks to many people, but in particular Xavier Caruso, David Roe and Julian Rüth are regularly working on this (on Zulip).

p -adics in Sage

There is now good support for p -adics in Sage, thanks to many people, but in particular Xavier Caruso, David Roe and Julian R  th are regularly working on this (on Zulip).

Support includes:

- Basic arithmetic
- Many different precision tracking modes (absolute / relative, fixed / capped precision)
- Hensel lifting (Newton's method)
- \exp and \log
- Frobenius, and Teichm  ller representatives
- Extensions
- Li_n ?
- Much more!

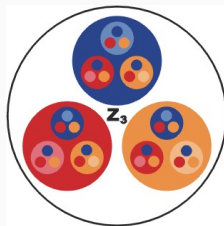
p -adic integration

To define Li_n , p -adically, we must define antiderivatives of p -adic functions.

Easy! Just write out power series locally and take the antiderivative termwise!

Problem: we can work out local antiderivatives here, and calculate integrals between nearby points, but we can't analytically continue. Distinct disks don't overlap in the p -adic topology.

A different constant of integration to be chosen on each p -adic disk.



Bad topology!

Assume more of the integral, to pin down the function defined:
assume Frobenius equivariance:

$$\int_{x^p}^{y^p} f(t) \, dt = \int_x^y f(t^p) \, d(t^p)$$



p -adic polylogarithms

Assume more of the integral, to pin down the function defined:
assume Frobenius equivariance:

$$\int_{x^p}^{y^p} f(t) dt = \int_x^y f(t^p) d(t^p)$$

For example: If $f(t) = 1/t$ we can define

$$\log(z) := \int_1^z \frac{dt}{t}$$

and find values for z (p -adically) near 1 by integrating a power series,

$$\log(z^p) = \int_1^{z^p} \frac{dt}{t} = \int_1^z \frac{pt^{p-1} dt}{t^p} = p \int_1^z \frac{dt}{t} = p \log(z)$$

so for a p^k – 1st root of unity ζ we have

$$\log(\zeta) = \log(\zeta^{p^k}) = p^k \log(\zeta) \implies \log(\zeta) = 0.$$

p -adic polylogarithms in Sage

Some initial cases (but with restrictions on p, n, z) implemented by Jennifer Balakrishnan (at a Sage days).

Sage Days 87: p -adics in Sage and the LMFDB (2017), I wrote a complete implementation and #20260 was merged.

p -adic polylogarithms in Sage

Some initial cases (but with restrictions on p, n, z) implemented by Jennifer Balakrishnan (at a Sage days).

Sage Days 87: p -adics in Sage and the LMFDB (2017), I wrote a complete implementation and #20260 was merged.

```
sage: K = Qp(5, prec=7);
```

```
sage: K(1 + 5).polylog(2)  
5 + 5^2 + 5^3 + O(5^4)
```

```
sage: K(1 + 5^2).polylog(2)  
5^2 + 5^4 + O(5^5)
```

```
sage: K(1 + 5^3).polylog(2)  
5^3 + O(5^5)
```

```
sage: K(1 + 5^4).polylog(2)  
5^4 + O(5^6)
```

```
sage: K(1 + 5^5).polylog(2)  
5^5 + O(5^6)
```

```
sage: K(1/2).polylog(2)  
3*5^2 + 3*5^3 + O(5^4)
```

```
sage: -K(1/2).log()^2/2  
3*5^2 + 3*5^3 + 2*5^4 + 5^5 +  
2*5^7 + O(5^8)
```

```
sage: K(7).polylog(3)  
3*5^3 + O(5^4)
```

How does it work?

Besser – de Jeu: “ Li^p -Service? An Algorithm for Computing p -Adic Polylogarithms.” Math. Comp. 77, no. 262 (2008).

- Near 0: use the power series $\text{Li}_n(z) = \sum_{k=1}^{\infty} \frac{z^k}{k^n}$
- Near ∞ : use the relation

$$\text{Li}_n(z) + (-1)^n \text{Li}_n(z^{-1}) = -\frac{1}{n!} \log^n(z)$$

to reduce to the first case.

How does it work?

Besser – de Jeu: “ Li^p -Service? An Algorithm for Computing p -Adic Polylogarithms.” Math. Comp. 77, no. 262 (2008).

- Near 0: use the power series $\text{Li}_n(z) = \sum_{k=1}^{\infty} \frac{z^k}{k^n}$
- Near ∞ : use the relation

$$\text{Li}_n(z) + (-1)^n \text{Li}_n(z^{-1}) = -\frac{1}{n!} \log^n(z)$$

to reduce to the first case.

- Else: Must be near a $(p^k - 1)$ st root of unity for some k (except near 1). Letting

$$\text{Li}_n^{(p)}(z) = \text{Li}_n(z) - \frac{1}{p^n} \text{Li}_n(z^p)$$

Reduces to computing $\text{Li}_m^{(p)}(\zeta^{p^j})$ for $m \leq n$ and $j < k$.

- Near 1: see the paper!

Application: The S -unit equation

One classic diophantine equation is the S -unit equation: for a fixed finite set of primes S

$$u + v = 1, u, v \in \mathbf{Q}^\times$$

where we ask that the only primes present in the factorization of u, v are those in S .

So

$$\frac{4}{3} - \frac{1}{3} = 1$$

is a solution of the $\{2, 3\}$ -unit equation, but not of the $\{2\}$ -unit equation or $\{3\}$ -unit equation alone.

The most difficult cases of this equation are when S is large, or over number fields instead.

In a joint project with Theresa Kumpitsch, Martin Lüdtkke, Angus McAndrew, Lie Qian, Elie Studnia, and Yujie Xu, we have been using p -adic polylogarithms (in Sage) to provably determine the full set of solutions to these equations.

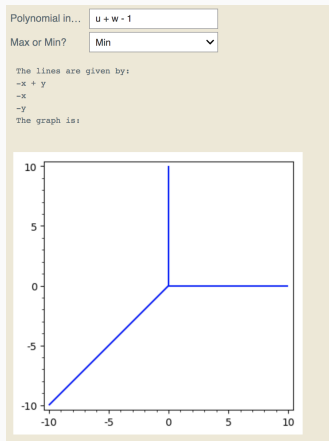
For now only for small S , over \mathbf{Q} .



The S -unit equation

Note that for any prime p , either $p|u$, $p|v$, or both $p|u^{-1}$ and $p|v^{-1}$.

Plotting $\nu_p(u)$ against $\nu_p(v)$ we get a diagonal Y shape:



Interact by Wang Weikun

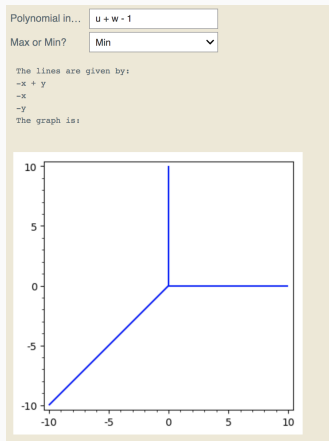
The S -unit equation

Note that for any prime p , either $p|u$, $p|v$, or both $p|u^{-1}$ and $p|v^{-1}$.

Plotting $\nu_p(u)$ against $\nu_p(v)$ we get a diagonal Y shape:

This is an instance of a more general phenomenon:
the valuations lie on
the tropical curve associated with
the defining equation $u + v = 1$.

Note: If $p \notin S$ then
 $u \pmod{p}$ cannot be any of $0, 1, \infty$.



Interact by Wang Weikun

Applications

Minhyong Kim has developed a programme of *non-abelian Chabauty*.

Extends classical Chabauty's method for finding rational and integral points on curves as zeroes of abelian integrals on the curve.

Applications

Minhyong Kim has developed a programme of *non-abelian Chabauty*.

Extends classical Chabauty's method for finding rational and integral points on curves as zeroes of abelian integrals on the curve.

One specific consequence of this theory due to Dan-Cohen–Wewers: there exists a commutative diagram for any fixed prime p not in S .

$$\begin{array}{ccc} \mathbf{P}^1 \setminus \{0, 1, \infty\}(\mathbf{Z}[\frac{1}{S}]) & \longrightarrow & \mathbf{P}^1 \setminus \{0, 1, \infty\}(\mathbf{Z}_p) \\ \downarrow (\nu_\ell(z), \nu_\ell(1-z))_{\ell \in S} & & \downarrow (\log(z), \log(1-z), -\mathrm{Li}_2(z)) \\ \mathbf{A}_{\mathbf{Q}_p}^{2|S|} & \xrightarrow{\quad\quad\quad} & \mathbf{A}_{\mathbf{Q}_p}^3 \\ (\sum_{\ell \in S} x_\ell \log(\ell), \sum_{\ell \in S} y_\ell \log(\ell), h(\underline{x}, \underline{y})) & & \end{array}$$

Applications

In this diagram everything is defined, except h , it is a bilinear form in the x_ℓ and y_ℓ .

Strategy:

- Given enough points in the top $\mathbf{P}^1 \setminus \{0, 1, \infty\}(\mathbf{Z}[\frac{1}{S}])$, we can find their image in the $\mathbf{A}_{\mathbf{Q}_p}^3$ going round the diagram both ways, commutativity then determines h .
- Given a subvariety V of $\mathbf{A}_{\mathbf{Q}_p}^3$ we can find all S -units that land in V by pulling back along the right vertical arrow.

Applications

In this diagram everything is defined, except h , it is a bilinear form in the x_ℓ and y_ℓ .

Strategy:

- Given enough points in the top $\mathbf{P}^1 \setminus \{0, 1, \infty\}(\mathbf{Z}[\frac{1}{S}])$, we can find their image in the $\mathbf{A}_{\mathbf{Q}_p}^3$ going round the diagram both ways, commutativity then determines h .
- Given a subvariety V of $\mathbf{A}_{\mathbf{Q}_p}^3$ we can find all S -units that land in V by pulling back along the right vertical arrow.
- Have to solve a polynomial (with \mathbf{Q}_p coefficients) combinations of $\log(z)$, $\log(1 - z)$, $\text{Li}_2(z)$.
- To find a useful collection of V 's covering all possible S -units when $|S| = 2$ we use the tropical picture, we have 3 components $\{-, |, /\}$ for each prime $\ell \in S$ (Betts–Dogra).

Example

When $S = \{2, 3\}$ we have many solutions

$$\begin{aligned} & \left\{2, \frac{1}{2}, -1\right\} \cup \left\{3, \frac{1}{3}, \frac{2}{3}, \frac{3}{2}, -\frac{1}{2}, -2\right\} \\ & \cup \left\{4, \frac{1}{4}, \frac{4}{3}, \frac{3}{4}, -\frac{1}{3}, -3\right\} \cup \left\{-\frac{1}{8}, \frac{1}{9}, \frac{9}{8}, \frac{8}{9}, 9, -8\right\} \end{aligned}$$

from which we can determine that:

$$h = \frac{1}{2} \log(2)^2 x_2 y_2 - \operatorname{Li}_2(-2) x_2 y_3 - \operatorname{Li}_2(3) x_3 y_2 + \frac{1}{2} \log(3)^2 x_3 y_3$$

Example

For one choice of V we get that for any S -unit z with $2|z$ and $3|(1-z)$ we have $\text{Li}_2(-2) \text{Li}_2(z) = \text{Li}_2(3) \text{Li}_2(1-z)$ which we can solve

```
sage: allr = allroots(K(1-3).log(p_branch)*K(3).log(p_branch)*Li2z -
K(3).polylog(2)*logz*logone_z,p)
sage: for r in allr:
....:     print("root: ",r)
....:     print(algdep(r, 2))

root:  2*5^-1 + 1 + 5^2 + 5^5 + 5^6 + 5^8 + 5^9 + 3*5^11 + 3*5^12 + 4*5^13 +
      4*5^14 + 2*5^15 + 4*5^16 + 3*5^17 + 4*5^18 + 2*5^19 + 0(5^20)
11775*x^2 - 119800*x - 28359
root:  2 + 0(5^24)
x - 2
root:  2 + 4*5 + 4*5^2 + 4*5^3 + 4*5^4 + 4*5^5 + 4*5^6 + 4*5^7 + 4*5^8 +
      4*5^9 + 4*5^10 + 4*5^11 + 4*5^12 + 4*5^13 + 4*5^14 + 4*5^15 + 4*5^16 +
      4*5^17 + 4*5^18 + 4*5^19 + 4*5^20 + 4*5^21 + 4*5^22 + 4*5^23 + 0(5^24)
x + 3
root:  3 + 4*5^23 + 0(5^24)
x - 3
root:  3 + 5^2 + 2*5^3 + 5^4 + 3*5^5 + 5^6 + 5^7 + 5^9 + 2*5^10 + 3*5^11 +
      2*5^12 + 3*5^13 + 3*5^14 + 4*5^15 + 5^16 + 4*5^17 + 3*5^18 + 2*5^22 +
      5^23 + 0(5^24)
128901*x^2 - 49672*x - 62943
root:  4 + 5 + 0(5^24)
x - 9
```