Frobenius lifts and point counting for smooth curves

Amnon Besser, François Escriva

(Joint work with Rob de Jeu) September, 25th 2013

The algorithm

INPUT: A smooth, complete curve C/R (+ some assumptions) together with some auxiliary data.

- Compute the matrix M_1 with entries $\omega_i \cup \omega_j$.
- For each missing point:
  - Estimate the required precision in t.
  - Solve some local equations to get a local lift of the Frobenius.
  - Compute all the contributions $\mathrm{Res}_{\mathcal{E}} \phi\omega_i \int \omega_j$ to the matrix M_2.

The algorithm

- From this recover the matrix M of the action of the Frobenius as $M = M_1^{-1} M_2$.

OUTPUT: If the starting precision is high enough, returns the numerator of the zeta function of C_k.

Proposition

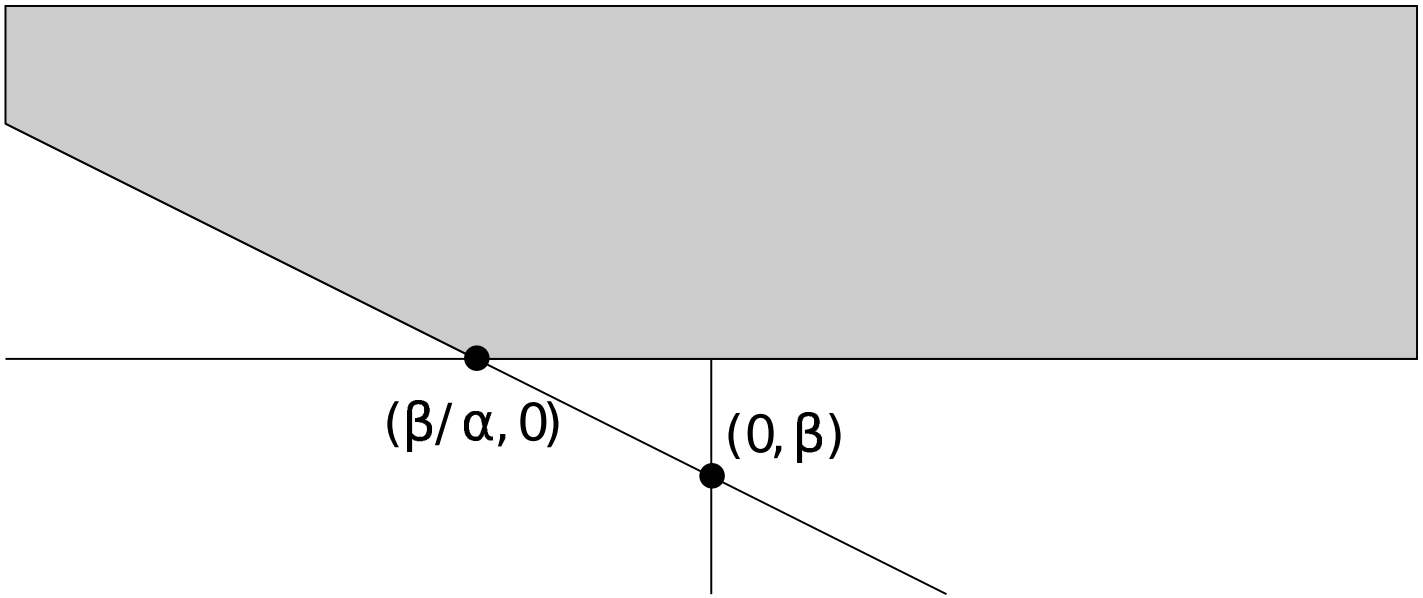
The asymptotic complexity of this algorithm is $\tilde{O}(pl^3)$, where the $\tilde{O}$ term depends polynomially on the genus, and $\#k = p^l$.

Finite precision calculations

- Fix a missing point, given η and ω of the second kind with local expansions $\eta = \sum_m a_m t^m \mathrm{d}t$ and $\omega = \sum_m b_m t^m \mathrm{d}t$, we need to compute $\mathrm{Res}_{\mathcal{E}} \omega \int \eta = \sum_{m \neq -1} \frac{a_m b_{-m-2}}{m+1}$.
- Can these residue computations be done with a finite precision in p and in t?

- For α, β rational numbers with $\alpha > 0$, we let:

$$R_{\alpha,\beta}((t)) = \left\{ \sum_{m \in \mathbb{Z}} a_m t^m \, , a_m \text{ in } R \text{ and } v(a_m) \geq -\alpha m + \beta \right\} .$$



Finite precision calculations

Proposition

At a given missing point, the expansions of all the ω_i and $\phi(\omega_j)$ are in some $R_{\alpha,*}((t)) \cdot \mathrm{d}t$.

- For N in $\mathbb{Q}_{>0}$, we define $I_N = \{x \in K \text{ with } v(x) \geq N\}$.
- Similarly

$$S^N_{\alpha,\beta}((t)) = \left\{ \sum_{m \in \mathbb{Z}} \overline{a_m} t^m \text{ with } \sum_{m \in \mathbb{Z}} a_m t^m \text{ in } R_{\alpha,\beta}((t)) \right\},$$

where we take the coefficients in $S^N = R/I_N$.

Finite precision calculations

- There is a well-defined multiplication

$$S^N_{\alpha,\beta_1}((t))/t^L \times S^N_{\alpha,\beta_2}((t))/t^L \to S^N_{\alpha,\beta_1+\beta_2}((t))/t^L\,.$$

Proposition

Given N in $\mathbb{Q}_{>0}$, the map $(\omega,\eta) \mapsto \mathrm{Res}_{\mathcal{E}}\omega \int \eta + I_N$ factors through $S^{N_1}_{\alpha,\beta_1}/t^{L_1}\cdot \mathrm{d}t \times S^{N_2}_{\alpha,\beta_2}/t^{L_2}\cdot \mathrm{d}t$ for suitable N_1, N_2 in $\mathbb{Q}_{>0}$ and positive integers L_1, L_2.

- The residue calculations can be done using the $S^{N_i}_{\alpha_i,\beta_i}((t))/t^{L_i}\cdot \mathrm{d}t$.

The implementation

Implementation done using SAGE.
A few remarks:

- All the operations in the $S^N_{\alpha,\beta}((t))/t^L$ have been implemented using NTL.
- To estimate the N_i's and L_i's, one needs to estimate α beforehand.

Timings

With an Intel Core i5@3.10GHz

g	1	1	2	2	9	9
#k	7	101	23	101	23	101
Time (s)	0.31	1.30	7.58	19.51	221.59	680.18

Rather slow, however

- The implementation does not use any "trick" and deals with general curves.
- Localizing seems to noticeably improve some of these timings.

Possible improvements

- Automating (part of) the computation of the auxiliary data.
- Avoid recomputing the local equations if the precision of the input data is high enough?
- Optimizing for special cases?
- Bottleneck when applying the local lift to the 1-forms.
- Parallelization.