Coleman Integration in Larger Characteristic

ANTS XIII — University of Wisconsin, Madison

Alex J. Best

17/7/2018

Boston University

The big picture

Kedlaya

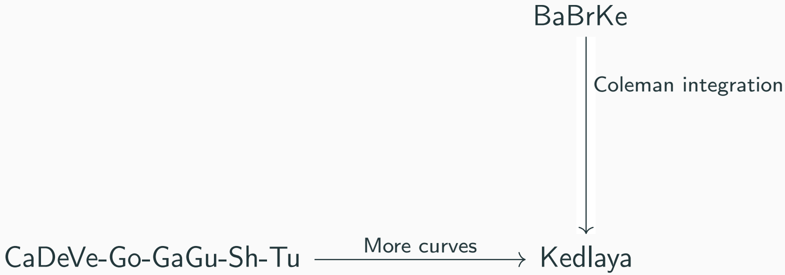
Arul, Balakrishnan, Best, Bradshaw, Castryk, Costa, Denef, Gaudry, Gurel, Harvey, Kedlaya, Magner, Minzlaff, Shieh, Triantafillou, Tuitman, Vercauteren, and more. . .

The big picture

$$\text{CaDeVe-Go-GaGu-Sh-Tu} \xrightarrow{\text{More curves}} \text{Kedlaya}$$

Arul, Balakrishnan, Best, Bradshaw, Castryk, Costa, Denef, Gaudry, Gurel, Harvey, Kedlaya, Magner, Minzlaff, Shieh, Triantafillou, Tuitman, Vercauteren, and more...

The big picture



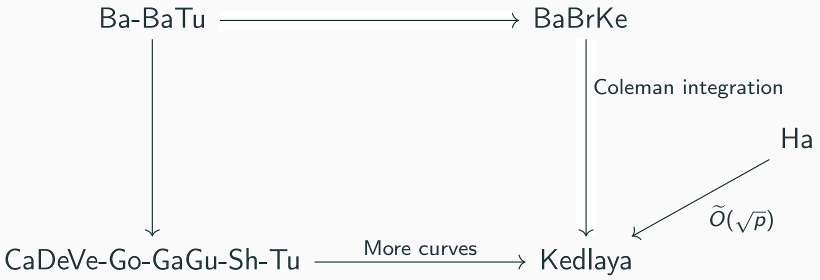
Arul, Balakrishnan, Best, Bradshaw, Castryk, Costa, Denef, Gaudry, Gurel, Harvey, Kedlaya, Magner, Minzlaff, Shieh, Triantafillou, Tuitman, Vercauteren, and more. . .

The big picture

Arul, Balakrishnan, Best, Bradshaw, Castryk, Costa, Denef, Gaudry, Gurel, Harvey, Kedlaya, Magner,

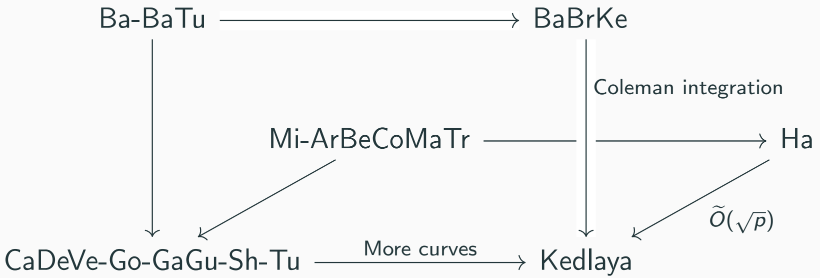
Minzlaff, Shieh, Triantafillou, Tuitman, Vercauteren, and more. . .

The big picture



Arul, Balakrishnan, Best, Bradshaw, Castryk, Costa, Denef, Gaudry, Gurel, Harvey, Kedlaya, Magner, Minzlaff, Shieh, Triantafillou, Tuitman, Vercauteren, and more...

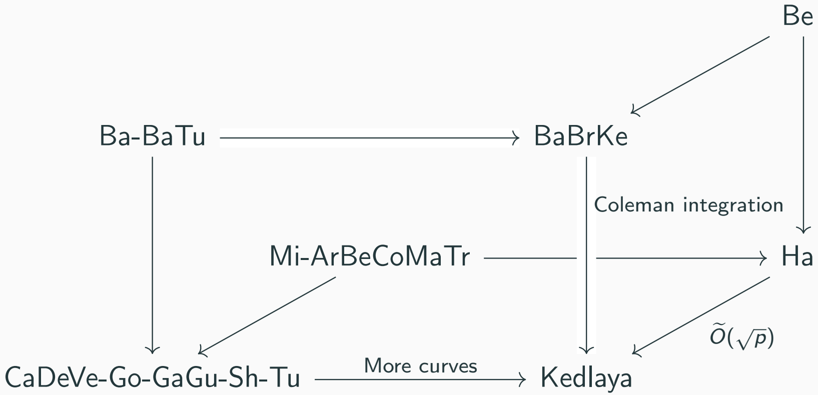
The big picture



Arul, Balakrishnan, Best, Bradshaw, Castryk, Costa, Denef, Gaudry, Gurel, Harvey, Kedlaya, Magner,

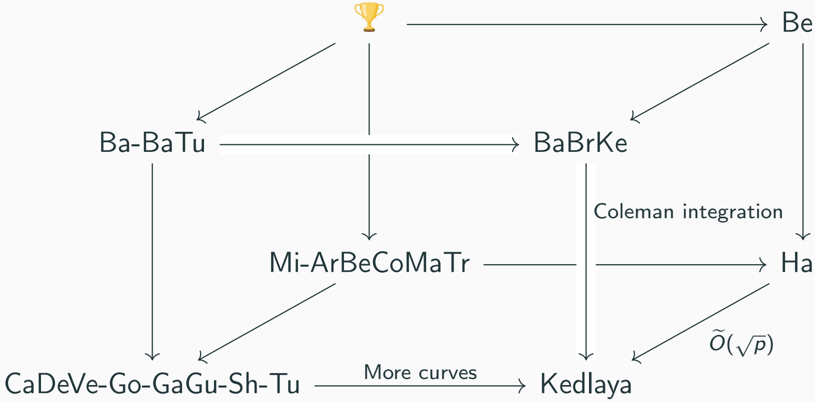
Minzlaff, Shieh, Triantafillou, Tuitman, Vercauteren, and more. . .

The big picture



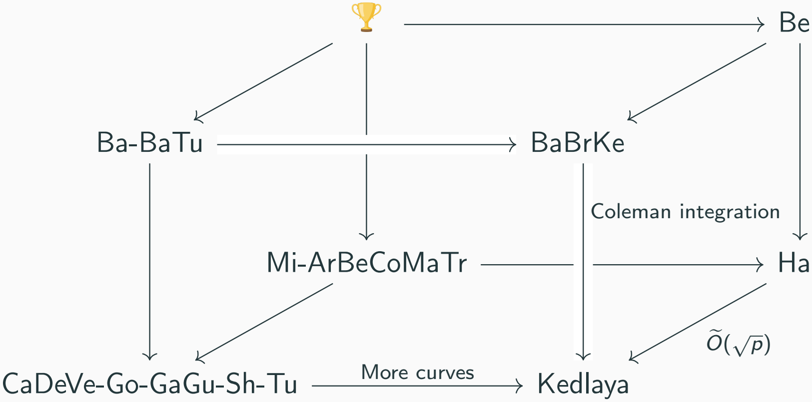
Arul, Balakrishnan, Best, Bradshaw, Castryk, Costa, Denef, Gaudry, Gurel, Harvey, Kedlaya, Magner, Minzlaff, Shieh, Triantafillou, Tuitman, Vercauteren, and more...

The big picture



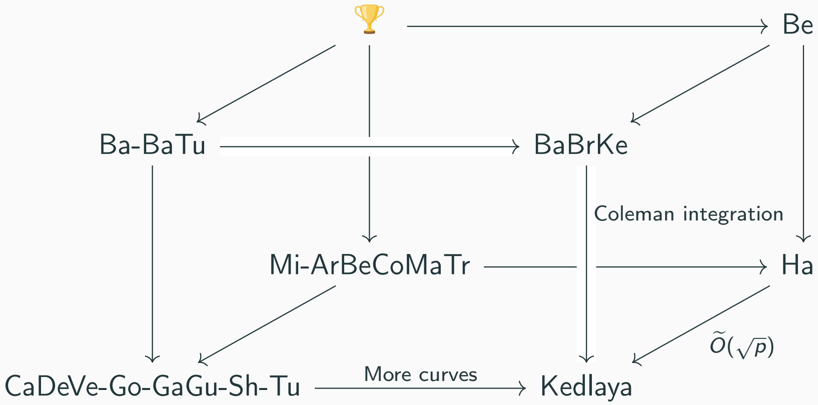
Arul, Balakrishnan, Best, Bradshaw, Castryk, Costa, Denef, Gaudry, Gurel, Harvey, Kedlaya, Magner, Minzlaff, Shieh, Triantafillou, Tuitman, Vercauteren, and more...

The big picture



Arul, Balakrishnan, Best, Bradshaw, Castryk, Costa, Denef, Gaudry, Gurel, Harvey, Kedlaya, Magner, Minzlaff, Shieh, Triantafillou, Tuitman, Vercauteren, and more. . .

The big picture



Arul, Balakrishnan, Best, Bradshaw, Castryk, Costa, Denef, Gaudry, Gurel, Harvey, Kedlaya, Magner,

Minzlaff, Shieh, Triantafillou, Tuitman, Vercauteren, and more...

There is (at least) one dimension missing: Small p!

Motivation

(Explicit) Coleman integration is a central tool in (non-abelian) Chabauty and can be applied to find: heights, torsion points and regulators also, but, algorithms lag behind the related ones for zeta functions.

Motivation

(Explicit) Coleman integration is a central tool in (non-abelian) Chabauty and can be applied to find: heights, torsion points and regulators also, but, algorithms lag behind the related ones for zeta functions.

Longer term goals:

- Adapt descendants of Kedlaya's algorithm to compute (iterated) Coleman integrals, e.g.:

Motivation

(Explicit) Coleman integration is a central tool in (non-abelian) Chabauty and can be applied to find: heights, torsion points and regulators also, but, algorithms lag behind the related ones for zeta functions.

Longer term goals:

- Adapt descendants of Kedlaya's algorithm to compute (iterated) Coleman integrals, e.g.:
  - Larger characteristic (Harvey)

Motivation

(Explicit) Coleman integration is a central tool in (non-abelian) Chabauty and can be applied to find: heights, torsion points and regulators also, but, algorithms lag behind the related ones for zeta functions.

Longer term goals:

- Adapt descendants of Kedlaya's algorithm to compute (iterated) Coleman integrals, e.g.:
  - Larger characteristic (Harvey)
  - Average polynomial time (Harvey)

Motivation

(Explicit) Coleman integration is a central tool in (non-abelian) Chabauty and can be applied to find: heights, torsion points and regulators also, but, algorithms lag behind the related ones for zeta functions.

Longer term goals:

- Adapt descendants of Kedlaya's algorithm to compute (iterated) Coleman integrals, e.g.:
  - Larger characteristic (Harvey)
  - Average polynomial time (Harvey)
  - Effectiver average polynomial time (Harvey-Sutherland)

Motivation

(Explicit) Coleman integration is a central tool in (non-abelian) Chabauty and can be applied to find: heights, torsion points and regulators also, but, algorithms lag behind the related ones for zeta functions.

Longer term goals:

- Adapt descendants of Kedlaya's algorithm to compute (iterated) Coleman integrals, e.g.:
  - Larger characteristic (Harvey)
  - Average polynomial time (Harvey)
  - Effectiver average polynomial time (Harvey-Sutherland)
- Applications to rational points, combining congruence information for many primes, 1-step (Mordell-Weil) sieving.

Coleman integration

Throughout we take $X/\mathbf{Z}_p$ a genus g odd degree hyperelliptic curve, and p an odd prime. We pick a lift of the Frobenius map, $\phi^*\colon X \to X$, and write $A^\dagger$ (resp. $A_{\mathrm{loc}}(X)$) for overconvergent (resp. locally analytic) functions on X.

Coleman integration

Throughout we take $X/\mathbf{Z}_p$ a genus g odd degree hyperelliptic curve, and p an odd prime. We pick a lift of the Frobenius map, $\phi^*\colon X \to X$, and write $A^\dagger$ (resp. $A_{\mathrm{loc}}(X)$) for overconvergent (resp. locally analytic) functions on X.

Theorem (Coleman)

There is a $\mathbf{Q}_p$-linear map $\int_b^x \colon \Omega^1_{A^\dagger} \otimes \mathbf{Q}_p \to A_{\mathrm{loc}}(X)$ for which:

$$\mathrm{d} \circ \int_b^x = \mathrm{id}\colon \Omega^1_{A^\dagger} \otimes \mathbf{Q}_p \to \Omega^1_{loc} \quad \text{“FTC”}$$

$$\int_b^x \circ\, \mathrm{d} = \mathrm{id}\colon A^\dagger \hookrightarrow A_{\mathrm{loc}}$$

$$\int_b^x \phi^*\omega = \phi^* \int_b^x \omega \quad \text{“Frobenius equivariance”}$$

Reduction to reduction

Balakrishnan-Bradshaw-Kedlaya reduce the problem of computing all Coleman integrals of basis differentials ω_i of $H^1_{\mathrm{dR}}(X)$ between $\infty \in X$ and a point $x \in X(\mathbf{Q}_p)$, to:

1. Finding "tiny integrals" between nearby points,
2. Writing $\phi^*\omega_i - \mathrm{d}f_i = \sum_j a_{ij}\omega_j$ and evaluating the primitive f_i for a point P near x, for each i.

Reduction to reduction

Balakrishnan-Bradshaw-Kedlaya reduce the problem of computing all Coleman integrals of basis differentials ω_i of $H^1_{\mathrm{dR}}(X)$ between $\infty \in X$ and a point $x \in X(\mathbf{Q}_p)$, to:

1. Finding "tiny integrals" between nearby points,
2. Writing $\phi^*\omega_i - \mathrm{d}f_i = \sum_j a_{ij}\omega_j$ and evaluating the primitive f_i for a point P near x, for each i.

Applying ϕ^* to the basis $x^i\,\mathrm{d}x/2y$ for $i = 0, \ldots, 2g-1$ gives

$$\phi^*\omega_i \equiv \sum_{j=0}^{N-1} \sum_{r=0}^{(2g+1)j} B_{j,r} x^{p(i+r+1)-1} y^{-p(2j+1)+1} \frac{\mathrm{d}x}{2y} \pmod{p^N}$$

$B_{j,r} \in \mathbf{Z}_p$ are in terms of coefficients of the curve and binomial coefficients.

Kedlaya's algorithm

Theorem (Kedlaya)

The action of ϕ^* *on* $H^1_{\mathrm{MW}}(X)$ *(which determines the zeta function of* X*) can be computed in time*

$$\widetilde{O}(p).$$

Kedlaya's algorithm

Theorem (Kedlaya)

The action of ϕ^* *on* $H^1_{\mathrm{MW}}(X)$ *(which determines the zeta function of* X*) can be computed in time*

$$\widetilde{O}(p).$$

Theorem (Harvey)

If $p > (2g+1)(2N-1)$ *the action of* ϕ^* *on* $H^1_{\mathrm{MW}}(X)$ *can be computed in time*

$$\widetilde{O}(\sqrt{p}).$$

Kedlaya's algorithm

Theorem (Kedlaya)

The action of ϕ^* on $H^1_{\mathrm{MW}}(X)$ (which determines the zeta function of X) can be computed in time

$$\widetilde{O}(p).$$

Theorem (Harvey)

If $p > (2g+1)(2N-1)$ the action of ϕ^* on $H^1_{\mathrm{MW}}(X)$ can be computed in time

$$\widetilde{O}(\sqrt{p}).$$

The problem solved here is almost the same: determining a_{ij} s.t.

$$\phi^*\omega_i - \sum_j a_{ij}\omega_j \in \mathrm{image}(\mathrm{d}).$$

Primitive technology

Revised problem

Computing f along with $\omega - \mathrm{d}f$ when reducing degree.

For vanilla Kedlaya this is "easy", the reduction procedure is transparent, whenever we subtract $\mathrm{d}g$ to reduce, add g onto f.

For faster variants, this is not so simple!

The reduction process

Harvey uses **horizontal** and **vertical** reductions to find the action of Frobenius on cohomology,

The reduction process

Harvey uses **horizontal** and **vertical** reductions to find the action of Frobenius on cohomology, abstractly we have:

Spaces of differentials W_t, indexed by degree, each of dimension $2g$.

Goal

Reduce all differentials from W_t to a cohomologous one in W_0, write in terms of fixed basis of W_0.

The reduction process

Harvey uses **horizontal** and **vertical** reductions to find the action of Frobenius on cohomology, abstractly we have:

Spaces of differentials W_t, indexed by degree, each of dimension $2g$.

Goal

Reduce all differentials from W_t to a cohomologous one in W_0, write in terms of fixed basis of W_0.

Relations in the de Rham cohomology $\rightsquigarrow$ linear maps $R(t)\colon W_t \to W_{t-1}\,\forall t$, with $R(t)\omega \sim \omega$.

The reduction process

Harvey uses **horizontal** and **vertical** reductions to find the action of Frobenius on cohomology, abstractly we have:

Spaces of differentials W_t, indexed by degree, each of dimension $2g$.

Goal

Reduce all differentials from W_t to a cohomologous one in W_0, write in terms of fixed basis of W_0.

Relations in the de Rham cohomology $\rightsquigarrow$ linear maps $R(t)\colon W_t \to W_{t-1} \, \forall t$, with $R(t)\omega \sim \omega$. Want to evaluate

$$W_t \ni \omega \mapsto R(1)R(2)\cdots R(t-1)R(t)\omega \in W_0$$

$O(\sqrt{p})$

Key fact

Entries of $R(t)$ are fractions of *linear* functions of t, with $\mathbf{Z}_p$ coefficients; work of Bostan-Gaudry-Schost (& Harvey) $\Longrightarrow$ products can be interpolated

$R(a, b) = R(a+1) \cdots R(b) \rightsquigarrow R(a+1+t, b+t)$

This is what gives a $\widetilde{O}(\sqrt{p})$ algorithm.

$O(\sqrt{p})$

Key fact

Entries of $R(t)$ are fractions of *linear* functions of t, with $\mathbf{Z}_p$ coefficients; work of Bostan-Gaudry-Schost (& Harvey) $\Longrightarrow$ products can be interpolated

$R(a, b) = R(a+1) \cdots R(b) \rightsquigarrow R(a+1+t, b+t)$

This is what gives a $\widetilde{O}(\sqrt{p})$ algorithm.

We also want an evaluation of the primitive f_ω for which $\omega - \mathrm{d}f_\omega = R(t)\omega$. We can keep this extra data throughout the recurrence as f_ω is linear in ω.

$O(\sqrt{p})$

Key fact

Entries of $R(t)$ are fractions of *linear* functions of t, with $\mathbf{Z}_p$ coefficients; work of Bostan-Gaudry-Schost (& Harvey) $\implies$ products can be interpolated

$R(a, b) = R(a+1) \cdots R(b) \rightsquigarrow R(a+1+t, b+t)$

This is what gives a $\widetilde{O}(\sqrt{p})$ algorithm.

We also want an evaluation of the primitive f_ω for which $\omega - \mathrm{d}f_\omega = R(t)\omega$. We can keep this extra data throughout the recurrence as f_ω is linear in ω.

Vital remark

We must use evaluations of primitives here, instead of trying to compute f as a power series.

A problem and a solution

Stumbling block

This is no longer linear in the index t! You cannot apply BGS to evaluate this recurrence faster.

A problem and a solution

Stumbling block

This is no longer linear in the index t! You cannot apply BGS to evaluate this recurrence faster.

Horner to the rescue!

Instead of computing a series $\sum_{i=0}^{N} a_i x^i$ by computing sequentially

$$\left(\sum_{i=t}^{N} a_i x^i \right)_{t=N,N-1,\ldots,0}$$

we can instead compute

$$((\cdots((a_N)x + a_{N-1})x + \cdots)x + a_0)$$

from the inside to the out. This *is* an iterated composition of linear functions, each of which is linear in the index t.

Explicit recurrence

In matrix form we augment the (numerators of) the reduction matrices:

$$
\begin{array}{c}
\\
y^{-2(t-1)}\,\mathrm{d}x/2y \\
\vdots \\
x^{2g-1}y^{-2(t-1)}\,\mathrm{d}x/2y \\
f(P)
\end{array}
\begin{array}{c}
\begin{array}{cccc} y^{-2t}\,\mathrm{d}x/2y & \cdots & x^{2g-1}y^{-2t}\,\mathrm{d}x/2y & f(P) \end{array} \\
\left(\begin{array}{ccc|c}
(2t-1)r_{0,0}+2s'_{0,0} & \cdots & (2t-1)r_{2g-1,0}+2s'_{2g-1,0} & \\
\vdots & \ddots & \vdots & \\
(2t-1)r_{0,2g-1}+2s'_{0,2g-1} & \cdots & (2t-1)r_{2g-1,2g-1}+2s'_{2g-1,2g-1} & \\
\hline
-S_0(x) & \cdots & -S_{2g-1}(x) & y^{-2}D_V(t)
\end{array}\right)
\end{array}
$$

so that we keep in memory a vector $v \in W_t \times \mathbf{Q}_p$ which gives the evaluation at the end.

Many integrals simultaneously

We may wish to do this with multiple points in several residue disks. Instead of repeating the whole procedure (repeating computing the Frobenius matrix), augment with many points.

$$
\begin{array}{c|ccc|ccc}
 & y^{-2t}\,\mathrm{d}x/2y & \cdots & x^{2g-1}y^{-2t}\,\mathrm{d}x/2y & f(P_1) & \cdots & f(P_L) \\
\hline
y^{-2(t-1)}\,\mathrm{d}x/y & (2t-1)r_{0,0}+2s'_{0,0} & \cdots & (2t-1)r_{2g-1,0}+2s'_{2g-1,0} & & & \\
\vdots & \vdots & \ddots & \vdots & & & \\
x^{2g-1}y^{-2(t-1)}\,\mathrm{d}x/y & (2t-1)r_{0,2g-1}+2s'_{0,2g-1} & \cdots & (2t-1)r_{2g-1,2g-1}+2s'_{2g-1,2g-1} & & & \\
\hline
f(P_1) & -S_0(x(P_1)) & \cdots & -S_{2g-1}(x(P_1)) & y^{-2}(P_1)D_V(t) & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
f(P_L) & -S_0(x(P_L)) & \cdots & -S_{2g-1}(x(P_L)) & 0 & \cdots & y(P_L)^{-2}D_V(t)
\end{array}
$$

Many integrals simultaneously

We may wish to do this with multiple points in several residue disks. Instead of repeating the whole procedure (repeating computing the Frobenius matrix), augment with many points.

$$
\begin{array}{c}
 \\
y^{-2(t-1)}\,\mathrm{d}x/y \\ \vdots \\ x^{2g-1}y^{-2(t-1)}\,\mathrm{d}x/y \\ f(P_1) \\ \vdots \\ f(P_L)
\end{array}
\begin{array}{c}
\begin{array}{ccccccc}
y^{-2t}\,\mathrm{d}x/2y & \cdots & x^{2g-1}y^{-2t}\,\mathrm{d}x/2y & f(P_1) & \cdots & f(P_L)
\end{array}\\
\left(\begin{array}{ccc|ccc}
(2t-1)r_{0,0}+2s'_{0,0} & \cdots & (2t-1)r_{2g-1,0}+2s'_{2g-1,0} & & & \\
\vdots & \ddots & \vdots & & & \\
(2t-1)r_{0,2g-1}+2s'_{0,2g-1} & \cdots & (2t-1)r_{2g-1,2g-1}+2s'_{2g-1,2g-1} & & & \\
\hline
-S_0(x(P_1)) & \cdots & -S_{2g-1}(x(P_1)) & y^{-2}(P_1)D_V(t) & \cdots & 0 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
-S_0(x(P_L)) & \cdots & -S_{2g-1}(x(P_L)) & 0 & \cdots & y(P_L)^{-2}D_V(t)
\end{array}\right)
\end{array}
$$

Note

This matrix and its iterates have the same fixed form, when running BGS don't try and interpolate entries that are always 0 $\rightsquigarrow$ better run time.

Thanks for listening!

Questions/comments?

Some amusing factoids

Core algorithm is potentially more general? Evaluation of primitives is “integration”, but maybe of a different type?

It is faster to evaluate the power series this way than to evaluate the power series you get from vanilla Kedlaya, even when multiple points are needed computing the power series is no use.