Number theory in (or not (yet) in) Sage

Kiran S. Kedlaya

Department of Mathematics, University of California, San Diego
kedlaya@ucsd.edu
http://kskedlaya.org/slides/

Sage Days: Opening Workshop for a Year of Coding Sprints
Institute for Mathematics and Its Applications
University of Minnesota
August 24, 2017

Contents

Goal of this talk

The goal of this talk is to demonstrate some of the most common examples of computations of interest to number theorists. These are feasible enough that they either *are* or *should be* implemented.

However, I will mostly avoid discussion of *how* to do specific things, aside from mentioning *which* software can handle them at present. Besides Sage, the most commonly used packages for number theory are Pari (which is included in Sage but will be discussed separately) and Magma (which is not open-source).

Goal of this talk

The goal of this talk is to demonstrate some of the most common examples of computations of interest to number theorists. These are feasible enough that they either *are* or *should be* implemented.

However, I will mostly avoid discussion of *how* to do specific things, aside from mentioning *which* software can handle them at present. Besides Sage, the most commonly used packages for number theory are Pari (which is included in Sage but will be discussed separately) and Magma (which is not open-source).

Contents

Primality testing and integer factorization

Euclid showed that every positive integer admits a unique (up to permutation) factorization as a product of primes. Determining whether a given integer is prime, and finding its prime factors if not, has been a major computational issue in number theory ever since. For example, Fermat conjectured that the integers

$$2^{2^n} + 1 \qquad (n = 0, 1, \dots)$$

are all prime, but Euler disproved this with some effort. Nowadays...

```
sage: factor(2^(2^5)+1)
641 * 6700417
```

Primality testing and integer factorization

Euclid showed that every positive integer admits a unique (up to permutation) factorization as a product of primes. Determining whether a given integer is prime, and finding its prime factors if not, has been a major computational issue in number theory ever since. For example, Fermat conjectured that the integers

$$2^{2^n} + 1 \qquad (n = 0, 1, \dots)$$

are all prime, but Euler disproved this with some effort. Nowadays...

```
sage: factor(2^(2^5)+1)
641 * 6700417
```

Primality testing

To generate an RSA modulus, one needs to be able to produce large prime numbers. Fortunately, it is easy to test for primality, using Fermat's little theorem

$$\gcd(a, p) = 1 \Longrightarrow a^{p-1} \equiv 1 \pmod{p}$$

as a first screen. This is implemented in *many* places; Pari provides this functionality to Sage:

```
sage: time p = next_prime(2^2048)
CPU times: user 35.2 s, sys: 0 ns, total: 35.2 s
Wall time: 35.3 s
sage: p - 2^2048
981
```

Primality testing

To generate an RSA modulus, one needs to be able to produce large prime numbers. Fortunately, it is easy to test for primality, using Fermat's little theorem

$$\gcd(a, p) = 1 \Longrightarrow a^{p-1} \equiv 1 \pmod{p}$$

as a first screen. This is implemented in *many* places; Pari provides this functionality to Sage:

```
sage: time p = next_prime(2^2048)
CPU times: user 35.2 s, sys: 0 ns, total: 35.2 s
Wall time: 35.3 s
sage: p - 2^2048
981
```

Factorization

RSA security depends on the fact that finding the prime factorization of a composite integer is much harder than testing for primality! Once the numbers get big enough, this is usually done using sort of *sieve algorithm*. For example, the *quadratic sieve* (for factoring N) involves finding a hash collision on the functor $x^2 \pmod N$.

Again, decent factorization exists in many systems, and again Sage relies on Pari:

```
sage: time factor(2^400 + 7)
CPU times: user 1min 4s, sys: 4 ms, total: 1min 4s
Wall time: 1min 4s
23 * 11800973 * 63817221995963 * 37381641955775213112 89 *
39880109696090444372755626550474667636180965440650557135\
4411167290234304335 11
```

Factorization

RSA security depends on the fact that finding the prime factorization of a composite integer is much harder than testing for primality! Once the numbers get big enough, this is usually done using sort of *sieve algorithm*. For example, the *quadratic sieve* (for factoring N) involves finding a hash collision on the functor $x^2 \pmod N$.

Again, decent factorization exists in many systems, and again Sage relies on Pari:

```
sage: time factor(2^400 + 7)
CPU times: user 1min 4s, sys: 4 ms, total: 1min 4s
Wall time: 1min 4s
23 * 11800973 * 63817221995963 * 37381641955775213112 89 *
39880109696090444372755626550474667636180965440650557135\
4411167290234304335 11
```

Factorization

RSA security depends on the fact that finding the prime factorization of a composite integer is much harder than testing for primality! Once the numbers get big enough, this is usually done using sort of *sieve algorithm*. For example, the *quadratic sieve* (for factoring N) involves finding a hash collision on the functor $x^2 \pmod N$.

Again, decent factorization exists in many systems, and again Sage relies on Pari:

```
sage: time factor(2^400 + 7)
CPU times: user 1min 4s, sys: 4 ms, total: 1min 4s
Wall time: 1min 4s
23 * 11800973 * 63817221995963 * 37381641955775213112 89 *
39880109696090444372755626550474667636180965440650557135\
4411167290234304335 11
```

Discrete logarithms

Various protocols based on modular exponentiation (e.g., Diffie-Hellman) depend on the difficulty of inverting this map, i.e., of computing *discrete logarithms*. This is similar to factorization, and yet again comes to Sage via Pari:

```
sage: p = next_prime(2^100); F = GF(p); a = F(3); b = F(2)
sage: time n = a.log(b)
CPU times: user 8.66 s, sys: 0 ns, total: 8.66 s
Wall time: 8.67 s
sage: print n, b^n
1219919803936672396839414039291 3
```

But the role of the group $\mathbb{F}_p^\times$ here can be played by other interesting computable finite groups, such as the rational points of an elliptic curve over $\mathbb{F}_p$. This brings us to...

Discrete logarithms

Various protocols based on modular exponentiation (e.g., Diffie-Hellman) depend on the difficulty of inverting this map, i.e., of computing *discrete logarithms*. This is similar to factorization, and yet again comes to Sage via Pari:

```
sage: p = next_prime(2^100); F = GF(p); a = F(3); b = F(2)
sage: time n = a.log(b)
CPU times: user 8.66 s, sys: 0 ns, total: 8.66 s
Wall time: 8.67 s
sage: print n, b^n
1219919803936672396839414039291 3
```

But the role of the group $\mathbb{F}_p^\times$ here can be played by other interesting computable finite groups, such as the rational points of an elliptic curve over $\mathbb{F}_p$. This brings us to...

Discrete logarithms

Various protocols based on modular exponentiation (e.g., Diffie-Hellman) depend on the difficulty of inverting this map, i.e., of computing *discrete logarithms*. This is similar to factorization, and yet again comes to Sage via Pari:

```
sage: p = next_prime(2^100); F = GF(p); a = F(3); b = F(2)
sage: time n = a.log(b)
CPU times: user 8.66 s, sys: 0 ns, total: 8.66 s
Wall time: 8.67 s
sage: print n, b^n
1219919803936672396839414039291 3
```

But the role of the group $\mathbb{F}_p^\times$ here can be played by other interesting computable finite groups, such as the rational points of an elliptic curve over $\mathbb{F}_p$. This brings us to...

Contents

Zeta functions

For X an algebraic variety over a finite field $\mathbb{F}_q$, its *zeta function* is

$$\zeta(X, T) = \exp\left(\sum_{n=1}^{\infty} \#X(\mathbb{F}_{q^n}) \frac{T^n}{n}\right) \in \mathbb{Z}[\![T]\!];$$

it represents a rational function (Dwork). Typically, one can bound the degree of this rational function in terms of geometric invariants; then in principle one can compute $\zeta(X, T)$ by enumerating $X(\mathbb{F}_{q^n})$ for enough values of n, but this is often infeasible.

For example, for X a curve of genus g over $\mathbb{F}_q$,

$$\zeta(X, T) = \frac{1 + a_1 T + \cdots + a_{2g-1} T^{2g-1} + q^g T^{2g}}{(1 - T)(1 - qT)}$$

where $a_{2g-i} = q^{g-i} a_i$.

Zeta functions

For X an algebraic variety over a finite field $\mathbb{F}_q$, its *zeta function* is

$$\zeta(X, T) = \exp\left(\sum_{n=1}^{\infty} \#X(\mathbb{F}_{q^n}) \frac{T^n}{n}\right) \in \mathbb{Z}[\![T]\!];$$

it represents a rational function (Dwork). Typically, one can bound the degree of this rational function in terms of geometric invariants; then in principle one can compute $\zeta(X, T)$ by enumerating $X(\mathbb{F}_{q^n})$ for enough values of n, but this is often infeasible.

For example, for X a curve of genus g over $\mathbb{F}_q$,

$$\zeta(X, T) = \frac{1 + a_1 T + \cdots + a_{2g-1} T^{2g-1} + q^g T^{2g}}{(1 - T)(1 - qT)}$$

where $a_{2g-i} = q^{g-i} a_i$.

Zeta functions of elliptic curves

Suppose $g = 1$. Then $a = q + 1 - \#X(\mathbb{F}_q)$ satisfies $|a| \leq 2\sqrt{q}$ (Hasse) and

$$\zeta(X, T) = \frac{1 - aT + qT^2}{(1 - T)(1 - qT)}$$

If one needs $X(\mathbb{F}_q)$ for crypto, one can find a using "baby step-giant step" (Shanks), or by computing $a \pmod{\ell}$ for various ℓ (Schoof). Via Pari...

```
sage: p = next_prime(2^300)
sage: E = EllipticCurve(GF(p), [1,8]); E
Elliptic Curve defined by y^2 = x^3 + x + 8 over ...
sage: time z = E.zeta_function()
CPU times: user 12.2 s, sys: 20 ms, total: 12.3 s
Wall time: 12.3 s
sage: -z.numerator().coefficients()[1] # this is "a"
1910426102709561065930788840630164762477616422
```

Zeta functions of elliptic curves

Suppose $g = 1$. Then $a = q + 1 - \#X(\mathbb{F}_q)$ satisfies $|a| \leq 2\sqrt{q}$ (Hasse) and

$$\zeta(X, T) = \frac{1 - aT + qT^2}{(1 - T)(1 - qT)}$$

If one needs $X(\mathbb{F}_q)$ for crypto, one can find a using "baby step-giant step" (Shanks), or by computing $a \pmod{\ell}$ for various ℓ (Schoof). Via Pari...

```
sage: p = next_prime(2^300)
sage: E = EllipticCurve(GF(p), [1,8]); E
Elliptic Curve defined by y^2 = x^3 + x + 8 over ...
sage: time z = E.zeta_function()
CPU times: user 12.2 s, sys: 20 ms, total: 12.3 s
Wall time: 12.3 s
sage: -z.numerator().coefficients()[1] # this is "a"
1910426102709561065930788840630164762477616422
```

Zeta functions of hyperelliptic curves

For $g > 1$, $X(\mathbb{F}_q)$ is not a group, but the class group of X equals $J(\mathbb{F}_q)$ for J the Jacobian of X. The order of this group equals $P(1)$ where P is the numerator of $\zeta(X, T)$.

For X a hyperelliptic curve, this can be computed using p-adic cohomology (K, Harvey). Sage (partly using Pari) provides this:

```
sage: p = next_prime(2^20)
sage: P.<x> = PolynomialRing(GF(p))
sage: H = HyperellipticCurve(x^11 + x + 1)
sage: time z = H.zeta_function()
CPU times: user 3.1 s, sys: 4 ms, total: 3.11 s
Wall time: 3.11 s
sage: z.numerator().coefficients()[0:4]
[1, -1034, 1524507, -1229851334]
```

Zeta functions of hyperelliptic curves

For $g > 1$, $X(\mathbb{F}_q)$ is not a group, but the class group of X equals $J(\mathbb{F}_q)$ for J the Jacobian of X. The order of this group equals $P(1)$ where P is the numerator of $\zeta(X, T)$.

For X a hyperelliptic curve, this can be computed using p-adic cohomology (K, Harvey). Sage (partly using Pari) provides this:

```
sage: p = next_prime(2^20)
sage: P.<x> = PolynomialRing(GF(p))
sage: H = HyperellipticCurve(x^11 + x + 1)
sage: time z = H.zeta_function()
CPU times: user 3.1 s, sys: 4 ms, total: 3.11 s
Wall time: 3.11 s
sage: z.numerator().coefficients()[0:4]
[1, -1034, 1524507, -1229851334]
```

Zeta functions of hyperelliptic curves

For $g > 1$, $X(\mathbb{F}_q)$ is not a group, but the class group of X equals $J(\mathbb{F}_q)$ for J the Jacobian of X. The order of this group equals $P(1)$ where P is the numerator of $\zeta(X, T)$.

For X a hyperelliptic curve, this can be computed using p-adic cohomology (K, Harvey). Sage (partly using Pari) provides this:

```
sage: p = next_prime(2^20)
sage: P.<x> = PolynomialRing(GF(p))
sage: H = HyperellipticCurve(x^11 + x + 1)
sage: time z = H.zeta_function()
CPU times: user 3.1 s, sys: 4 ms, total: 3.11 s
Wall time: 3.11 s
sage: z.numerator().coefficients()[0:4]
[1, -1034, 1524507, -1229851334]
```

Zeta functions of hyperelliptic curves

For $g > 1$, $X(\mathbb{F}_q)$ is not a group, but the class group of X equals $J(\mathbb{F}_q)$ for J the Jacobian of X. The order of this group equals $P(1)$ where P is the numerator of $\zeta(X, T)$.

For X a hyperelliptic curve, this can be computed using p-adic cohomology (K, Harvey). Sage (partly using Pari) provides this:

```
sage: p = next_prime(2^20)
sage: P.<x> = PolynomialRing(GF(p))
sage: H = HyperellipticCurve(x^11 + x + 1)
sage: time z = H.zeta_function()
CPU times: user 3.1 s, sys: 4 ms, total: 3.11 s
Wall time: 3.11 s
sage: z.numerator().coefficients()[0:4]
[1, -1034, 1524507, -1229851334]
```

Variants

Not yet available in Sage. Coding sprint, anyone?

- Magma provides a generalization of the previous method that applies to arbitrary curves (Tuitman).
- It is possible to do something similar for some higher-dimensional varieties, such as smooth projective hypersurfaces (Abbott–K–Roe) and nondegenerate hypersurfaces in toric varieties (Costa–Harvey–K). Costa has a C implementation.
- If one starts with a hyperelliptic curve over $\mathbb{Q}$ and considers its reductions modulo various primes, one can amortize the computation of the zeta functions (Harvey, Harvey–Sutherland). Sutherland has a C implementation (for $g \leq 3$).

This last case is needed to compute the curve's *L-function*...

Variants

Not yet available in Sage. Coding sprint, anyone?

- Magma provides a generalization of the previous method that applies to arbitrary curves (Tuitman).
- It is possible to do something similar for some higher-dimensional varieties, such as smooth projective hypersurfaces (Abbott–K–Roe) and nondegenerate hypersurfaces in toric varieties (Costa–Harvey–K). Costa has a C implementation.
- If one starts with a hyperelliptic curve over $\mathbb{Q}$ and considers its reductions modulo various primes, one can amortize the computation of the zeta functions (Harvey, Harvey–Sutherland). Sutherland has a C implementation (for $g \leq 3$).

This last case is needed to compute the curve's *L-function*...

Variants

Not yet available in Sage. Coding sprint, anyone?

- Magma provides a generalization of the previous method that applies to arbitrary curves (Tuitman).
- It is possible to do something similar for some higher-dimensional varieties, such as smooth projective hypersurfaces (Abbott–K–Roe) and nondegenerate hypersurfaces in toric varieties (Costa–Harvey–K). Costa has a C implementation.
- If one starts with a hyperelliptic curve over $\mathbb{Q}$ and considers its reductions modulo various primes, one can amortize the computation of the zeta functions (Harvey, Harvey–Sutherland). Sutherland has a C implementation (for $g \leq 3$).

This last case is needed to compute the curve's *L-function*...

Variants

Not yet available in Sage. Coding sprint, anyone?

- Magma provides a generalization of the previous method that applies to arbitrary curves (Tuitman).
- It is possible to do something similar for some higher-dimensional varieties, such as smooth projective hypersurfaces (Abbott–K–Roe) and nondegenerate hypersurfaces in toric varieties (Costa–Harvey–K). Costa has a C implementation.
- If one starts with a hyperelliptic curve over $\mathbb{Q}$ and considers its reductions modulo various primes, one can amortize the computation of the zeta functions (Harvey, Harvey–Sutherland). Sutherland has a C implementation (for $g \leq 3$).

This last case is needed to compute the curve's *L-function*...

Variants

Not yet available in Sage. Coding sprint, anyone?

- Magma provides a generalization of the previous method that applies to arbitrary curves (Tuitman).
- It is possible to do something similar for some higher-dimensional varieties, such as smooth projective hypersurfaces (Abbott–K–Roe) and nondegenerate hypersurfaces in toric varieties (Costa–Harvey–K). Costa has a C implementation.
- If one starts with a hyperelliptic curve over $\mathbb{Q}$ and considers its reductions modulo various primes, one can amortize the computation of the zeta functions (Harvey, Harvey–Sutherland). Sutherland has a C implementation (for $g \leq 3$).

This last case is needed to compute the curve's *L-function*...

Contents

L-functions

An *L-function* is a complex-analytic function represented by a *Dirichlet series* $L(s) = \sum_{n=1}^{\infty} a_n n^{-s}$ satisfying certain extra conditions. Typically, this series factors as an *Euler product* $\prod_p L_p(s)$; e.g., the Riemann zeta function

$$\zeta(s) = \sum_{n=1}^{\infty} n^{-s} = \prod_p (1 - p^{-s})^{-1}.$$

Many different types of L-functions arise in number theory. The L-Functions and Modular Forms Database (http://www.lmfdb.org) is a systematic catalog of L-functions in number theory and associated objects; its construction leads to many interesting computational problems!

One general problem is to evaluate the function outside the domain of absolute convergence of the power series. This is handled by code of Dokchitser in Sage and Magma.

L-functions

An *L-function* is a complex-analytic function represented by a *Dirichlet series* $L(s) = \sum_{n=1}^{\infty} a_n n^{-s}$ satisfying certain extra conditions. Typically, this series factors as an *Euler product* $\prod_p L_p(s)$; e.g., the Riemann zeta function

$$\zeta(s) = \sum_{n=1}^{\infty} n^{-s} = \prod_p (1 - p^{-s})^{-1}.$$

Many different types of L-functions arise in number theory. The L-Functions and Modular Forms Database (`http://www.lmfdb.org`) is a systematic catalog of L-functions in number theory and associated objects; its construction leads to many interesting computational problems!

One general problem is to evaluate the function outside the domain of absolute convergence of the power series. This is handled by code of Dokchitser in Sage and Magma.

L-functions

An *L-function* is a complex-analytic function represented by a *Dirichlet series* $L(s) = \sum_{n=1}^{\infty} a_n n^{-s}$ satisfying certain extra conditions. Typically, this series factors as an *Euler product* $\prod_p L_p(s)$; e.g., the Riemann zeta function

$$\zeta(s) = \sum_{n=1}^{\infty} n^{-s} = \prod_p (1 - p^{-s})^{-1}.$$

Many different types of L-functions arise in number theory. The L-Functions and Modular Forms Database (http://www.lmfdb.org) is a systematic catalog of L-functions in number theory and associated objects; its construction leads to many interesting computational problems!

One general problem is to evaluate the function outside the domain of absolute convergence of the power series. This is handled by code of Dokchitser in Sage and Magma.

L-functions of elliptic (and other) curves

Recall from Jen's talk that an elliptic curve E over $\mathbb{Q}$ has an associated L-function $L(E,s) = \prod_p L_p(E,s)$ where for the (all but finitely many) p for which E reduces mod p to an elliptic curve, $L_p(E,s)$ is the numerator of the zeta function of that reduction evaluated at $T = p^{-s}$.

This function extends analytically to $\mathbb{C}$ (Wiles, Taylor, etc.) and its order of vanishing at $s = 1$ is conjectured to equal the rank of $E(\mathbb{Q})$ (Birch–Swinnerton-Dyer).

A similar story holds (conjecturally) for an arbitrary curve over $\mathbb{Q}$.

L-functions of elliptic (and other) curves

Recall from Jen's talk that an elliptic curve E over $\mathbb{Q}$ has an associated L-function $L(E,s) = \prod_p L_p(E,s)$ where for the (all but finitely many) p for which E reduces mod p to an elliptic curve, $L_p(E,s)$ is the numerator of the zeta function of that reduction evaluated at $T = p^{-s}$.

This function extends analytically to $\mathbb{C}$ (Wiles, Taylor, etc.) and its order of vanishing at $s = 1$ is conjectured to equal the rank of $E(\mathbb{Q})$ (Birch–Swinnerton-Dyer).

A similar story holds (conjecturally) for an arbitrary curve over $\mathbb{Q}$.

L-functions of elliptic (and other) curves

Recall from Jen's talk that an elliptic curve E over $\mathbb{Q}$ has an associated L-function $L(E,s) = \prod_p L_p(E,s)$ where for the (all but finitely many) p for which E reduces mod p to an elliptic curve, $L_p(E,s)$ is the numerator of the zeta function of that reduction evaluated at $T = p^{-s}$.

This function extends analytically to $\mathbb{C}$ (Wiles, Taylor, etc.) and its order of vanishing at $s = 1$ is conjectured to equal the rank of $E(\mathbb{Q})$ (Birch–Swinnerton-Dyer).

A similar story holds (conjecturally) for an arbitrary curve over $\mathbb{Q}$.

The Sato-Tate conjecture and generalizations

For E an elliptic curve over $\mathbb{Q}$, the factors $L_p(E,s) = 1 - a_p p^{-s} + p^{1-2s}$ have the property that the limiting distribution of the values $a_p/\sqrt{p}$ in $[-2,2]$ can only take one of two values, depending on whether or not E has "complex multiplication" (Taylor, Clozel, etc.).

Something similar is conjectured for other curves, but with more exceptions possible. For instance, for genus 2 curves, one expects 34 distinct distributions to occur (Fité–K–Rotger–Sutherland).

See Sutherland's home page `http://math.mit.edu/~drew` for pictures.

The Sato-Tate conjecture and generalizations

For E an elliptic curve over $\mathbb{Q}$, the factors $L_p(E,s) = 1 - a_p p^{-s} + p^{1-2s}$ have the property that the limiting distribution of the values $a_p/\sqrt{p}$ in $[-2,2]$ can only take one of two values, depending on whether or not E has "complex multiplication" (Taylor, Clozel, etc.).

Something similar is conjectured for other curves, but with more exceptions possible. For instance, for genus 2 curves, one expects 34 distinct distributions to occur (Fité–K–Rotger–Sutherland).

See Sutherland's home page `http://math.mit.edu/~drew` for pictures.

The Sato-Tate conjecture and generalizations

For E an elliptic curve over $\mathbb{Q}$, the factors $L_p(E,s) = 1 - a_p p^{-s} + p^{1-2s}$ have the property that the limiting distribution of the values $a_p/\sqrt{p}$ in $[-2,2]$ can only take one of two values, depending on whether or not E has "complex multiplication" (Taylor, Clozel, etc.).

Something similar is conjectured for other curves, but with more exceptions possible. For instance, for genus 2 curves, one expects 34 distinct distributions to occur (Fité–K–Rotger–Sutherland).

See Sutherland's home page `http://math.mit.edu/~drew` for pictures.

Modular forms

Another source of L-functions are modular forms; these are certain analytic functions on the upper half-plane which transform nicely under the action of certain congruence subgroups of $\mathrm{SL}_2(\mathbb{Z})$. There are (at least) three approaches to computing modular forms:

- *modular symbols* (Manin): compute the action of Hecke operators on singular cohomology of modular curves via triangulations. Implemented in Magma (Stein) and Sage (Stein).
- *trace formulas* (Cohen): implemented in Pari (Cohen).
- *quadratic forms* (Birch): use the Jacquet–Langlands correspondence to pass to a quaternion algebra, where reduction theory of ternary quadratic forms kicked in. Implemented in C++ (Hein); I'm currently trying to get this into Sage (ticket #23342).

Modular forms

Another source of L-functions are modular forms; these are certain analytic functions on the upper half-plane which transform nicely under the action of certain congruence subgroups of $\mathrm{SL}_2(\mathbb{Z})$. There are (at least) three approaches to computing modular forms:

- *modular symbols* (Manin): compute the action of Hecke operators on singular cohomology of modular curves via triangulations. Implemented in Magma (Stein) and Sage (Stein).
- *trace formulas* (Cohen): implemented in Pari (Cohen).
- *quadratic forms* (Birch): use the Jacquet–Langlands correspondence to pass to a quaternion algebra, where reduction theory of ternary quadratic forms kicked in. Implemented in C++ (Hein); I'm currently trying to get this into Sage (ticket #23342).

Modular forms

Another source of L-functions are modular forms; these are certain analytic functions on the upper half-plane which transform nicely under the action of certain congruence subgroups of $\mathrm{SL}_2(\mathbb{Z})$. There are (at least) three approaches to computing modular forms:

- *modular symbols* (Manin): compute the action of Hecke operators on singular cohomology of modular curves via triangulations. Implemented in Magma (Stein) and Sage (Stein).
- *trace formulas* (Cohen): implemented in Pari (Cohen).
- *quadratic forms* (Birch): use the Jacquet–Langlands correspondence to pass to a quaternion algebra, where reduction theory of ternary quadratic forms kicked in. Implemented in C++ (Hein); I'm currently trying to get this into Sage (ticket #23342).

Modular forms

Another source of L-functions are modular forms; these are certain analytic functions on the upper half-plane which transform nicely under the action of certain congruence subgroups of $\mathrm{SL}_2(\mathbb{Z})$. There are (at least) three approaches to computing modular forms:

- *modular symbols* (Manin): compute the action of Hecke operators on singular cohomology of modular curves via triangulations. Implemented in Magma (Stein) and Sage (Stein).
- *trace formulas* (Cohen): implemented in Pari (Cohen).
- *quadratic forms* (Birch): use the Jacquet–Langlands correspondence to pass to a quaternion algebra, where reduction theory of ternary quadratic forms kicked in. Implemented in C++ (Hein); I'm currently trying to get this into Sage (ticket #23342).

Automorphic forms

Modular forms and elliptic curves over $\mathbb{Q}$ are related by an instance of the *Langlands correspondence*. This can be used to tabulate elliptic curves (Cremona).

Similar relationship are predicted between other *automorphic forms* (for algebraic groups other than SL_2) and other algebro-geometric objects. For example, genus 2 curves should be related to Siegel modular forms; a precise statement is the *paramodular conjecture* (Brumer–Kramer).

However, despite having general predictions from Langlands, it is not always straightforward to match things up this way, especially in the automorphic-to-geometric direction. One good source of geometric objects are *hypergeometric motives*, which are supported in Magma and maybe soon in Sage (ticket #23671).

Automorphic forms

Modular forms and elliptic curves over $\mathbb{Q}$ are related by an instance of the *Langlands correspondence*. This can be used to tabulate elliptic curves (Cremona).

Similar relationship are predicted between other *automorphic forms* (for algebraic groups other than SL_2) and other algebro-geometric objects. For example, genus 2 curves should be related to Siegel modular forms; a precise statement is the *paramodular conjecture* (Brumer–Kramer).

However, despite having general predictions from Langlands, it is not always straightforward to match things up this way, especially in the automorphic-to-geometric direction. One good source of geometric objects are *hypergeometric motives*, which are supported in Magma and maybe soon in Sage (ticket #23671).

Automorphic forms

Modular forms and elliptic curves over $\mathbb{Q}$ are related by an instance of the *Langlands correspondence*. This can be used to tabulate elliptic curves (Cremona).

Similar relationship are predicted between other *automorphic forms* (for algebraic groups other than SL_2) and other algebro-geometric objects. For example, genus 2 curves should be related to Siegel modular forms; a precise statement is the *paramodular conjecture* (Brumer–Kramer).

However, despite having general predictions from Langlands, it is not always straightforward to match things up this way, especially in the automorphic-to-geometric direction. One good source of geometric objects are *hypergeometric motives*, which are supported in Magma and maybe soon in Sage (ticket #23671).

Contents

Additional topics

- Rational points on curves: see Jen's talk.
- Pari provides basic tools for number fields, like computing class numbers. Also available in Magma.
- Pari provides LLL for lattices, and applications like converting a floating-point real number into a rational or algebraic number. Also available in Magma.
- Add your favorite example here...

Additional topics

- Rational points on curves: see Jen's talk.
- Pari provides basic tools for number fields, like computing class numbers. Also available in Magma.
- Pari provides LLL for lattices, and applications like converting a floating-point real number into a rational or algebraic number. Also available in Magma.
- Add your favorite example here...

Additional topics

- Rational points on curves: see Jen's talk.
- Pari provides basic tools for number fields, like computing class numbers. Also available in Magma.
- Pari provides LLL for lattices, and applications like converting a floating-point real number into a rational or algebraic number. Also available in Magma.
- Add your favorite example here...

Additional topics

- Rational points on curves: see Jen's talk.
- Pari provides basic tools for number fields, like computing class numbers. Also available in Magma.
- Pari provides LLL for lattices, and applications like converting a floating-point real number into a rational or algebraic number. Also available in Magma.
- Add your favorite example here...

Additional topics

- Rational points on curves: see Jen's talk.
- Pari provides basic tools for number fields, like computing class numbers. Also available in Magma.
- Pari provides LLL for lattices, and applications like converting a floating-point real number into a rational or algebraic number. Also available in Magma.
- Add your favorite example here...