$L$-functions in PARI/GP

Karim Belabas (with Bill Allombert, Henri Cohen, Pascal Molin)
First part: Theory
Let $\Gamma_{\mathbb{R}}(s) = \pi^{-s/2}\Gamma(s/2)$, where $\Gamma$ is Euler’s gamma function; given a $d$-tuple $A = [\alpha_1, \ldots, \alpha_d] \in \mathbb{C}^d$, let $\gamma_A := \prod_{\alpha \in A} \Gamma_R(s + \alpha)$.

Given

- a sequence $a = (a_n)_{n \geq 1}$ of complex numbers such that $a_1 = 1$,
- a positive conductor $N \in \mathbb{Z}_{>0}$,
- a gamma factor $\gamma_A$ as above,

we consider the Dirichlet series

$$L(a, s) = \sum_{n \geq 1} a_n n^{-s}$$

and the associated completed function

$$\Lambda_{N,A}(a, s) = N^{s/2} \cdot \gamma_A(s) \cdot L(a, s).$$
A weak $L$-function is a Dirichlet series

$$L(s) = \sum_{n \geq 1} a_n n^{-s}$$

such that

- The coefficients $a_n = O_\varepsilon (n^{C+\varepsilon})$ have polynomial growth. Equivalently, $L(s)$ converges absolutely in some right half-plane $\Re(s) > C + 1$.
- The function $L(s)$ has a meromorphic continuation to the whole complex plane with finitely many poles.

This becomes an $L$-function if it satisfies a functional equation: there exist a “dual” sequence $a^*$ defining a weak $L$-function $L(a^*, s)$, an integer $k$, and completed functions

$$\Lambda(a, s) = N^{s/2} \gamma_A(s) \cdot L(a, s),$$

$$\Lambda(a^*, s) = N^{s/2} \gamma_A(s) \cdot L(a^*, s),$$

such that $\Lambda(a, k - s) = \Lambda(a^*, s)$ for all regular points.
In number theory, additional constraints arise

- \( a^* = \varepsilon \cdot \overline{a} \) for some root number \( \varepsilon \) of modulus 1; often, \( \varepsilon = \pm 1 \);
- the complex coefficients \( a \) live in the ring of integer of some fixed number field, often in \( \mathbb{Z} \) or a cyclotomic ring \( \mathbb{Z}[[\zeta]] \);
- the growth exponent such that \( a_n = O_{\varepsilon}(n^{C+\varepsilon}) \) can be taken as \( C = (k - 1)/2 \) if \( L \) is entire (Ramanujan-Petersson), and \( C = k - 1 \) otherwise;
- the \( L \)-function satisfies an Euler product \( L(s) = \prod_{p \text{ prime}} L_p(s) \), where the local factor \( L_p(s) \) is a rational function in \( p^{-s} \);
- the \( \alpha_i \) are integers, often in \{0, 1\}.

The current PARI implementation assumes that \( a^* = \varepsilon \cdot \overline{a} \) and chooses \( C \) as above; these restrictions are being removed.
To an $L$-function, we associate a Theta function via Mellin inversion: for positive real $t > 0$, we let

$$
\theta(a, t) := \frac{1}{2\pi i} \int_{\Re(s)=c} t^{-s} \Lambda(s) \, ds
$$

where $c$ is any positive real number $c > C + 1$ such that $c + \Re(a) > 0$ for all $a \in A$. In fact, we have

$$
\theta(a, t) = \sum_{n \geq 1} a_n K(nt/N^{1/2}) \quad \text{where} \quad K(t) := \frac{1}{2\pi i} \int_{\Re(s)=c} t^{-s} \gamma_A(s) \, ds
$$

and this function is analytic for complex $t$ such that $\Re(t^{2/d}) > 0$, i.e. in a cone containing the positive real half-line. The functional equation for $\Lambda$ translates into

$$
\theta(a, 1/t) - t^k \theta(a^*, t) = P_\Lambda(t),
$$

where $P_\Lambda$ is a polynomial in $t$ and $\log t$ given by the Taylor development of the polar part of $\Lambda$: there are no $\log$’s if all poles are simple, and $P = 0$ if $\Lambda$ is entire.
First Goal: Approximate $L(a, s)$, $\Lambda(a, s)$, $\theta(a, t)$ and their derivatives at regular points.

1. Compute the inverse Mellin transform of $\gamma_A(s)$:

$$G(x) = \frac{1}{2\pi i} \int_{\Re(s)=c} x^{-s} \gamma_A(s) \, ds.$$  

For large $x$, $G(x)$ decreases exponentially, roughly as $\exp(-d\pi \Re(x^{2/d}))$. Complexity $\tilde{O}(B^c)$ for absolute error $< 2^{-B}$ and $c(d) \leq 3$ (e.g. $c(1) = 1$).

2. Compute

$$\theta(a, t) = \sum_{n \geq 1} a_n G(nt/N^{1/2});$$

for $t \geq 1$, absolute error $2^{-B}$, use roughly $N^{1/2} B^{d/2}$ terms.
Main algorithms (2/2)

(3) Compute, for $h$ small enough, $\Lambda(a, s) \approx \sum_{n \in \mathbb{Z}} \Lambda(a, s + 2\pi in/h)$

$$= \text{explicit polar part} + h \sum_{m \geq 1} e^{mhs} \theta(a, e^{mh}) + h \sum_{m \geq 1} e^{mh(k-s)} \theta(a^*, e^{mh})$$

The coefficients $\theta(a, e^{mh})$, $\theta(a^*, e^{mh})$ are independent of $s$!

(4) Compute

$L(a, s) = \Lambda(a, s) N^{-s/2} / \gamma_A(s)$. 

Secundary Goal: If some of the quantities needed before are unknown (e.g. $N$ or $a_2$ or...), guess them from $\theta$’s functional equation evaluated in many points.
Second part: Practice
Data structures describing $L$ and Theta functions

In PARI/GP we have 3 levels of description for Theta or $L$-functions:

- An **Lmath** is an high-level description of the underlying mathematical situation, to which e.g., we associate the $a_p$ as traces of Frobenius elements; this is done via constructors to be described shortly.

- An **Ldata** is a low-level description, containing the complete datum $(a, a^*, A, k, N, \Lambda$'s polar part). This is obtained via the function `lfuncreate`.

- An **Linit** contains an Ldata and everything needed for fast *numerical* computations in a certain *domain*: it specifies
  
  (1) the functions to be considered either $L^{(j)}(s)$ or $\theta^{(j)}(t)$ for derivatives of order $j \leq m$, where $m$ is now fixed;

  (2) the range of arguments $t$ or $s$, respectively to certain cones and rectangular regions;

  (3) the output bit accuracy.

  This is obtained via the functions `lfuninit` and `lfunthetainit` respectively.
First example: Riemann zeta

\[ L = \text{lfuncreate}(1); \quad \text{``1'' = Riemann zeta function} \]
\[ \text{lfun}(L, 2) \]
\[ \text{lfunzeros}(L, 30) \]
\[ \text{\texttt{\textbackslash pb}} \ 32 \]
\[ L = \text{lfuninit}(L, \{1/2, 0, 30\}); \]
\[ \text{ploth}(t = 0, 30, \text{lfunhardy}(L, t)) \]

Generalization: Kronecker character. If \( D \) is a fundamental discriminant, then \texttt{lfuncreate}(D) is \( L((D/.), s) \).
Second example: Dedekind zeta

\begin{verbatim}
L = lfuncreate('x^3-2); \Q(2^(1/3))
lfun(L, 2)
lfunzeros(L,30)
\pb 32
L = lfuninit(L, [1/2, 0, 30]);
ploth(t = 0, 30, lfunhardy(L,t))
\end{verbatim}
Third example: Hasse-Weil zeta functions

E = ellinit([0,0,1,-7,6]);
L = lfuncreate(E); \\L(E,s)

lfun(L, 1)
lfun(E, 1)
lfun(L, 1, 1)\L'
lfun(L, 1, 2)\2nd derivative
lfun(L, 1, 3)\3rd derivative
ellanalyticrank(E)
lfunzeros(L,30)
\pb 32
Lbad = lfuninit(L, [1/2, 0, 30]); \\BUG !!!
ploth(t = 0, 30, lfunhardy(Lbad,t))
L = lfuninit(L, [1, 0, 30]); \Better
ploth(t = 0, 30, lfunhardy(L,t))
Dirichlet characters (1/3)

In PARI/GP, given a \textit{finite} abelian group

\[ G = (\mathbb{Z}/o_1\mathbb{Z})g_1 \oplus \cdots \oplus (\mathbb{Z}/o_d\mathbb{Z})g_d, \]

with fixed generators \( g_i \) of respective order \( o_i \), then

- the \textit{column} vector \([x_1, \ldots, x_d]\) represents the element \( g \cdot x := \sum_{i \leq d} x_i g_i \);
- the \textit{row} vector \([c_1, \ldots, c_d]\), represents the character mapping \( g_i \mapsto e(c_i / o_i) \) for each \( i \).

The group \( G \) is given by a GP structure, e.g. \texttt{bid}, \texttt{bnf}, \texttt{bnr}. We can choose \((g_i) := G.\text{gen}\) (SNF generators), hence \((o_i) = G.\text{cyc}\) and \( o_d \mid \cdots \mid o_1 \) (elementary divisors). But it is possible to choose other generators.
For Dirichlet characters modulo $q = \prod_p p^{e_p}$, there is another standard choice: Conrey generators (smallest primitive roots mod $p^{e_p}$). Conrey logarithm/exponential: map between

- elements in $(\mathbb{Z}/q\mathbb{Z})^*$: \texttt{znconreyexp},
- their discrete logs in terms of the Conrey generators: \texttt{znconreylog}, a column vector.

To such an element $m \in (\mathbb{Z}/q\mathbb{Z})^*$ we attach the Conrey character $\chi_q(m, \cdot)$.

See also \texttt{znconreychar} (in terms of SNF generators); so three possible representation of a character: one in terms of SNF generators and two (exp/log) in terms of Conrey generators.
Dirichlet characters (3/3)

G = idealstar(, 100);
G.cyc
chi = [2,0]; \ in terms of SNF gens.
m = znconreyexp(G, chi)
c = znconreylog(G, m)
s = ideallog(, m, G) znconreylog(G, chi)
znconreychar(G, m)
znconreychar(G, c) \ Bad input !
znconreychar(G, s) \ OK
Dirichlet $L$-function

N = 100; G = idealstar(, N); \((Z/100Z)^*\)
G.cyc
chi = [2, 0]
L = lfuncreate([G, chi]);

znconreyconductor(G, chi) \not primitive !
lfun(L, 1)
lfunlambdada(L, 1)
lfuntheta(L, 1)
N = znconreyconductor(G, chi, &chi0)
G0 = idealstar(,N);
Hecke $L$-function

K = bnfinit(x^3-7);
G = bnrinit(K, [11, [1]]);
G.cyc
chi = [1]
L = lfuncreate([G, chi]);

lfun(L, 0)
L = lfuninit(L, [1/2,1/2,30]);
lfun(L, 0)
lfun(L, 1)
lfunzeros(L,30)
ploth(t = 0, 30, lfunhardy(L,t))