CHAPTER 5

Euler Products and Euler Sums

The preceding chapters dealt with problems of a purely analytic nature. We now come to problems which are much more closely related to number theory. In the present section, we study the numerical computation of sums and products over the set P of prime numbers. Since this set is so complicated, this seems a priori hopeless, until one recalls the fundamental property of the Riemann zeta function for $\Re(s) > 1$:

$$\zeta(s) = \sum_{n\geq 1} \frac{1}{n^s} = \prod_{p\in P} \left(1 - \frac{1}{p^s}\right)^{-1} .$$

Thus if we can express our computation in terms of values of ζ we may be in good shape.

5.1. Euler Sums

First, let us show how to compute for $\Re(s) > 1$

$$S(s) = \sum_{p\in P} \frac{1}{p^s} .$$

The idea is the following: thanks to the above product formula for $\zeta(s)$, we have

$$\log(\zeta(s)) = -\sum_{p\in P} \log(1 - 1/p^s) = \sum_{p\in P} \sum_{m\geq 1} 1/(mp^{ms})$$
$$= \sum_{m\geq 1} \sum_{p\in P} 1/p^{ms} = \sum_{m\geq 1} S(ms)/m .$$

Now recall the *second* Möbius inversion formula: if $f(n) = \sum_{m\geq 1} g(mn)$, then $g(n) = \sum_{m\geq 1} \mu(m) f(mn)$. The proof of this is immediate:

$$\sum_{m\geq 1} \mu(m) f(mn) = \sum_{m\geq 1} \mu(m) \sum_{\ell\geq 1} g(\ell mn) = \sum_{N\geq 1} g(Nn) \sum_{m|N} \mu(m) = g(n)$$

by the basic property of the Möbius function.

By what we have seen above we have $\log(\zeta(ns))/n = \sum_{m\geq 1} S(mns)/(mn)$, so applying Möbius inversion to $f(n) = \log(\zeta(ns))/n$ and $g(m) = S(ms)/m$ we deduce that $S(ns) = n \sum_{m\geq 1} \mu(m) \log(\zeta(mns))/(mn)$, so that

$$S(s) = \sum_{m\geq 1} \frac{\mu(m)}{m} \log(\zeta(ms)) .$$

Since $z(M) = 1 + O(2^{-M})$ the convergence of this series is at least in $O(2^{-m})$, which is not bad. But we can do much better: it is clear that the proof of the

formula obtained above gives more generally

$$S_{>N}(s) = \sum_{m\geq 1} \frac{\mu(m)}{m} \log(\zeta_{>N}(ms)) \ ,$$

where $S_{>N}$ means that we restrict the sum to primes $p > N$, and $\zeta_{>N}$ that we take the Euler product for zeta restricted to primes $p > N$ (which is of course *not* the same as restricting the sum defining ζ to $n > N$). We have thus proved the following:

PROPOSITION 5.1.1. *Set $S(s) = \sum_{p\in P} p^{-s}$. For any $N > 0$ and s such that $\Re(s) > 1$ we have*

$$S(s) = \sum_{p\in P,\ p\leq N} \frac{1}{p^s} + \sum_{m\geq 1} \frac{\mu(m)}{m} \log(\zeta_{>N}(ms)) \ ,$$

where

$$\zeta_{>N}(s) = \zeta(s) \prod_{p\in P,\ p\leq N} (1 - 1/p^s) \ .$$

The whole point of introducing the parameter N is that $\log(\zeta_{>N}(ms)) = O(N^{-ms})$, so the series will converge much faster. Note, however, that N must not be chosen too large because in that case $\zeta_{>N}$ would be extremely close to 1 and there may be cancellation errors. In practice we advise choosing $10 < N < 100$ for instance.

Note in passing that the above proposition proves that $S(s)$ can be analytically continued to the half-plane $\Re(s) > 0$ with the rational points $1/k$ for $k \in \mathbb{Z}_{\geq 1}$ removed, where $S(s)$ has logarithmic singularities, and it is easy to show that it has a natural boundary $\Re(s) = 0$. To give a random example, we compute in the same way that

$$S(1/\sqrt{2}) = 0.335427961892072609411843548297775399 24 + i\pi \ .$$

For the following, we recall that the valuation of a power series is its order at $x = 0$:

COROLLARY 5.1.2. *Let $A(x) = \sum_{m\geq 1} a(m)x^m$ be a power series with no constant term with nonzero radius of convergence r and valuation v, and set $S(A; s) = \sum_{p\in P} A(1/p^s)$. Define*

$$c(n) = \sum_{d|n} \frac{\mu(d)}{d} a\left(\frac{n}{d}\right) \ .$$

Then for all $N \geq 1$ and $\Re(s) > \max(-\log(r)/\log(N), 1/v)$ we have

$$S(A; s) = \sum_{p\in P,\ p\leq N} A(1/p^s) + \sum_{n\geq 1} c(n) \log(\zeta_{>N}(ns)) \ .$$

In particular, if $a(1) = 0$ and $r > 1/2$ we have

$$S(A; 1) = \sum_{p\in P,\ p\leq N} A(1/p) + \sum_{n\geq 2} c(n) \log(\zeta_{>N}(n)) \ .$$

PROOF. Simply write $S_{>N}(A; s) = \sum_{m\geq 1} a(m)S_{>N}(ms)$ and use the proposition. The details are left to the reader. □

Since it is not immediate to estimate the speed of convergence in the general case, we restrict to the case of a rational function F. If we write $F = N/D$ with N and D coprime polynomials, the expansion at infinity of F is the expansion at 0 of $F(1/x) = N(1/x)/D(1/x)$, so the radius of convergence r is equal to the modulus of the zero of $D(1/x)$ closest to the origin, hence $1/r$ is equal to the modulus of the zero of $D(x)$ furthest from the origin, and this is approximately given by the command `polrootsbound` which gives an upper bound for this largest modulus, which we will use since it is considerably faster than computing the roots using `polroots`.

In addition, note that when computing $\log(\zeta_N(s))$ by the evident formula given above, there is considerable cancellation, so it is necessary to increase the working accuracy. All this is done in the following program:

SumEulerrat.gp

```
LogzetaN(s, N) =
{ my(B = getlocalbitprec());

  B += ceil(abs(real(s) - 1) * exponent(2*N));
  localbitprec(B);
  if (bitprecision(s) < B, s = bitprecision(s, B));
  log(zeta(s) * prodeuler(p = 2, N, 1 - p^(-s)));
}

Sdmob(ser, n) =
  sumdiv(n, d, moebius(d) * polcoeff(ser, n/d) / d);

/* Compute sum_{p >= a, prime} F(p^s), F rational function. */
SumEulerrat(F, s = 1, a = 2) =
{ my(B = getlocalbitprec(), vx = variable(F), rs, N, r, lim);
  my(FI, vF, sal, S);

  FI = subst(F, vx, 1/vx); vF = valuation(FI, vx);
  rs = real(s);
  if (rs <= 1 / vF, error("real(s)  <=  1/v"));
  localbitprec(32);
  r = 1 / max(polrootsbound(denominator(F)), 1);
  N = ceil(max(30, a+1));
  if (rs <= -log(r)/log(N), error("real(s) too small"));
  lim = ceil(B * log(2) / log(N^rs * r)) + 1;
  localbitprec(B);
  sal = bitprecision(1., B + 32)*FI + O(vx^(lim+1));
  bernvec(floor((lim * s + 1) / 2)); /* cache the B_{2n} */
  S = sum(n = vF, lim,
          my(m = Sdmob(sal, n)); if (m, m * LogzetaN(n*s, N)));
  forprime (p = a, N, S += subst(F, vx, p^s));
  return (bitprecision(S, B));
}
```

Examples (all essentially instantaneous):

```
? SumEulerrat(1 / x^2)
% = 0.45224742004106549850654336483224793417
? SumEulerrat(1 / (x^2 + 1))
% = 0.38905955531696837171041438969249635814
? /* SumEulerrat(1 / (x^(3/2) + 1)) would be illegal, so: */
? SumEulerrat(1 / (x^3 + 1), 1/2)
% = 0.71426973554924155313504391134630993732
```

5.2. Euler Products

A completely similar method can be used to compute Euler *products* numerically. The result is as follows:

COROLLARY 5.2.1. *Let $B(x) = 1 + \sum_{m\geq 1} b(m)x^m$ be a power series with constant term 1 with nonzero radius of convergence r, and assume that $B(z) \neq 0$ for $|z| < r$. Let v be the valuation of $B(x) - 1$ and set $P(B; s) = \prod_{p\in P} B(1/p^s)$. Write $\log(B(x)) = \sum_{m\geq 1} a(m)x^m$ and define as above*

$$c(n) = \sum_{d|n} \frac{\mu(d)}{d} a\left(\frac{n}{d}\right) .$$

Then for all $N \geq 1$ and $\Re(s) > \max(-\log(r)/\log(N), 1/v)$ we have

$$P(B; s) = \prod_{p\in P,\ p\leq N} B(1/p^s) \prod_{n\geq 1} \zeta_{>N}(ns)^{c(n)} .$$

In particular, if $b(1) = 0$ and $r > 1/2$ we have

$$P(B; 1) = \prod_{p\in P,\ p\leq N} B(1/p) \prod_{n\geq 2} \zeta_{>N}(n)^{c(n)} .$$

Furthermore $c(n)$ satisfies the recurrence

$$c(n) = b(n) - \frac{1}{n} \sum_{1\leq k\leq n-1} kc(k) \sum_{1\leq q\leq n/k} b(n - qk) .$$

PROOF. The first part of the corollary is immediate by applying the preceding corollary to $A(x) = \log(B(x))$, which has the same valuation as $B(x) - 1$, and at least radius of convergence r since we assume that B has no zeros in $|z| < r$. For the recurrence, we note that

$$B'(x) = \sum_{m\geq 1} mb(m)x^{m-1} = B(x)\log(B(x))' = B(x) \sum_{m\geq 1} ma(m)x^{m-1} ,$$

so the recurrence follows by identification of coefficients and the formula $\sum_{d|n} dc(d) = na(n)$ which defines $c(n)$. □

REMARKS 5.2.2. (1) The coefficients $c(n)$ are the unique exponents such that we have the formal expansion

$$1 + \sum_{m\geq 1} b(n)x^m = \prod_{n\geq 1} (1 - x^n)^{-c(n)} .$$

(2) It is usually preferable to use the formula giving $c(n)$ in terms of $a(n)$. However it may happen that $a(n)$ is not easy to compute directly, and one can then use the recurrence for $c(n)$.

A possible program is as follows, essentially copied from the preceding one:

ProdEulerrat.gp

```
/* Compute prod_{p >= a, prime} F(p^s), F rational function. */
ProdEulerrat(F, s = 1, a = 2) =
{ my(B = getlocalbitprec(), vx = variable(F));
  my(FIm1, rs, N, r, lim, vF, sal, S);

  FIm1 = subst(F, vx, 1/vx) - 1;
  vF = valuation(FIm1, vx);
  rs = real(s);
  if (rs <= 1/vF, error("real(s) <= 1/v"));
  localbitprec(32);
  r = 1 / max(polrootsbound(numerator(F)),
              polrootsbound(denominator(F)));
  N = ceil(max(30, a+1));
  if (rs <= -log(r)/log(N), error("real(s) too small"));
  lim = ceil(B * log(2) / log(N^rs * r)) + 1;
  localbitprec(B);
  sal = log(1 + bitprecision(1., B + 32)*FIm1 + O(vx^(lim+1)));
  bernvec(floor((lim * s + 1) / 2)); /* cache the B_{2n} */
  S = sum(n = vF, lim,
          my(m = Sdmob(sal,n)); if (m, m * LogzetaN(n*s, N)));
  S = exp(S);
  forprime (p = a, N, S *= subst(F, vx, p^s));
  return (bitprecision(S, B));
}
```

Examples (again essentially instantaneous):

```
? ProdEulerrat(1 + 1 / x^3) - zeta(3) / zeta(6)
% = 0.E-38
? ProdEulerrat(1 - 1 / (x - 1)^2, 1, 3) /* Twin prime constant */
% = 0.660161815846869573927812110014555778 43
```

Note that both of these programs are already implemented in `Pari/GP` under the respective names `sumeulerrat` and `prodeulerrat`.

5.3. Variants Involving $\log(p)$ or $\log(\log(p))$

A large number of similar sums or products over primes can be computed in an analogous manner. A simple example is $\sum_{p\in P} \log(p)/p^s$ for $\Re(s)$, for which we can modify Proposition 5.1.1 essentially by replacing $\log(\zeta(s)) = -\sum_{p\in P} \log(1 - p^{-s})$ by its derivative $\zeta'(s)/\zeta(s) = -\sum_{p\in P} \log(p)p^{-s}/(1 - p^{-s})$. In this way we immediately compute for instance that

$$\sum_{p\in P} \frac{\log(p)}{p^2} = 0.49309110936876446219782 62\cdots$$

We can also compute *limits* such as

$$\lim_{s\to 1^+}\left(\sum_{p\in P}\frac{1}{p^s}-\log(\zeta(s))\right)=-0.31571845205389007685\cdots,$$

simply by suppressing the term $m=1$ in the formula expressing $S(s)$ in terms of $\log(\zeta(ms))$, and

$$\lim_{x\to\infty}\left(\sum_{\substack{p\in P\\ p\le x}}\frac{1}{p}-\log(\log(x))\right)=0.261497212847642783755\cdots,$$

using the easily proved formula

$$\lim_{x\to\infty}\left(\sum_{\substack{p\in P\\ p\le x}}\frac{1}{p}-\log(\log(x))\right)=\gamma+\lim_{s\to 1^+}\left(\sum_{p\in P}\frac{1}{p^s}-\log(\zeta(s))\right),$$

where as usual γ is Euler's constant.

A little more challenging is the computation of sums over primes p when $\log(p)$ is in the *denominator* of the summand, typically sums like $T(s)=\sum_{p\in P}1/(p^s\log(p))$ for $\Re(s)\ge 1$. When $\log(p)$ is on the numerator, we have to use the derivative of $\log(\zeta(s))$. Analogously, here we have to use the *integral* of $\log(\zeta(s))$, in other words

$$\int_s^\infty \log(\zeta(t))\,dt\,.$$

Even though there are faster methods for computing this, for simplicity we use the doubly-exponential method, and to compute say, the 30 values of this integral for integer $s\in[1,30]$ to 38 decimals requires only 0.81 seconds. The following simple-minded program computes $T(s)$, using the formula given in the proposition below:

SumEulerlog.gp

```
/* Compute sum_{p prime} 1/(p^s log(p)). */
SumEulerlog(s, N = max(2, 30/abs(s))) =
{ my(B = getlocalbitprec(), S = 0, LN, T, lim);

  localbitprec(32); LN = log(N);
  lim = ceil(B*log(2)/LN);
  localbitprec(B + 32);
  forprime(p = 2, N, S += 1 / (p^s * log(p)));
  LN = bitprecision(LN, B + 32);
  T = intnuminit(0,[oo, 1]);
  forsquarefree(K = 1, lim,
    my([k] = K, m = moebius(K) / k, a = 1 / (k * LN));
    /* Tk = intnuminit(0, [oo, 1/a]) */
    my(Tk = vector(#T, i, if (i == 1, T[1], T[i] * a)));
    S += m * intnum(t = 0, oo, LogzetaN(k * (t + s) , N), Tk));
  return (S);
}
```

In a few seconds we obtain for instance

$$\sum_{p\in P}\frac{1}{p\log(p)} = 1.6366163233512608685696 58\cdots$$

$$\sum_{p\in P}\frac{1}{p^2\log(p)} = 0.5077821878591993187743 75\cdots$$

Note that the series $\sum 1/(p\log(p))$ barely converges (and as usual is over the irregular set of primes), so it is remarkable that there is no problem in computing its sum.

We leave as an exercise for the reader the task of writing a program to compute $\sum_{p\in P} 1/(p^a\log^b(p))$ with $a>1$ or $a=1$ and $b\geq 1$, arbitrary a, and arbitrary integer b.

An even harder problem is when $\log(\log(p))$ (or worse) occurs. This can also be dealt with, but is more technical. The idea is as follows: recall that we have a linear formula expressing $S(s)=\sum_{p\in P}1/p^s$ in terms of $f(s)=\log(\zeta(s))$. By derivation, it follows that we also have a formula for $S^{(k)}(s)=(-1)^k\sum_{p\in P}(\log(p))^k/p^s$ in terms of $f^{(k)}(s)$. For instance this is how we computed $\sum_{p\in P}\log(p)/p^2$. We now would like to differentiate *with respect to* k, since the derivative with respect to k of $\log(p)^k$ is $\log(\log(p))\log(p)^k$, which is exactly what is needed after setting $k=0$. A priori this does not seem realistic, but in fact is quite easy:

PROPOSITION 5.3.1. (1) *For* $\Re(s)>0$ *and* $\Re(x)\geq 1$ *we have*

$$\sum_{p\in P}\frac{1}{p^x\log(p)^s} = \frac{1}{\Gamma(s)}\sum_{k\geq 1}\frac{\mu(k)}{k}\int_0^\infty t^{s-1}\log(\zeta(k(t+x)))\,dt\ .$$

(2) *For* $\Re(s)>-1$ *and* $\Re(x)\geq 1$ *we have*

$$\sum_{p\in P}\frac{1}{p^x\log(p)^s} = -\frac{1}{\Gamma(s+1)}\sum_{k\geq 1}\mu(k)\int_0^\infty t^s\frac{\zeta'}{\zeta}(k(t+x))\,dt\ .$$

PROOF. (Sketch.) For (1) it is easy to check the absolute convergence of the series and of the integral. We then proceeds as above for the evaluation of $S(s)$: we express $\log(\zeta)$ as a sum over primes thanks to the Euler product and use Möbius inversion. For (2), change s into $s+1$ and take the derivative with respect to x. The details of these proofs are left to the reader. □

Using this proposition, one can then compute the Taylor series expansion of the left hand side around $s=0$ for instance, and obtain hundreds of decimals of expressions such as

$$\sum_{p\in P}\frac{\log(\log(p))}{p\log(p)}$$

and

$$\lim_{x\to\infty}\left(\sum_{\substack{p\in P\\ p\leq x}}\frac{\log(\log(p))^n}{p}-\frac{\log(\log(x))^{n+1}}{n+1}\right).$$

All these computations are left as exercises for the reader. For instance, we can write the following program, which is a simple modification of the previous one:

SumEulerloglog.gp

```
/* Compute sum_{p prime} log(log(p))/(p^s log(p)). */
SumEulerloglog(s, N = max(2, 30/abs(s))) =
{ my(B = getlocalbitprec(), S = 0, LN, T, lim, E);

  localbitprec(32); LN = log(N);
  lim = ceil(B*log(2)/LN);
  localbitprec(B + 32);
  forprime(p = 2, N, S += log(log(p)) / (p^s * log(p)));
  LN = bitprecision(LN, B + 32);
  T = intnuminit(0,[oo, 1]);
  E = Euler();
  forsquarefree(K = 1, lim,
    my([k] = K, m = moebius(K) / k, a = 1 / (k * LN));
    /* Tk = intnuminit(0, [oo, 1/a]) */
    my(Tk = vector(#T, i, if (i == 1, T[1], T[i] * a)));
    S -= m * intnum(t = 0, oo, (log(t) + E)
                               * LogzetaN(k * (t+s) , N), Tk));
  return (S);
}
```

In a few seconds we compute that

$$\sum_{p\in P}\frac{\log(\log(p))}{p\log(p)} = 0.64108021565998466048335188915139995189134 51\cdots$$

Once again we leave as an exercise for the reader the task of writing a program to compute $\sum_{p\in P}\log^k(\log(p))/(p^a\log^b(p))$ with $a>1$ or $a=1$ nd $b\geq 1$, arbitrary a, arbitrary integer b, and arbitrary nonnegative integer k.

5.4. Variants Involving Quadratic Characters

We now would like to compute prime sums and Euler products which, in addition to involving regular functions over primes, also involve a quadratic character $\left(\frac{D}{n}\right)$ where D is a not necessarily fundamental discriminant.

A natural idea is to introduce, in addition to the Riemann ζ function, the Dirichlet L-functions $L(\chi,s)$ for Dirichlet characters χ, in particular for $\chi(n)=\left(\frac{D}{n}\right)$.

We can easily generalize the results of the previous sections to this case. If for any character χ we set

$$S(\chi,s)=\sum_p\frac{\chi(p)}{p^s}$$

then, exactly as before and with evident notation Möbius inversion gives

$$S(\chi,s)=\sum_{p\in P,\ p\leq N}\frac{\chi(p)}{p^s}+\sum_{m\geq 1}\frac{\mu(m)}{m}\log(L_{>N}(\chi^m,ms))\ .$$

We will briefly study below the computation of $L_{>N}(\chi^m,ms)$.

Let us specialize to the case where χ is a quadratic character $\left(\frac{D}{n}\right)$ for some (not necessarily fundamental) discriminant D. Then $\chi^k=\chi$ if k is odd and $\chi^k=\chi_0$,

the trivial character modulo D ($\chi_0(n) = 1$ if $(n, D) = 1$ and $\chi_0(n) = 0$ otherwise) if $k \geq 2$ is even. Thus,

$$S(\chi, s) = \sum_{\chi(p)=1} \frac{1}{p^s} - \sum_{\chi(p)=-1} \frac{1}{p^s} ,$$

hence

$$\sum_{\chi(p)=1} \frac{1}{p^s} = \frac{1}{2}(S(\chi_0, s) + S(\chi, s)) \quad \text{and}$$
$$\sum_{\chi(p)=-1} \frac{1}{p^s} = \frac{1}{2}(S(\chi_0, s) - S(\chi, s)) ,$$

where

$$S(\chi_0, s) = S(s) - \sum_{p|D} \frac{1}{p^s}$$

with the notation of the previous section.

For $j = -1$, 0, 1 corresponding to the three possible values of χ, let $A_j(x) = \sum_{m\geq 1} a_j(m)x^m$ be power series satisfying the same assumptions as in Corollary 5.1.2. We set

$$S(A; s) = \sum_{j\in\{-1,0,1\}} \sum_{p\in P,\ \chi(p)=j} A_j(1/p^s) .$$

A generalization of Corollary 5.1.2, which is an immediate consequence of the above formulas is as follows:

PROPOSITION 5.4.1. *Define*

$$c_j(n) = \sum_{d|n,\ 2\nmid d} \frac{\mu(d)}{d} a_j\left(\frac{n}{d}\right) .$$

Then for all $N \geq 1$ and $\Re(s) > \max(-\log(r)/\log(N), 1/v)$ we have

$$\begin{aligned} S(A; s) = &\sum_{j\in\{-1,0,1\}} \sum_{p\in P,\ p\leq N\ \chi(p)=j} A_j(1/p^s) \\ &+ \frac{1}{2}\sum_{n\geq 1}(c_1(n) - c_{-1}(n)) \log(L_{>N}(\chi, ns)) \\ &+ \frac{1}{2}\sum_{n\geq 1}(c_1(n) + c_{-1}(n) - c_1(n/2)) \log(\zeta_{p\nmid D,\ >N}(ns)) \\ &+ \sum_{p|D,\ p>N} A_0(1/p^s) , \end{aligned}$$

with the usual convention that $c_1(x) = 0$ if $x \notin \mathbb{Z}$ and where we set

$$\zeta_{p\nmid D,\ >N}(s) = L_{>N}(\chi_0, s) = \zeta_{>N}(s) \prod_{p|D,\ p>N} \left(1 - \frac{1}{p^s}\right) .$$

It is clear that such formulas can be generalized to Dirichlet characters χ of higher order r, and the result involves in a simple manner the functions $L(\chi^b, N)$ for $0 \leq b < r$. The quadratic case is especially simple since it involves $L(\chi, N)$ and $L(\chi^0, N) = L(\chi_0, N)$ which is essentially the Riemann ζ function.

We must also explain how to compute the quantities $L_{>N}(\chi, ns)$, or equivalently $L(\chi, ns)$, at least when $\chi = \left(\frac{D}{\cdot}\right)$. When $|D|$ is very small (for instance $D = -3$ or $D = -4$), we can use the variant of the `sumalt` algorithm explained to us by B. Allombert and described in Section 4.5.3.

When $\Re(ns)$ is large and we do not need too much accuracy, we can also directly use the definition (either as a sum or as an Euler product). Otherwise, we can trivially reduce to the case of a fundamental discriminant D, and we can use one of several methods for computing $L(\chi, s)$ such as the one described in Chapter 9, which in `GP` is simply `lfun(D,n*s)` (note that the `lfun` command can be used with general Dirichlet characters, not only quadratic, and vastly more general types of L-functions; however, it does require the existence of a functional equation of standard type, which explains the necessity in our situation of reducing to fundamental discriminants).

5.5. Variants Involving Congruences

Let a and k be coprime positive integers. Instead of considering Euler sums or products over all primes, we can consider the same but restricted to primes congruent to a modulo k. This is easily done as follows. For all a coprime to k and s with $\Re(s) > 1$ we define

$$S_a(s) = \sum_{p \in P,\ p \equiv a \pmod k} \frac{1}{p^s} \quad \text{and} \quad T(\chi, s) = \sum_{a \bmod k} \chi(a) S_a(s) \ .$$

For any character χ modulo k we have

$$\begin{aligned} \log(L(\chi, s)) &= \sum_{p \in P} -\log\left(1 - \frac{\chi(p)}{p^s}\right) = \sum_{m \geq 1} \frac{1}{m} \sum_{p \in P} \frac{\chi^m(p)}{p^{ms}} \\ &= \sum_{m \geq 1} \frac{1}{m} \sum_{a \bmod k} \chi^m(a) S_a(ms) = \sum_{m \geq 1} \frac{T(\chi^m, ms)}{m} \ . \end{aligned}$$

It follows that for any integer $n \geq 1$

$$\frac{\log(L(\chi^n, ns))}{n} = \sum_{m \geq 1} \frac{T(\chi^{mn}, mns)}{mn} \ ,$$

so by the second Möbius inversion formula we have in particular

$$T(\chi, s) = \sum_{a \bmod k} \chi(a) S_a(s) = \sum_{m \geq 1} \frac{\mu(m)}{m} \log(L(\chi^m, ms)) \ .$$

Now by orthogonality of characters, if we denote by X_k the group of characters of $(\mathbb{Z}/k\mathbb{Z})^*$ we have for b coprime to k

$$\sum_{\chi \in X_k} \overline{\chi}(b) T(\chi, s) = \sum_{a \bmod k} S_a(s) \sum_{\chi \in X_k} \chi(ab^{-1}) = \phi(k) S_b(s) \ ,$$

hence

$$S_a(s) = \frac{1}{\phi(k)} \sum_{\chi \in X_k} \overline{\chi}(a) \sum_{m \geq 1} \frac{\mu(m)}{m} \log(L(\chi^m, ms)) \ .$$

As above, it is then immediate to compute any convergent Euler sum of the form $\sum_{p \equiv a \pmod k} f(p)$ or any convergent Euler product of the form $\prod_{p \equiv a \pmod k} f(p)$.

As mentioned above, to compute $L(\chi^n, ns)$ (hence $L_{>N}(\chi^n, ns)$) the simplest is to use the preprogrammed `Pari/GP lfun` program.

As an application, if we set $P_a = \prod_{p\equiv a \pmod 5} 1/(1-1/p^2)$, we immediately compute that

$$P_1 = 1.010915160601019522604956584289514920984538627581738523 7\cdots$$
$$P_2 = 1.368572053876649085860763890483109990170207828885895205 0\cdots$$
$$P_3 = 1.135764878668921626868643009472082289511936413005468744 1\cdots$$
$$P_4 = 1.004960323922297558993749624810252184795510294188022880 1\cdots.$$

Of course $P_1P_2P_3P_4 = (1-1/5^2)\zeta(2) = 4\pi^2/25$.

The product P_2P_3 was computed in [**ERS19**] using a different and more complicated method which apparently does not allow the computation of P_2 and P_3 individually, however see [**Ram19**].

5.6. Hardy–Littlewood Constants: Quadratic Polynomials

Let $A(X) \in \mathbb{Z}[X]$ be a polynomial with integer coefficients and assume that $A(X)$ is irreducible in $\mathbb{Q}[X]$ and has content 1. For any prime p, we let $\omega(p) = \omega(A,p)$ be the number of solutions in $\mathbb{F}_p$ of $A(x) = 0$. Then conjecturally, the number of integers n with $1 \le n \le N$ such that $|A(n)|$ is prime should be asymptotic to

$$\frac{H(A)}{\deg(A)} \cdot \frac{N}{\log N}, \quad \text{where} \quad H(A) = \prod_p \frac{p-\omega(p)}{p-1}$$

is the so-called *Hardy–Littlewood constant* of the polynomial A. It is therefore interesting to compute this Euler product. In particular, polynomials with a large $H(A)$ should have asymptotically more prime values.

The case of linear polynomials being trivial ($H(aX+b) = a/\phi(a)$ when $\gcd(a,b) = 1$), in the present section we first treat the next simplest case of quadratic polynomials, and consider polynomials of larger degree later. Thus, let $A(X) = aX^2+bX+c$ be an irreducible quadratic polynomial with $\gcd(a,b,c) = 1$, and let $D = b^2-4ac$ be its discriminant. It is easy to see that $\omega(p)$ is given by the following formulas.

- If $p \nmid a$, then $\omega(p) = 1 + \left(\frac{D}{p}\right)$.
- If $p \mid a$ and $p \nmid b$, then $\omega(p) = 1$.
- If $p \mid a$ and $p \mid b$ (hence $p \nmid c$), then $\omega(p) = 0$.

A little computation left to the reader shows the following:

PROPOSITION 5.6.1. *Let $A(X) = aX^2+bX+c$ be an irreducible polynomial of degree 2 with $\gcd(a,b,c) = 1$. The Hardy–Littlewood constant of A is given by the formula*

$$H(A) = c_2 \prod_{p>2} \left(1 - \frac{\left(\frac{D}{p}\right)}{p-1}\right) \prod_{p\mid a,\ p\nmid 2b} \frac{p-1}{p-2} \prod_{p\mid \gcd(a,b),\ p>2} \frac{p}{p-1},$$

where $c_2 = 0$ if c is even and $a+b$ is even, $c_2 = 1$ if $a+b+c$ is odd, and $c_2 = 2$ if c is odd and $a+b$ is even.

Write $D = D_0 f^2$ with D_0 a fundamental discriminant. Since

$$\prod_{p>2}\left(1-\frac{\left(\frac{D}{p}\right)}{p-1}\right) = \prod_{p>2}\left(1-\frac{\left(\frac{D_0}{p}\right)}{p-1}\right)\prod_{p\mid D,\ p>2}\left(1-\frac{\left(\frac{D_0}{p}\right)}{p-1}\right)^{-1},$$

we must therefore compute the Euler product

$$C(D_0) = \prod_{p>2}\left(1-\frac{\left(\frac{D_0}{p}\right)}{p-1}\right)$$

for a fundamental discriminant D_0. With a slight abuse of notation, set $C(D_0,s) = \prod_{p>2}(1-\left(\frac{D_0}{p}\right)/(p^s-1))$, so that $C(D_0) = C(D_0,1)$, and set $\chi(n) = \left(\frac{D_0}{n}\right)$. Then for $\Re(s) > 1$

$$-\log(C(D_0,s)) = -\sum_{p,\ \chi(p)=1}\log\frac{p^s-2}{p^s-1} - \sum_{p,\ \chi(p)=-1}\log\frac{p^s}{p^s-1},$$

hence with the notation of Section 5.4, $C(D_0,s) = e^{-S(A;s)}$ with $A_0(1/p^s) = 0$,

$$A_1(1/p^s) = -\log\left(\frac{p^s-2}{p^s-1}\right) = \sum_{m\geq 1}\frac{2^m-1}{mp^{ms}},$$

$$A_{-1}(1/p^s) = -\log\left(\frac{p^s}{p^s-1}\right) = -\sum_{m\geq 1}\frac{1}{mp^{m}s},$$

so that $a_1(m) = (2^m-1)/m$ and $a_{-1}(m) = -1/m$. Thus,

$$c_1(n) = \frac{1}{n}\sum_{d\mid n,\ 2\nmid d}\mu(d)(2^{n/d}-1)$$

and

$$c_{-1}(n) = -\frac{1}{n}\sum_{d\mid n,\ 2\nmid n}\mu(d).$$

Set $a(n) = (c_1(n)-c_{-1}(n))/2$ and $b(n) = (c_1(n)+c_{-1}(n)-c_1(n/2))/2$, so that

$$S(A;s) = \sum_{n\geq 1}a(n)\log(L(\chi,ns)) + \sum_{n\geq 1}b(n)\log(\zeta_{p\nmid D_0}(ns)).$$

Then by the above we have

$$a(n) = \frac{1}{2n}\sum_{d\mid n,\ 2\nmid d}\mu(d)2^{n/d}$$

and one easily checks that $b(n) = a(n)-a(n/2)$ if $n > 1$, while $b(1) = 0$ (recall that we set $a(x) = 0$ if x is not integral), so we can make $s \to 1$ by limiting the second sum to $n \geq 2$. We can thus compute $C(D_0)$, using as above $L_{p>N}(\chi,n)$ and $\zeta_{p\nmid D_0,\ p>N}(n)$. The following program implements this:

HardyLittlewood2.gp

```
/* Auxiliary functions. */
ZetaDN(P, s) = zeta(s) * prod(j = 1, #P, 1 - P[j]^(-s));

LchiN(L, Ebad, s) =
{ my([P, E] = Ebad);
```

```
  lfun(L, s) * prod(j = 1, #P, subst(E[j], 'x, P[j]^(-s)));
}

LchiNinit(D, P) =
{ my(Ebad = [], Pbad = []);
  foreach (P, p,
    my(s = kronecker(D, p));
    if (s, Ebad = concat(Ebad, 1 - s*'x);
           Pbad = concat(Pbad, p)));
  return ([Pbad, Ebad]);
}

Oddpart(n) = n >> valuation(n,2);

/* The real work; D is fundamental. */
HLW2(D, N) =
{ my(B = getlocalbitprec(), lim, S1, S2, L, P, v, Ebad);

  localbitprec(32); lim = ceil(B*log(2)/log(N/2));
  localbitprec(B + lim + exponent(lim));
  L = lfuninit(D, [1/2, lim, 0]);
  v = vector(lim);
  forfactored(X = 1, lim,
    my([n, fan] = X, S = 0, P = fan[,1]);
    if (n % 2 == 0, P = P[^1]);
    X = matconcat([P, vectorv(#P,i,1)]);
    fordivfactored(X, Y,
      my([d] = Y); \\ odd squarefree divisor of X
      S += moebius(Y) << (n/d));
    v[n] = S / (2*n);
  );
  P = setunion(factor(abs(D))[,1]~, primes([2, N]));
  Ebad = LchiNinit(D, P);
  S1 = sum(n = 1, lim, v[n] * log(LchiN(L, Ebad, n)));
  S2 = sum(n = 2, lim, (v[n] - if (n%2 == 0, v[n/2]))
                       * log(ZetaDN(P, n)));
  return (S1 + S2);
}

/* Compute the Hardy-Littlewood constant of aX^2+bX+c. */
HardyLittlewood2(A, N = 50) =
{ my(D = poldisc(A), S, P);

  if (poldegree(A) != 2, error("polynomial of degree != 2"));
  my([a, b, c] = Vec(A));
  if (issquare(D) || gcd([2 * a, a + b, c]) > 1, return (0));
  N = max(N, 3);
  /* Take care of the prime p = 2. */
```

```
54    S = if ((a + b) % 2, 1., 2.);
55    /* Take care of odd primes dividing a. */
56    P = factor(Oddpart(a))[,1];
57    foreach (P, p,
58      S *= if (b % p, (p - 1) / (p - 2), p / (p - 1))
59    );
60    /* Take care of odd primes dividing the index f. */
61    my([D0, f] = coredisc(D, 1));
62    P = factor(Oddpart(f))[,1];
63    S /= vecprod([1 - kronecker(D0, p) / (p - 1) | p <- P]);
64    /* Take care of the primes p <= N. */
65    S *= prodeuler(p = 3, N, 1 - kronecker(D0, p) / (p - 1));
66    /* Do the real work */
67    return (S * exp(-HLW2(D0, N)));
68  }
```

For example, using this program we compute that

$$H(X^2+X+41) = 6.6395463549428433306471137152997759329371091\ldots$$
$$H(X^2+X+75) = 0.6219533598519743400087125748592568290582438\ldots$$
$$H(2X^2-199) = 7.3291180993696071658232761749275362031861488\ldots\,.$$

Thus, we can reasonably expect that the first and third polynomials will produce many primes (and in some sense are record-breaking), while the second will produce much fewer primes (but it is easy to find polynomials of the same shape which produce even less).

Two remarks:

REMARKS 5.6.2.
(1) The use of the preliminary `lfuninit` command in the `HLW2` program avoids recomputing some data for each value of n, and speeds up the program considerably.
(2) Note the use of the `forfactored` and `fordivfactored` commands, which avoid factoring several times the same integers (refer to the `Pari/GP` manual for the use of these functions).
(3) The bit accuracy `B` needs to be increased by two increments: first a quantity `lim` which measures the difference between relative and absolute accuracy (`GP` works with relative, but we need absolute), and second a quantity `exponent(lim)` which measures approximately the number of bits of accuracy lost when summing `lim` terms. In other scripts this additional quantity was often arbitrarily set to 5, 10, or more often 32, but here it is essential to optimize since the speed of the `lfuninit` command is extremely sensitive to the bit accuracy.

As an amusing exercise, we can search for quadratic polynomials whose Hardy–Littlewood constant is large, and in view of the above examples, larger than 6, say. We may of course assume that $a > 0$, and since changing X into $X+1$ changes b into $b+2a$ and changing X into $-X$ changes b into $-b$ we may assume that $0 \le b \le a$. This of course only excludes the most basic polynomial transformations. In addition, since we are looking for *large* $H(A)$, we may assume that c is odd (otherwise all even x are excluded), hence that $a+b$ is even (otherwise all odd x

are excluded), and this exactly corresponds to the case $c_2 = 2$ in the formula for $H(A)$. We can thus write the following:

HardyLittlewoodsearch.gp

```
HardyLittlewoodsearch(lima, limc) =
{ my(V = List());

  localbitprec(64);
  if (limc % 2 == 0, limc++);
  for (a = 1, lima,
    forstep (b = a%2, a, 2,
      my(g = gcd(a, b), P = a * 'x^2 + b * 'x);
      forstep (c = -limc, limc, 2,
        if (gcd(c, g) > 1, next());
        my(r = HardyLittlewood2(P + c, 50));
        if (r > 6.5, listput(V, [[a,b,c], r]); print(V[#V]))
      )
    )
  ); \\ sort with respect to 'r'
  return (vecsort(Vec(V), 2, 12));
}
```

Note that the flag 12 in the `vecsort` command means first that we want to sort by descending instead of ascending order, and second that we want to remove duplicate entries.

Running this program with `lima=12` and `limc=1000`, we see that in that range the largest $H(A)$ is indeed obtained for $A(X) = 2X^2 - 199$ (and a few equivalent polynomials) with $H(A) = 7.3291\cdots$ as given above, followed by $A(X) = 6X^2 + 6X + 31$ (and equivalent) with $H(A) = 6.9208\cdots$.

On the other hand, by making a less systematic search using a truncated form of the formula for $H(A)$, it is easy to find quadratic polynomals with $H(A) > 8$, for instance $H(37X^2 + 23X - 8863) = 8.097818\cdots$. In any case, it is easy to show that $H(A)$ is unbounded.

5.7. Hardy–Littlewood Constants: General Polynomials

5.7.1. Cubic Polynomials. Let $A(X) = aX^3 + bX^2 + cX + d$ be a cubic polynomial. In order for A to represent infinitely many primes it is evidently necessary that A be irreducible and that $\gcd(a, b, c, d) = 1$. But writing

$$A(X) = 6a(X(X-1)(X-2)/6) + (6a+2b)(X(X-1)/2) + (a+b+c)X + d$$

we see that a stronger condition is $\gcd(6a, 2b, a+b+c, d) = 1$ (similar to the condition $\gcd(2a, a+b, c) = 1$ used in the previous script). Let D be the discriminant of the polynomial A. The primes p dividing a or D are finite in number, so for such primes $\omega(p)$ can be computed directly (we will see below the `GP` commands). Thus, let p be a prime not dividing a or D, and let K be the cubic number field defined by the polynomial A. Since $p \nmid aD$ the decomposition of the polynomial A modulo p reflects the decomposition of the prime p in the extension $K/\mathbb{Q}$, hence $\omega(p)$ is equal to the number of prime ideals of K above p which are of residual degree 1.

In other words, with evident notation, if p splits as 3, 21, or 111 then $\omega(p) = 0$, 1, or 3 respectively. Thus again with evident notation we have

$$H(A) = \prod_{p|aD} \frac{p-\omega(p)}{p-1}(P_3P_{111})(1) \quad \text{with}$$

$$P_3(s) = \prod_{\substack{p\nmid aD \\ p\mathbb{Z}_K=\mathfrak{p}_3}} \frac{p^s}{p^s-1} \quad \text{and} \quad P_{111}(s) = \prod_{\substack{p\nmid aD \\ p\mathbb{Z}_K=\mathfrak{p}_1\mathfrak{p}_1'\mathfrak{p}_1''}} \frac{p^s-3}{p^s-1}\,.$$

Thus $\log(P_3(s)) = \sum_{k\geq 1} S_3(ks)/k$ and $\log(P_{111}(s)) = -\sum_{k\geq 1}(3^k-1)S_{111}(ks)/k$ with $S_3(s) = \sum_{p\nmid aD,\ p\mathbb{Z}_K=\mathfrak{p}_3} 1/p^s$ and similarly for $S_{111}(s)$.

We must now distinguish the Galois type of the number field K. Assume first that K is a cyclic cubic field, i.e., that D is a square. This case is essentially identical with the quadratic case, with 2 replaced by 3 (this is true more generally for C_ℓ-fields with ℓ prime). More precisely, we have

$$S_3(s) = \frac{1}{3}\sum_{3\nmid n} \frac{\mu(n)}{n}\log((\zeta^3/\zeta_K)_{p\nmid aD}(ns))$$

$$S_{111}(s) = \sum_{n\geq 1}\frac{\mu(n)}{n}\log(\zeta_{p\nmid aD}(ns)) - S_3(s)\,.$$

Thus, if we set

$$c_1(n) = \frac{1}{n}\sum_{d|n}\mu(d)(3^{n/d}-1) \quad \text{and} \quad c_2(n) = \frac{1}{3n}\sum_{d|n,\ 3\nmid d}\mu(d)3^{n/d}$$

we have

$$\begin{aligned}\log((P_3P_{111})(s)) &= -\sum_{n\geq 1}(c_1(n)\log(\zeta_{p\nmid aD}(ns)) + c_2(n)\log((\zeta^3/\zeta_K)_{p\nmid aD}(ns))) \\ &= \sum_{n\geq 1} a(n)\log((\zeta_K/\zeta)_{p\nmid aD}(ns)) + \sum_{n\geq 1} c(n)\log(\zeta_{p\nmid aD}(ns))\end{aligned}$$

with $a(n) = -c_2(n)$ and $c(n) = 2c_2(n) - c_1(n)$. Since $c_1(1) = 2$ and $c_2(1) = 1$, we have $c(1) = 0$ (otherwise our Euler product would not converge), and for $n \geq 2$ we have $c_1(n) = (1/n)\sum_{d|n}\mu(d)3^{n/d}$. An easy computation shows that $c_1(n) = 3c_2(n) - c_2(n/3)$, so that $c(n) = -c_2(n) + c_2(n/3) = a(n) - a(n/3)$, exactly analogous to the quadratic case.

Assume now that K is a noncyclic cubic field, and let k its quadratic resolvent field, in other words $k = \mathbb{Q}(\sqrt{D})$. We first note that it follows from the quadratic case that (with evident notation)

$$S_{21}(s) = \frac{1}{2}\sum_{2\nmid n}\frac{\mu(n)}{n}\log((\zeta^2/\zeta_k)_{p\nmid aD}(ns))$$

(although a priori we do not need S_{21}, we will need it for S_{111}). A completely similar computation gives

$$S_3(s) = \frac{1}{3}\sum_{3\nmid n}\frac{\mu(n)}{n}\log((\zeta\zeta_k/\zeta_K)_{p\nmid aD}(ns))\,,$$

hence

$$S_{111}(s) = \sum_{n\geq 1} \frac{\mu(n)}{n} \log(\zeta_{p\nmid aD}(ns)) - S_{21}(s) - S_3(s) \ .$$

Thus, if in addition to c_1 and c_2 defined above we set

$$c_3(n) = \frac{1}{2n} \sum_{d|n,\ 2\nmid d} \mu(d)(3^{n/d} - 1) \ ,$$

we have

$$\begin{aligned}\log((P_3P_{111})(s)) &= -\sum_{n\geq 1}(c_1(n)\log(\zeta_{p\nmid aD}(ns)) \\ &\quad + c_2(n)\log((\zeta\zeta_k/\zeta_K)_{p\nmid aD}(ns)) + c_3(n)\log((\zeta^2/\zeta_k)_{p\nmid aD}(ns))) \\ &= \sum_{n\geq 1} a(n)\log((\zeta_K/\zeta)_{p\nmid aD}(ns)) + \\ &\quad \sum_{n\geq 1} b(n)\log((\zeta_k/\zeta)_{p\nmid aD}(ns)) + \sum_{n\geq 1} c(n)\log(\zeta_{p\nmid aD}(ns)) \ ,\end{aligned}$$

with $a(n) = -c_2(n)$, $b(n) = c_2(n) - c_3(n)$, and $c(n) = -c_1(n) + c_2(n) + c_3(n)$, and As usual $c_1(1) = 2$, $c_2(1) = 1$, and $c_3(1) = 1$, so $c(1) = 0$ as it should for convergence. As above, since $c_1(n) = 3c_2(n) - c_2(n/3)$ we check that $c(n) = a(n) - b(n) - a(n/3)$.

This shows that the formulas in the cyclic and noncyclic case can be unified by setting $b(n) = 0$ (and $\zeta_k/\zeta = 1$) in the cyclic case.

This leads to the following program:

HardyLittlewood3.gp

```
/* In the cyclic cubic case, we need z^3/z_K=z^2/(z_K/z) */
/* In the noncyclic cubic case, we need z^2/z_k=z/L_D
   and zz_k/z_K=z_k/(z_K/z)=zL_D/(z_K/z). */

/* Auxiliary functions. */
/* ZetaDN and LchiN are defined in Hardylittlewood2.gp. */

LchiKNinit(nf, P) =
{
  my(Ebad = [], Pbad = [], D = nf.disc);
  foreach (P, p,
    my(E, v = idealprimedec(nf, p));
    E = if (#v == 1, if (D % p == 0, next);
                     1 + 'x + 'x^2,
             #v == 2, if (D % p == 0, 1 - 'x, 1 - 'x^2),
             #v == 3, (1 - 'x)^2);
    Ebad = concat(Ebad, E);
    Pbad = concat(Pbad, p);
  );
  return ([Pbad, Ebad]);
}

HLW3(A, N, P, D) =
{ my(flnoncyc = (D != 1), nf, lim);
```

```
  my(LK, va, vb, vc, S1, S2, S3);
  my(B = getlocalbitprec(), EKbad);

  localbitprec(32);
  lim = ceil(B*log(2)/log(N/3)); nf = nfinit(A);
  localbitprec(B + ceil(1.585*lim) + exponent(lim));
  LK = lfundiv(lfuncreate(A), lfuncreate(1));
  LK = lfuninit(LK, [1/2, lim, 0]);
  va = vector(lim); vb = vector(lim); vc = vector(lim);
  forfactored(X = 1, lim,
    my([n] = X);
    S2 = S3 = 0;
    fordivfactored(X, Y,
      my([d] = Y, n3 = 3^(n/d), mob = moebius(Y));
      if (mob,
        if (d % 2 && flnoncyc, S2 += mob * (n3 - 1));
        if (d % 3, S3 += mob * n3)
      )
    );
    va[n] = -S3 / (3*n);
    if (flnoncyc, vb[n] = -(va[n] + S2 / (2*n)));
    vc[n] = va[n] - vb[n] - if (n%3 == 0, va[n/3], 0);
  );
  EKbad = LchiKNinit(nf, P);
  S1 = sum(n = 1, lim, va[n] * log(LchiN(LK, EKbad, n)));
  S2 = 0;
  if (flnoncyc,
    my(LD, Echibad = LchiNinit(D, P));
    LD = lfuninit(D, [1/2, lim, 0]);
    S2 = sum(n = 1, lim, vb[n] * log(LchiN(LD, Echibad, n)))
  );
  S3 = sum(n = 2, lim, vc[n] * log(ZetaDN(P, n)));
  return (S1 + S2 + S3);
}

/* Compute the Hardy-Littlewood constant of aX^3+bX^2+cX+d. */
HardyLittlewood3(A, N = 50) =
{ my(DA = poldisc(A), S = 1., Da6, P, v = variable(A));

  if (poldegree(A) != 3, error("polynomial of degree != 3"));
  my([a, b, c, d] = Vec(A));
  if (!polisirreducible(A) ||
       gcd([6 * a, 2 * b, a + b + c, d]) > 1, return (0));
  N = max(N, 5);
  /* Take care of bad primes and p <= N. */
  Da6 = abs(6 * a * DA);
  P = setunion(factor(Da6)[,1]~, primes([5, N]));
  foreach (P, p, S *= (p - #polrootsmod(A, p)) / (p - 1));
```

```
73    /* Do the real work. */
74    if (a != 1, A = a^2 * subst(A, v, v / a));
75    return (S * exp(HLW3(A, N, P, coredisc(DA))));
76  }
77
78  HardyLittlewood(A, N = 50) =
79  { my(d = poldegree(A));
80
81    if (d <= 0, return (0),
82        d == 1,
83          my(a = abs(polcoeff(A, 1))); return (a / eulerphi(a)),
84        d == 2, return (HardyLittlewood2(A, N)),
85        d == 3, return (HardyLittlewood3(A, N)),
86        error("HardyLittlewood not implemented for d >= 4")
87    );
88  }
```

This program illustrates a number of important GP commands:

(1) As in the quadratic case, we need the `lfun` command, but now not only for $\zeta_k/\zeta = L(\chi_D)$ (obtained in the function `LchiN`), but also for the nontrivial ζ_K/ζ obtained as follows: `lfuncreate(A)` creates ζ_K, `lfuncreate(1)` creates ζ in the necessary format for `lfun` functions, and finally `lfundiv` performs the division ζ_K/ζ.

(2) Note the use of `#polrootsmod(A, p)` which computes $\omega(p)$ directly, but is used only for a small number of primes (those dividing $6aD$ and those less than or equal to N).

(3) Note the use of `#idealprimedec(nf, p)` which allows the computation of the Euler factor of ζ_K/ζ at these same primes, using the function `LchiKNEulerbad` to determine this factor for the 5 possible types of prime decomposition in K. In fact, by a basic result of algebraic number theory, for the primes p not dividing `Da6`, we could use `#idealprimedec(nf, p)` instead of `#polrootsmod(A, p)` (of course, `idealprimedec` is necessary for the Euler factors).

5.7.2. C_ℓ **Polynomials.** What we did in the cyclic cubic case (not in the S_3 case) can immediately be generalized to the cyclic C_ℓ case for ℓ prime, generalizing both the quadratic and the cyclic cubic case.

Let $A(X) = \sum_{0\le n\le \ell} a_n X^n$ be an irreducible polynomial of degree exactly equal to ℓ, and assume that the number field K defined by A is abelian with Galois group C_ℓ. We can immediately copy what we did above. First, setting $D(A) = \mathrm{disc}(A)$ and $E(A) = \ell! a_\ell D(A)$, with evident notation we have

$$H(A) = \prod_{p\mid E(A)} \frac{p-\omega(p)}{p-1}(P_\ell P_{1^\ell})(1) \quad \text{with}$$

$$P_\ell(s) = \prod_{\substack{p\nmid E(A)\\ p\mathbb{Z}_K=\mathfrak{p}_\ell}} \frac{p^s}{p^s-1} \quad \text{and} \quad P_{1^\ell}(s) = \prod_{\substack{p\nmid E(A)\\ p\mathbb{Z}_K=\mathfrak{p}_1\mathfrak{p}_1'\cdots}} \frac{p^s-\ell}{p^s-1}\,,$$

and performing exactly the same computation we find that if we set

$$a(n) = -\frac{1}{\ell n} \sum_{d|n,\ \ell\nmid d} \mu(d)\ell^{n/d}$$

and $c(n) = a(n) - a(n/\ell)$, we have

$$\log((P_\ell P_{1^\ell})(s)) = \sum_{n\geq 1} a(n)\log((\zeta_K/\zeta)_{p\nmid E(A)}(ns)) + \sum_{n\geq 2} c(n)\log(\zeta_{p\nmid E(A)})(ns)\ .$$

We leave to the reader the immediate task of writing the corresponding `GP` script which will be essentially identical to the cyclic cubic one.

5.7.3. General Polynomials. The case of general polynomials of degree $d \geq 4$ is more complicated, and it is not completely clear how to treat them, at least in a straightforward manner. As in the quadratic and cubic case, we can evidently reduce to the computation of what one can call the Hardy–Littlewood constants of *number fields* (as opposed to polynomials). When the number field is *abelian* over $\mathbb{Q}$, there is no problem since we have d Artin L-functions which are simply Hecke L-series, and the splitting of the primes is completely understood in terms of these series (we have mentioned the case of C_ℓ-number fields above, when ℓ is prime).

If it is not abelian, or even Galois, it is not clear to the authors how to proceed in general, although it may not be too difficult. What we explained above for S_3-cubic fields can also easily be done for A_4-quartic fields, but it is not completely clear how to do it for D_4- or S_4-quartic fields. We leave to the reader to explore this avenue of computation.