Some new GP features

A tutorial

B. Allombert

IMB
CNRS/Université de Bordeaux

06/01/2025

factorlimit

We have changed the way the table of prime numbers is stored:

- The default `primelimit` is 2^{20} instead of 5×10^5.
- Primes are now indexed, so functions like prime, primepi etc are much faster.

```
? sum(i=1,5*10^5,primepi(i))
%1 = 10854805343
? ##
  ***   last result computed in 41 ms.
```

- a new default `factorlimit` (2^{20} by default) is used for trial factorization, which now uses a sieve, which is faster but requires more memory.

```
? for(i=1,100,factor(3^1000-1,5*10^5))
? ##
  ***   last result computed in 90 ms.
```

mapapply

`mapapply(~M,x,f,u)` if *x* is in the domain of *M*, apply *f* to the image of *x*, otherwise returns *u*(). The code below defines a function `maplistput(~M,k,b)` that appends *v* to the image of *k* by *M*:

```
? maplistput(~M, k, v) =
{
  mapapply(~M, k,
       (~y)->listput(~y,v),
       ()->List(v));
}
```

mapapply

```
? M = Map();
%6 = Map([;])
? maplistput(~M, "a", 1); M
%7 = Map(["a", List([1])])
? maplistput(~M, "a", 2); M
%8 = Map(["a", List([1, 2])])
? maplistput(~M, "b", 3); M
%9 = Map(["a", List([1, 2]); "b", List([3])])
? maplistput(~M, "a", 4); M
%10 = Map(["a", List([1, 2, 4]); "b", List([])])
```

nfweilheight

`nfweilheight(nf, v)` returns the absolute logarithmic Weil height of *v* seen as an element of the projective space over `nf`.

```
? Q = nfinit(x); Qi = nfinit(x^2 + 1);
? nfweilheight(Q, [1, 2, -3, 101])
%12 = 4.615120516841259450884198266912989156 9
? nfweilheight(Qi, [1, 2, -3, 101])
%13 = 4.615120516841259450884198266912989156 9
? nfweilheight(Qi, [1, x+2, x+3])
%14 = 1.151292546497022842008995727342182103 8
```

bnrstarkunit

Returns the characteristic polynomial of the (conjectural) Stark unit corresponding to class field theory parameters, which defines the corresponding Abelian extension. The ground field attached to `bnr` must be totally real and all but one infinite place must become complex in the class field, which must be a quadratic extension of its totally real subfield.

```
? bnf = bnfinit(y^2 - 2);
? bnr = bnrinit(bnf, [15, [1,0]]);
? P = lift(bnrstarkunit(bnr))
%17 = x^8 + (-9000*y - 12728)*x^7 + (57877380*y + 8
? rnfisabelian(bnf,P)
%18 = 1
? rnfconductor(bnf, P)[1]
%19 = [[15,0;0,15],[1,0]]
```

ellmaninconstant

`ellmaninconstant(E)` returns the Manin constant of the rational elliptic curve E.

```
? E  = ellinit("11a1");
? E2 = ellinit("11a2");
? E3 = ellinit("11a3");
? Et = elltwist(E,-7);
? E2t = elltwist(E2,-7);
? E3t = elltwist(E3,-7);
? F = intformal(Polrev(ellan(E,1000)));
? h = (-13+sqrt(-7))/22;
? z = 2*real(subst(F,x,exp(2*I*Pi*h))/sqrt(-7))
%28 = 0.30662673609548676460425818313168139005
```

ellmaninconstant

```
? ellztopoint(Et,z)
%29 = [9.0000000000,-63.99999999]
? ellztopoint(E2t,z)
%30 = [355.06250000,-3.515624999]
? ellztopoint(E3t,z)
%31 = [8.6896562869,-36.65083602]
? ellztopoint(E3t,z*5)
%32 = [3.9999999999,-12.99999999]
? ellmaninconstant(E)
%33 = 1
? ellmaninconstant(E2)
%34 = 1
? ellmaninconstant(E3)
%35 = 5
```

ellchangecompose, ellchangeinvert

```
? E=ellinit([1,2,3,4,5]);
? F=ellinit([61/16,127/32]);
? urst=ellisisom(E,F)
%38 = [1,-3/4,-1/2,-9/8]
? ellchangecurve(E,urst)[1..5]
%39 = [0,0,0,61/16,127/32]
? ursti = ellchangeinvert(E,urst)
%40 = [1,3/4,1/2,3/2]
? ellchangecurve(F,ursti)[1..5]
%41 = [1,2,3,4,5]
? ellchangecompose(E,urst,ursti)
%42 = [1,0,0,0]
```

hyperellordinate

`hyperellordinate(H,x)` returns the possible ordinates given the abscissa of a point on the curve *H*.

```
? P = x^6+3*x+2;
? hyperellordinate(P,1/2)
%44 = [15/8,-15/8]
? hyperellisoncurve(P,[1/2,15/8])
%45 = 1
? W = [x^6+5, x];
? hyperellordinate(W,1)
%47 = [2,-3]
? hyperellisoncurve(W,[1,2])
%48 = 1
```

derivative of `psi`

It is now possible to compute the derivatives of `psi`.

```
? (psi(21,10)-psi(1,10))/10!
%49 = 1.0004941886041119179811698149249862114
? harmonic(20,11)*1.
%50 = 1.0004941886041119179811698149249862114
? (psi(21+O(101^10),10)-psi(1+O(101^10),10))/10!
%51 = 77+32*101+50*101^2+8*101^3+74*101^4+81*101^5+
? harmonic(20,11)+O(101^10)
%52 = 77+32*101+50*101^2+8*101^3+74*101^4+81*101^5+
```

p-adic Hurwitz zeta with character

When *s* is p-adic, the third parameter of zetahurwitz is the power of the Teichmuller character to use.

```
? zetahurwitz(3+O(11^7),0,0)
%53 = 5*11^-1+1+10*11+9*11^2+5*11^3+8*11^4+3*11^5+O
? zeta(3+O(11^7))
%54 = 5*11^-1+1+10*11+9*11^2+5*11^3+8*11^4+3*11^5+O
? zetahurwitz(3+O(11^7),0,-2)
%55 = 5+4*11+11^2+5*11^4+8*11^5+O(11^6)
? -psi(O(11^7),2)/2!
%56 = 5+4*11+11^2+5*11^4+8*11^5+O(11^6)
```

Class groups for imaginary quadratic fields

We implemented the MPQS algorithm for class groups for imaginary quadratic fields.

```
? quadclassunit(1-2^128)
%57 = [156268974727446497 28,
%      [2034752275096 9596,12,2,2,2,2,2,2],
%      [Qfb(211744127470077 3864,-4429167859022 32929,
? ##
  ***   last result computed in 1,022 ms.
```

Algebras

new GP functions `algnewprec`, `algeltfromnf`, `algmodprinit`, `algmodpr`, `alginvol`, `algmodprlift`, `algeichlerbasis`, `algquattobasis`, `algbasistoquat`, `algisquatalg`.
See Aurel's tutorial

removed functions

We removed three useless GP functions:

1. `trap`: use `iferr` instead.
2. `listcreate(n)`: use `List()` instead (*n* was ignored).
3. `listkill(L)`: not needed anymore, do `L=List()` to empty a list.

XEUS-GP, a jupyter engine for PARI/GP notebook

We released XEUS-GP, a jupyter engine for PARI/GP notebooks which is designed to behave similarly to GP.