Some new GP features

A tutorial

B. Allombert

IMB
CNRS/Université de Bordeaux

27/02/2023

forvec

A short-cut is available:

```
? forvec(v=[[0,1],[0,2]],print1(v," "))
[0,0] [0,1] [0,2] [1,0] [1,1] [1,2]
? forvec(v=[2,3],print1(v," "))
[0,0] [0,1] [0,2] [1,0] [1,1] [1,2]
```

nfeltissquare, nfeltispower

`nfeltissquare` tests if a number field element is a square, and `nfeltispower` tests if a number field element is a *n*-power

```
? nf=nfinit(a^2-2);
? nfeltissquare(nf,-70*a+99,&z)
%4 = 1
? z
%5 = [7,-5]~
? nfbasistoalg(nf,nfeltpow(nf,z,2))
%6 = Mod(-70*a+99,a^2-2)
? nfeltispower(nf,-70*a+99,3,&z)
%7 = 1
? z
%8 = [3,-2]~
```

bnrcompositum

`bnrcompositum` allows to compute compositum of Abelian extensions given by class field parameters.

```
? Q = bnfinit(y);
? bnr1 = bnrinit(Q, [7, [1]]); bnr1.cyc
%10 = [6]
? bnr2 = bnrinit(Q, [13, [1]]); bnr2.cyc
%11 = [12]
? H1 = Mat(2); bnrclassfield(bnr1, H1)
%12 = [x^2 + 7]
? H2 = Mat(2); bnrclassfield(bnr2, H2)
%13 = [x^2 - 13]
? [bnr,H] = bnrcompositum([bnr1, H1], [bnr2,H2]);
? bnrclassfield(bnr,H)
%15 = [x^2 - 13, x^2 + 7]
```

lfuncreate([bnr, subg]): Dedekind zeta for the Abelian extension defined by lass field parameters.

```
? bnf = bnfinit(a^2 - a - 9);
? bnr = bnrinit(bnf, [2, [0,0]]); subg = Mat(3);
? L = lfuncreate([bnr, subg]);
? P = bnrclassfield(bnr,subg,2)
%19 = x^6-24*x^4+144*x^2-148
? lfunan(P,100) == lfunan(L,100)
%20 = 1
```

elltrace

`elltrace(E,P)` computes the sum of the Galois conjugates of the point P on the elliptic curve corresponding to E.

```
? E = ellinit([-13^2, 0]);
? P = Mod([2,5], a^2-2); \\ defined over Q, seen ov
? elltrace(E,P) == ellmul(E,P,2)
%23 = 1
? P = Mod([-10*x^3+10*x-13, -16*x^3+16*x-34], x^4-x
? ellisoncurve(E,P)
%25 = 1
? Q = elltrace(E,P)
%26 = [11432100241 / 375584400, 1105240264347961 /
? ellisoncurve(E,Q)
%27 = 1
```

hyperelliptic curves

An hyperelliptic curve in PARI is given by a pair `[P,Q]` of polynomial which define the curve

$$y^2 + Q(x)y = P(x)$$

`hyperelldisc` computes the discriminant of the curve

```
? hyperelldisc([x^5,1])
%28 = 3125
```

hyperellminimalmodel

`hyperellminimalmodel` computes a minimal model of the curve, that is a model with minimal discriminant.

```
? W = [x^6+216*x^3+324,0];
? D = hyperelldisc(W)
%30 = 1828422898924853919744000
? Wn = hyperellminimalmodel(W)
%31 = [2*x^6+18*x^3+1,x^3];
? hyperelldisc(Wn)
%32 = 29530050606000
```

The minimal discriminant can be computed directly with `hyperellminimaldisc`

```
? hyperellminimaldisc(W)
%33 = 29530050606000
```

hyperellminimalmodel

`hyperellminimalmodel` also return the variable change, which can be applied with `hyperellchangecurve`

```
? Wn = hyperellminimalmodel(W,&M)
%34 = [2*x^6+18*x^3+1,x^3];
? M
%35 = [18,[3,0;0,1],9*x^3]
? hyperellchangecurve(W, M)
%36 = [2*x^6+18*x^3+1,x^3]
```

The variable change is given by $[e, [a, b; c, d], H]$. If (x, y) is a point on the new model, the corresponding point (X, Y) on the original model is given by

$$X = (ax + b)/(cx + d)$$

$$Y = e(y + H(x))/(cx + d)^{g+1}$$

.

hyperellisoncurve

`hyperellisoncurve` check whether a point is on the curve:

```
? L = hyperellratpoints(Wn,10)
%37 = [[-2,5],[-2,3],[0,1],[0,-1],[2,13],[2,-21],[1
? hyperellisoncurve(Wn, [2,13])
%38 = 1
? my([x,y]=[2,13]);y^2+x^3*y-(2*x^6+18*x^3+1)
%39 = 0
```

genus2igusa

For genus 2 curve, the Igusa invariants can be computed with genus2igusa

```
? genus2igusa(W)
%40 = [404352,-6701667840,1237283079782400,11384638
? genus2igusa(Wn)
%41 = [2808,-323190,414363600,264770303175,29530050
? genus2igusa(Wn,2)
%42 = 2808
? genus2igusa(Wn,4)
%43 = -323190
```

factormodcyclo

`factormodcyclo(n,p)` factors the n-th cyclotomic polynomial modulo p, faster than `factormod`.

```
? lift(factormodcyclo(15, 11))
%44 = [x^2+3*x+9,x^2+4*x+5,x^2+5*x+3,x^2+9*x+4]~
? factormodcyclo(15, 11, 1) \\ single
%45 = Mod(1, 11)*x^2 + Mod(5, 11)*x + Mod(3, 11)
? z1 = lift(factormod(polcyclo(12345),11311)[,1]);
time = 32,498 ms.
? z2 = factormodcyclo(12345,11311);
time = 47 ms.
? z1 == z2
%48 = 1
```

qfminimize

Given a square symmetric matrix G with rational coefficients, and non-zero determinant, return $[H, U]$ such that $H = c * U \ * G * U$ for some rational c, and H integral with minimal determinant. The coefficients of U are usually nonintegral.

```
? G = matdiagonal([650, -104329, -104329]);
? [H,U]=qfminimize(G); H
%50 = [-1,0,0;0,-1,0;0,0,1]
? U
%51 = [0,0,1/5;5/323,-1/323,0;-1/323,-5/323,0]
? U~*G*U
%52 = [-26,0,0;0,-26,0;0,0,26]
```

Hence $c = 26$ in this example.

Lerch transcendent

```
? lerchphi(I,2,1) -( Catalan + I * Pi^2/48 )
%53 = 0.E-38-7.3468396926392969 25E-40*I
? lerchzeta(2,1,1/4) -( Catalan + I * Pi^2/48 )
%54 = 0.E-38-7.3468396926392969 25E-40*I
```

intnumosc

`intnumosc` allows to compute oscillating integrals.

```
? intnum(x = 0, [oo,Pi*I],besselj(0,x))
%55 = 1.0358754514716231320822785644343149590 \\ WR
? intnumosc(x = 0, besselj(0,x),Pi)
%56 = 0.99999999999999999999999999999999999990 \\ G
```

setdelta

`setdelta` computes the symmetric difference of two sets:

```
? setdelta(Set([2,3,5,7,11]),Set([1,2,3,4,5]))
%57 = [1,4,7,11]
```