Some new GP features

A tutorial

B. Allombert

IMB
CNRS/Université de Bordeaux

08/01/2024

polfromroots

`polfromroots(`$[a_1, \ldots, a_n]$`)` returns the monic polynomial whose roots are $[a_1, \ldots, a_n]$, that is

$$\prod_{i=1}^{n} (x - a_i) \quad .$$

```
? P = polfromroots([1,2,3,4,5])
%1 = x^5-15*x^4+85*x^3-225*x^2+274*x-120
? nfroots(,%)
%2 = [1,2,3,4,5]~
```

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]
```

parforstep, parforprimestep

`parforstep`, `parforprimestep`: like `parfor` and `forstep`, `forprimestep`

```
? f(p)=vecmin(abs(qfbsolve(Qfb(1,0,1),p)));
? export(f);
? S=0;n=0; parforprimestep(p=2,100000,Mod(1,4),\
    f(p), v, S+=v/sqrt(p); n++);
? S/n-(4-2*sqrt(2))/Pi
%8 = 0.0011410512306778232792698714891924635094
```

normfu

`quadclassunit` now returns the norm of the fundamental unit. This can be accessed by `.normfu`

```
? C = quadclassunit(5*13)
%9 = [2,[2],[Qfb(2,5,-5)],2.77...,-1]
? C.normfu
%10 = -1
? C = quadclassunit(13*17)
%11 = [2,[2],[Qfb(5,11,-7)],2.70...,1]
? C.normfu
%12 = 1
```

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)
%14 = 1
? z
%15 = [7,-5]~
? nfbasistoalg(nf,nfeltpow(nf,z,2))
%16 = Mod(-70*a+99,a^2-2)
? nfeltispower(nf,-70*a+99,3,&z)
%17 = 1
? z
%18 = [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
%20 = [6]
? bnr2 = bnrinit(Q, [13, [1]]); bnr2.cyc
%21 = [12]
? H1 = Mat(2); bnrclassfield(bnr1, H1)
%22 = [x^2 + 7]
? H2 = Mat(2); bnrclassfield(bnr2, H2)
%23 = [x^2 - 13]
? [bnr,H] = bnrcompositum([bnr1, H1], [bnr2,H2]);
? bnrclassfield(bnr,H)
%25 = [x^2 - 13, x^2 + 7]
```

Dedekind zeta for Abelian extension

`lfuncreate([bnr, subg])`: Dedekind zeta for the Abelian extension defined by class 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)
%29 = x^6-24*x^4+144*x^2-148
? lfunan(P,100) == lfunan(L,100)
%30 = 1
```

ellisisom

`ellisisom(E,F)`: checks whether the curve E and F are isomorphic.

```
? E=ellinit([1,2,3,4,5]);
? F=ellinit([61/16,127/32]);
? urst=ellisisom(E,F)
%33 = [1,-3/4,-1/2,-9/8]
? ellchangecurve(E,urst)[1..5]
%34 = [0,0,0,61/16,127/32]
```

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);
```

P is a rational point seen over a quadratic extension.

```
? elltrace(E,P) == ellmul(E,P,2)
%37 = 1
```

elltrace

```
? P = [-10*x^3+10*x-13, -16*x^3+16*x-34];
? P = Mod(P, x^4-x^3+2*x-1);
? ellisoncurve(E,P)
%40 = 1
? Q = elltrace(E,P)
%41 = [11432100241 / 375584400,
%       1105240264347961 / 7278825672000]
? ellisoncurve(E,Q)
%42 = 1
```

elliscm, polisclass

`elliscm(E)` checks whether the elliptic curve E over a number field has CM. `polisclass(P)` checks whether P is a class polynomial.

```
? E = ellinit([0,1,0,-3,1]); elliscm(E)
%43 = -8
? ellisomat(E)[2]
%44 = [1,2;2,1]
? Enf = ellinit(E,nfinit(a^2+2));
? elliscm(Enf)
%46 = -8
? ellisomat(Enf)
%47 = -8
? F=ellinit([a],nfinit(polclass(-23,,a)));
? elliscm(F)
%49 = -23
```

ellsupersingularj

`ellsupersingularj(p)` returns a random supersingular j-invariant defined over $\mathbb{F}_{p^2}$, almost uniformly distributed.

```
? j = ellsupersingularj(1009)
%51 = 12*w+295
? ellissupersingular(j)
%52 = 1
? a = ffgen([1009,2],'a);
? j = ellsupersingularj(a)
%54 = 867*a+721
? ellissupersingular(j)
%55 = 1
```

hyperelliptic curves

A 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])
%56 = 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)
%58 = 18284228989248539197440000
? Wn = hyperellminimalmodel(W)
%59 = [2*x^6+18*x^3+1,x^3];
? hyperelldisc(Wn)
%60 = 29530050606000
```

The minimal discriminant can be computed directly with `hyperellminimaldisc`

```
? hyperellminimaldisc(W)
%61 = 29530050606000
```

hyperellminimalmodel

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

```
? Wn = hyperellminimalmodel(W,&M)
%62 = [2*x^6+18*x^3+1,x^3];
? M
%63 = [18,[3,0;0,1],9*x^3]
? hyperellchangecurve(W, M)
%64 = [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` checks whether a point is on the curve:

```
? L = hyperellratpoints(Wn,10)
%65 = [[-2,5],[-2,3],[0,1],[0,-1],[2,13],
%       [2,-21],[1/2,7/4],[1/2,-15/8],
%       [2/3,65/27],[2/3,-73/27]]
? hyperellisoncurve(Wn, [2,13])
%66 = 1
? my([x,y]=[2,13]);y^2+x^3*y-(2*x^6+18*x^3+1)
%67 = 0
```

genus2igusa

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

```
? genus2igusa(W)
%68 = [404352,-6701667840,1237283079782400,
%      1138463840096206848000,
%      18284228989248539197440000]
? genus2igusa(Wn)
%69 = [2808,-323190,414363600,
%      264770303175,295300506060000]
? genus2igusa(Wn,2)
%70 = 2808
? genus2igusa(Wn,4)
%71 = -323190
```

factormodcyclo

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

```
? lift(factormodcyclo(15, 11))
%72 = [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
%73 = 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
%76 = 1
```

qfsolve

`qfsolve` now allows to provide the factorisation of the discriminant of the form.

```
? M=matdiagonal([1,1,1,-(53!+1)]);
? qfsolve(M)
%78 = [-40867229639170742768520048063854592,-780737
? ##
  ***   last result computed in 3,131 ms.
? F=factor(matdet(M));
? ##
  ***   last result computed in 3,130 ms.
? qfsolve([M,F])
%82 = [-40867229639170742768520048063854592,-780737
? ##
  ***   last result computed in 2 ms.
```

qfminimize

Given a square symmetric matrix G with rational coefficients, and non-zero determinant, return $[H, U]$ such that $H = c^t UGU$ 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
%85 = [-1,0,0;0,-1,0;0,0,1]
? U
%86 = [0,0,1/5;5/323,-1/323,0;-1/323,-5/323,0]
? U~*G*U
%87 = [-26,0,0;0,-26,0;0,0,26]
```

Hence $c = 26$ in this example.

Lerch transcendent

$$\Phi(z, s, a) = \sum_{n \geq 0} z^n (n + a)^{-s}$$

```
? lerchphi(I,2,1) -( Catalan + I * Pi^2/48 )
%88 = 0.E-38-7.346839692639296925E-40*I
```

$$L(s, a, \lambda) = \sum_{n \geq 0} e^{2\pi i \lambda n} (n + a)^{-s}$$

```
? lerchzeta(2,1,1/4) -( Catalan + I * Pi^2/48 )
%89 = 0.E-38-7.346839692639296925E-40*I
```

intnumosc

`intnumosc` allows to compute oscillating integrals of quasi-periodic functions.

```
? intnum(x = 0, [oo,Pi*I],besselj(0,x))
%90 = 1.0358754514716231320822785644343149590
```

(wrong)

```
? intnumosc(x = 0, besselj(0,x),Pi)
%91 = 0.99999999999999999999999999999999999990
```

(correct)

setdelta

`setdelta` computes the symmetric difference of two sets:

```
? setdelta(Set([2,3,5,7,11]),Set([1,2,3,4,5]))
%92 = [1,4,7,11]
```

plotell

`plotell` allows to plot circles and ellipses

```
? plotinit(1);plotmove(1,0,0);
? plotell(1,50,50); plotdraw([1,100,100]);
```

Miscellaneous

```
? 7#
%93 = 210
? 2*3*5*7
%94 = 210
? Qfb(2*x^2+3*x+4)
%95 = Qfb(2,3,4)
? Qfb([2,3,4])
%96 = Qfb(2,3,4)
? Qfb([2,3/2;3/2,4])
%97 = Qfb(2,3,4)
? M=matrix(3,4);[#M, #M~]
%98 = [4,3]
? digits(11,-2)
%99 = [1,1,1,1,1]
```