Some new GP features

A tutorial

B. Allombert

IMB
CNRS/Université de Bordeaux

14/01/2019

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 676541

fileopen

GP provides a new I/O interface that mirrors the C interface and is faster. Files are opened with `fileopen` and closed with `fileclose`.
To create a file with the power of 2 on separated lines:

```
? n = fileopen("myfile","w");
? for(i=1,10,filewrite(n,2^i));
? fileclose(n)
```

to read a file physical line by physical line:

```
? n = fileopen("myfile");
? while (l = filereadstr(n), print(l))
? fileclose(n)
```

fileextern

to read a file logical line by logical line:

```
? n = fileopen("myfile");
? while (l = fileread(n), print(l))
? fileclose(n)
```

to read "ls" output line by line:

```
? n = fileextern("ls /");
? while (l = filereadstr(n), print(l))
? fileclose(n)
```

forprimestep

To loop over prime number in an arithmetic progression:

```
? forprimestep(p = 2, 50, Mod(1,5), print(p))
11
31
41
```

For consistency, `forstep` also allow this:

```
? forstep(p = 2, 20, Mod(1,5), print(p))
6
11
16
```

forsquarefree

`forsquarefree` is identical to `forfactored` except that it only loops over squarefree numbers

```
? forsquarefree(N=1,10,print(N))

[1,matrix(0,2)]
[2,Mat([2,1])]
[3,Mat([3,1])]
[5,Mat([5,1])]
[6,[2,1;3,1]]
[7,Mat([7,1])]
[10,[2,1;5,1]]
```

forsquarefree

```
? my(s=0.);forsquarefree(N=1,10^6, \
  s+=moebius(N)/N[1]^2);s
%16 = 0.60792710204046183281498023606240746441
? ##
  ***   last result computed in 729 ms.
? my(s=0.);forfactored(N=1,10^6, \
  s+=moebius(N)/N[1]^2);s
%17 = 0.60792710204046183281498023606240746441
? ##
  ***   last result computed in 1,060 ms.
```

factor

`factor` allows now to specify the factorisation domain as a second parameter.

```
? factor(x^4+1)
%18 = Mat([x^4+1,1])
? factor(x^4+1,I)
%19 = [x^2-I,1;x^2+I,1]
? factor(x^4+1,Mod(1,3))
%20 = [Mod(1,3)*x^2+Mod(1,3)*x+Mod(2,3),1;Mod(1,3)*
? factor(x^4+1,ffgen(9,'a))
%21 = [x+a,1;x+(a+1),1;x+2*a,1;x+(2*a+2),1]
? factor(x^4+1,Mod(a,a^2-2))
%22 = [x^2+Mod(-a,a^2-2)*x+1,1;x^2+Mod(a,a^2-2)*x+1
```

factormod

factormod and polrootsmod now handle finite fields too:

```
? a=ffgen(3^2,'a);
? factormod((x^4+1)*Mod(1,3))
%24 = [Mod(1,3)*x^2+Mod(1,3)*x+Mod(2,3),1;Mod(1,3)*
? factormod(x^4+1,3)
%25 = [Mod(1,3)*x^2+Mod(1,3)*x+Mod(2,3),1;Mod(1,3)*
? factormod((x^4+1)*a^0)
%26 = [x+a,1;x+(a+1),1;x+2*a,1;x+(2*a+2),1]
? factormod(x^4+1,a)
%27 = [x+a,1;x+(a+1),1;x+2*a,1;x+(2*a+2),1]
? polrootsmod(x^4+1,a)
%28 = [a,a+1,2*a,2*a+2]~
? polrootsmod((x^4+1)*a^0)
%29 = [a,a+1,2*a,2*a+2]~
```

factormodSQF, factormodDDF

idem but return the square free factorization (resp. the distinct degree factorization):

```
? factormodSQF(x*(x+1)*(x^2+1)^2,3)
%30 = [x^2+x,1;x^2+1,2]
? factormodDDF((x^4+1)*(x^2+a))
%31 = [x^4+1,1;x^2+a,2]
```

plotexport

The functions `plothexport` and `plotexport` return a string that is the SVG or PostScript representation of the plot.

```
? write("sin.svg",plothexport("svg",x=0,1,sin(x)))
? plotinit(1);plotmove(1,10,10);plotrbox(1,20,20);
? plotexport("ps",1)
%34 = "%!\n50 50 translate\n/p {moveto 0 2 rlineto
```

plotcolor

`plotcolor` allows to specify arbitrary colors:

```
? {
  plotinit(1);
  plotcolor(1,"#003399");
  plotbox(1,600,400,1);
  plotcolor(1,"#ffcc00");
  for(j=0,11,
    plotmove(1,300+130*cos(2*Pi*j/12),
               200-130*sin(2*Pi*j/12));
    for(i=0,4,
      plotrline(1,cos(2*Pi*3*i/5)*40,
                 -sin(2*Pi*3*i/5)*40)));
  plotdraw([1,0,0]);
}
```

string functions

The function `Strchr`, `Strexpand`, `Strprintf`, `Strtex` has been renamed to `strchr`, `strexpand`, `strprintf`, `strtex`.
Two new functions have been added:

```
? strsplit("a,b,c,d",",")
%36 = ["a","b","c","d"]
? strjoin(["a","b","c"],":")
%37 = "a:b:c"
```

galoisgetname

`galoisgetname(o,n)` returns a string describing the group of order *o* and index *n* in the GAP4 library of small groups.

```
? N = galoisgetgroup(12); \\ # of abstract groups
? for(i=1, N, print(i,":",galoisgetname(12,i)))

1:C3 : C4
2:C12
3:A4
4:D12
5:C6 x C2
```

galoisgetgroup

`galoisgetgroup(o,n)` returns the corresponding abstract group.

```
? G=galoisgetgroup(12,3);
? [T,o]=galoischartable(G);
? T~
%42 =
%[1    1    1   1]
%[1 -y-1    y   1]
%[1    y -y-1   1]
%[3    0    0  -1]
```

qfbsolve

`qfbsolve(q,n)` now accept non-prime *n* and returns all the solutions up to units of positive norms.

```
? qfbsolve(Qfb(1,0,1),65)
%43 = [[8,-1],[7,4],[7,-4],[-8,-1]]
? qfbsolve(Qfb(1,1,-1),-1)
%44 = [[0,1]]
```

hypergeom

`hypergeom` allow to compute hypergeometric functions

$$hypergeom([a_1,\ldots,a_p],[b_1,\ldots,b_q],z) = \sum_{n=0}^{\infty} \frac{\prod_{i=1}^{p}(a_i)_n}{\prod_{j=1}^{q}(b_j)_n}(z^n)/(n!)$$

```
? hypergeom([],[],2)
%45 = 7.3890560989306502272304274605750078132
? f(z)=hypergeom([1,2],[3],z);
? lindep([f(1/2),f'(1/2),f''(1/2)])
%47 = [-8,4,1]~
```

airy

`airy` allow to compute the Airy functions *Ai* and *Bi* which gives a basis of solutions of the differential equation $y'' = xy$.

```
? airy(2)
%48 = [0.03492413042327437914,3.2980949999782147100]
? airy''(2)/2
%49 = [0.03492413042327437914,3.2980949999782147110]
```

lfunsympow

`lfunsympow(E,m)` return the L-function associated to the m symmetric power of the elliptic curve E defined over $\mathbb{Q}$.

```
? E=ellinit([0,-1,1,-10,-20]);
? L=lfunsympow(E,2);
? lfun(L,2)
%52 = 1.0575992445909578493475116523231674725
? -(2*Pi*E.omega[1]*imag(E.omega[2]))/11
%53 = 1.0575992445909578493475116523231674725
```

polteichmuller

This function allow to find a model of an unramified extension of $\mathbb{Q}_p$ such that the Frobenius is given by $X \mod P \mapsto X^p \mod P$ up to a fixed p-adic precision.

```
? T = ffinit(3, 3, 't)
%54 = Mod(1,3)*t^3 + Mod(1,3)*t^2 + Mod(1,3)*t + Mo
? P = polteichmuller(T,3,5)
%55 = t^3 + 166*t^2 + 52*t + 242
? subst(P, t, t^3) % (P*Mod(1,3^5))
%56 = Mod(0, 243)
```

mfgaloisprojrep

If F is a modular form of weight 1 and type A_4 or S_4, `mfgaloisprojrep` gives a polynomial that defines the kernel of the projective Artin representation attached to F.

```
? mfl=mfinit([124,1,0],1);
? apply(mfgaloistype,mfl)
%58 = [[],[-12]]
? mf=mfl[2];
? F=mfeigenbasis(mf)[1];
? P=mfgaloisprojrep(mf,F)
%61 = x^12+196*x^10+14376*x^8+469152*x^6+5836432*x^
? G=galoisinit(P);
? T=galoisfixedfield(G,G.gen[3],1)
%63 = x^4+392*x^3+57504*x^2+3741312*x+91097344
```

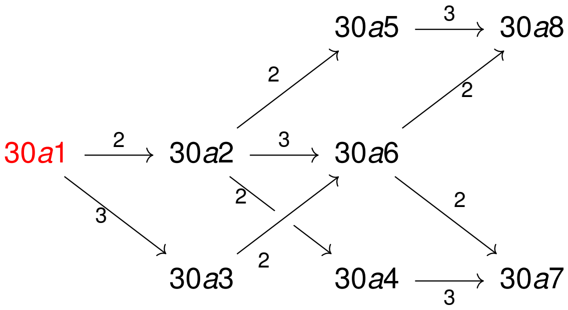
ellisotree

return the oriented graph of isogeny of prime degrees that preserve the Néron differential.

```
? E=ellinit([1, 0, 1, 1, 2]);
? [L,M]=ellisotree(E);
? M
%66 =
%[0 2 0 0 3 0 0 0]
%[0 0 2 2 0 3 0 0]
%[0 0 0 0 0 0 3 0]
%[0 0 0 0 0 0 0 3]
%[0 0 0 0 0 2 0 0]
%[0 0 0 0 0 0 2 2]
%[0 0 0 0 0 0 0 0]
%[0 0 0 0 0 0 0 0]
```

ellisotree

which is the adjacency matrix of the following graph:



algebras

The following functions for semi-simple algebras have been added: `alglatadd`, `alglatcontains`, `alglatelement`, `alglathnf`, `alglatindex`, `alglatinter`, `alglatlefttransporter`, `alglatmul`, `alglatrighttransporter`, `alglatsubset`, `algsplit`
Please Ask aurel!

parallelism

new functions `export`, `exportall`, `unexport`, `unexportall`
See tomorrow talk.

idealispower

`idealispower` allow to compute the *n*-th root of an ideal when it exists.

```
? K = nfinit(x^3 - 2);
? A = [46875, 30966, 9573; 0, 3, 0; 0, 0, 3];
? idealispower(K, A, 3, &B)
%69 = 1
? B
%70 =
%[75 22 41]
%[ 0  1  0]
%[ 0  0  1]
```

idealredmodpower

Try to reduce an ideal modulo powers.

```
? T = x^6+108; nf = nfinit(T); a = Mod(x,T);
? setrand(1); u = (2*a^2+a+3)*random(2^1000*x^6)^2;
? b = idealredmodpower(nf,u,2);
? v2 = nfeltmul(nf,u, nfeltpow(nf,b,2))
%74 = [34, 47, 15, 35, 9, 3]~
```

Miscellaneous

```
? dirpowers(10,3)
%75 = [1,8,27,64,125,216,343,512,729,1000]
? pollaguerre(5)
%76 = -1/120*x^5+5/24*x^4-5/3*x^3+5*x^2-5*x+1
? log1p(1)
%77 = 0.69314718055994530941723212145817656807
? log1p(10.^-30)
%78 = 9.99999999999999999999999999999500000000E-31
? log(1.+10.^-30)
%79 = 9.9999999729360509 69E-31
? serchop(1/x+x+3*x^2+O(x^3),0)
%80 = x+3*x^2+O(x^3)
? serchop(1/x+x+3*x^2+O(x^3),2)
%81 = 3*x^2+O(x^3)
```

getlocalprec

```
? localprec(100);getlocalprec()
%82 = 115
? localbitprec(1000);getlocalbitprec()
%83 = 1000
```