Some new GP features

A tutorial

B. Allombert

IMB
CNRS/Université de Bordeaux

20/01/2020

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 676541
Some new GP features

References

It is now possible to pass vectors and matrices by references with ~.

? V = [1, 2, 3];
? f(V, x) = V[x]++;
? f(V, 1)
%3 = 2
? V
%4 = [1, 2, 3]
? f(~V, x) = V[x]++;
? f(~V, 1)
%6 = 2
? V
%7 = [2, 2, 3];
Some new GP features

References

It also works with lists and map. For better legibility it is encouraged to use ~ with listput and mapput.

? L = List([1,2,3]);
? f(~L,x) = listput(~L,x^2);
? f(~L,5)
%10 = 25
? L
%11 = List([1,2,3,25])
User-defined member functions now always use references:

```plaintext
? a.inc=a[1]++;  
? V.inc  
%13 = 2  
? V  
%14 = [3,2,3]
```
Some new GP features

References

Using references avoid copies when passing large objects:

? W=vector(1000,i,i!);
? f(x)=x[1];
? g(~x)=x[1];
? x.b = x[1];
? default(timer,1);
? for(i=1,10^5,W[1])
time = 5 ms.
? for(i=1,10^5,f(W))
time = 3,718 ms.
? for(i=1,10^5,g(~W))
time = 12 ms.
? for(i=1,10^5,W.b)
time = 12 ms.
? default(timer,0);
Some new GP features

polrootspadic

polrootspadic now support unramified extensions:

? T = y^2+y+1; p = 2;
? lift(polrootspadic(x^3-x^2+64*y, [T,p], 5))
%26 = [(2^3+O(2^5))*y+(2^3+O(2^5)),
% (2^3+2^4+O(2^5))*y+(2^3+2^4+O(2^5)),
% 1+O(2^5)]~

This also works with hyperellpadicfrobenius.
Some new GP features

qfbsolve

qfbsolve now accepts a flag to select output:
  ▶ bit 0: return one / all the solutions modulo units of positive norms.
  ▶ bit 1: return primitive / non-primitive solutions

Primitive means that $x$ and $y$ are coprime.

? qfbsolve(Qfb(1,0,1),65,0)
%27 = [8,-1]

? qfbsolve(Qfb(1,0,1),65,1)
%28 = [[8,-1],[7,4],[7,-4],[-8,-1]]

? qfbsolve(Qfb(1,0,1),65,2)
%29 = [8,-1]

? qfbsolve(Qfb(1,0,1),65,3)
%30 = [[8,-1],[7,4],[7,-4],[-8,-1]]
Some new GP features

qfbsolve

? qfbsolve(Qfb(1,0,1),20,0)
%31 = []
? qfbsolve(Qfb(1,0,1),20,1)
%32 = []
? qfbsolve(Qfb(1,0,1),20,2)
%33 = [-4,-2]
? qfbsolve(Qfb(1,0,1),20,3)
%34 = [[-4,-2],[4,-2]]
Some new GP features

matreduce

matreduce reduce factorization matrices with redundant factors.

? M = matconcat([factor(12),factor(20)]~)
%35 = [2,2;3,1;2,2;5,1]
? F=matreduce(M)
%36 = [2,4;3,1;5,1]
? factorback(F)
%37 = 240
**fft, fftinv**

Compute fast Fourier transform of order $2^n$, given the vector of roots of unity.

```plaintext
> P = x^3+2*x^2+3*x+4; w = rootsof1(4)
%38 = [1,I,-1,-I]~
> f = fft(w, P)
%39 = [10,2+2*I,2,2-2*I]
> apply(z->subst(P,x,z),w)
%40 = [10,2+2*I,2,2-2*I]~
> fi = fftinv(w, f)
%41 = [16,12,8,4]
> Polrev(fi/#fi)
%42 = x^3+2*x^2+3*x+4
```
Some new GP features

fft, fftinv

Over a finite field:

\[
\begin{align*}
? & \ w = \text{powers}(\text{znprimroot}(5), 3) \\
%43 & = [\text{Mod}(1, 5),\text{Mod}(2, 5),\text{Mod}(4, 5),\text{Mod}(3, 5)] \\
? & \ f = \text{fft}(w, P) \\
%44 & = [\text{Mod}(0, 5),\text{Mod}(1, 5),\text{Mod}(2, 5),\text{Mod}(3, 5)] \\
? & \ \text{apply}(z->\text{subst}(P, x, z), w) \\
%45 & = [\text{Mod}(0, 5),\text{Mod}(1, 5),\text{Mod}(2, 5),\text{Mod}(3, 5)] \\
? & \ fi = \text{fftinv}(w, f) \\
%46 & = [\text{Mod}(1, 5),\text{Mod}(2, 5),\text{Mod}(3, 5),\text{Mod}(4, 5)] \\
? & \ \text{lift}(\text{Polrev}(fi/#fi)) \\
%47 & = x^3+2*x^2+3*x+4
\end{align*}
\]
Euler numbers and polynomials

Analogous to Bernoulli polynomials $B_n(x)$ and numbers $B_n$ (bernpol, bernfrac) satisfying

$$\frac{te^{xt}}{e^t - 1} = \sum_{n \geq 0} B_n(x) \frac{t^n}{n!}, \quad B_n = B_n(0),$$

we now have Euler polynomials $E_n(x)$ (eulerpol) and numbers $E_n$ (eulerfrac)

$$\frac{2}{e^{xt} + 1} = \sum_{n \geq 0} E_n(x) \frac{t^n}{n!}, \quad E_n = 2^n E_n(1/2)$$

\%48 = \text{serlaplace(1/cosh(t+O(t^10)))}
\%48 = 1-t^2+5*t^4-61*t^6+1385*t^8+O(t^10)
\%49 = \text{vector(10,i,eulerfrac(i))}
\%49 = [0,-1,0,5,0,-61,0,1385,0,-50521]
Some new GP features

eulerfrac, eulerpol, eulervec, eulerianpol

Similarly, in addition to $\text{bernvec}(n) = [B_0, B_2, \ldots, B_{2n}]$ we
now have $\text{eulervec}(n) = [E_0, E_2, \ldots, E_{2n}]$. We also have
Eulerian polynomials $A_n(x)(\text{eulerianpol})$

$$\frac{x}{1 + x - e^{tx}} = \sum_{n \geq 0} A_n(x + 1) \frac{t^n}{n!}, \quad \text{s.t.} \sum_{j \geq 0} x^j j^n = \frac{x A_n(x)}{(1 - x)^{n+1}}.$$

? eulervec(5)
%50 = [1,-1,5,-61,1385,-50521]

? eulerpol(5)
%51 = x^5-5/2*x^4+5/2*x^2-1/2

? vector(4,i,eulerianpol(i))
%52 = [1,x+1,x^2+4*x+1,x^3+11*x^2+11*x+1]
Asymptotic expansion

The function `asympnum` computes numerically as many terms of an asymptotic expansion \( f(n) \approx a_0 + a_1/n + \cdots + a_k/n^k \) as it can. But it fails when the expansion is not rational. The variant `asympnumraw` takes \( k \) as extra argument (mandatory!) and approximates \((a_0, \ldots, a_k)\) without assumptions. This allows for instance to take periods into account before rationalizing.

\[
\begin{align*}
? f(n) &= n! / (n^n \times \exp(-n) \times \sqrt{n}) \\
? \text{asympnum}(f) \\
\%54 &= [] \quad \text{\textbackslash \ text fail} \\
? v = \text{asympnumraw}(f, 3) \\
\%55 &= [2.506\ldots, 0.208\ldots, 0.008\ldots, -0.006\ldots] \\
? \text{bestappr}(v / v[1]) \\
\%56 &= [1, 1/12, 1/288, -139/51840] \quad \text{\textbackslash \ Stirling exp.}
\end{align*}
\]

In the above, we don’t need to know that \( v[1] \approx \sqrt{2\pi}. \)
ffmaprel

Extend partial maps between finite fields.

? a = ffgen([3,5], 'a);
? b = ffgen([3,10], 'b);
? m = ffembed(a, b);
? mi = ffinvmap(m);

$m$ is the inclusion from $\mathbb{F}_{3^5}$ to $\mathbb{F}_{3^{10}}$. $mi$ is the reverse partial map from the image of $m$ to $\mathbb{F}_{3^5}$. `ffmaprel` extends $mi$ to a map from $\mathbb{F}_{3^{10}}$ to an algebraic extension of $\mathbb{F}_{3^5}$. 
Some new GP features

**ffmaprel**

```plaintext
? R = ffmaprel(mi,b)
%61 = Mod(b,b^2+(a+1)*b+(a^2+2*a+2))

This can be used to compute relative minimal polynomials:

? minpoly(R)
%62 = x^2+(a+1)*x+(a^2+2*a+2)

? trace(R)
%63 = 2*a+2

? norm(R)
%64 = a^2+2*a+2
```
Some new GP features

**nfsubfields**

* nfsubfieldsmax return the maximal subfields,  
* nfsubfieldscm return the maximal CM subfields, see Aurel talk.

? P = x^8+3*x^4+5;  
? nfsubfields(P)  
%66 = [[x, 0], [x^2-3*x+5, -x^4], [x^4+3*x^2+5, -x^2], [x^8+3*x^4+5, x]]  
? nfsubfieldsmax(P)  
%67 = [[x^4+3*x^2+5, x^2]]  
? nfsubfieldscm(P)  
%68 = [x^2+11, 2*x^4+3]
**nfdiscfactors**

 nfdiscfactors returns the discriminant and its factorization.

 ? nfdiscfactors(x^3+3*x+7)  
%69 = [-1431, [3, 3; 53, 1]]

 nfbasis now can return the discriminant in an optional argument:

 ? nfbasis(x^3+3*x+7, &dK)  
%70 = [1, x, x^2]  
? dK  
%71 = -1431
idealismaximal, idealdown

idealismaximal checks whether an ideal is maximal and returns the corresponding prid:

? a = 'a;
? nf = nfinit(a^3-2);
? idealismaximal(nf,7)
%74 = [7, [7,0,0]~,1,3,1]
? idealismaximal(nf,5)
%75 = 0

idealdown returns a generator of the intersection of the ideal with \( \mathbb{Z} \).

? id2 = idealprimedec(nf,2)[1];
? id3 = idealprimedec(nf,3)[1];
? idealdown(nf, idealmul(nf, id2, id3))
%78 = 6
Some new GP features

bnrclassfield

bnrclassfield computes ray class fields without the limitation of \texttt{rnfkummer}. See Aurel tutorial.

? bnf=bnfinit(a^2+41); bnf.cyc
79 = [8]
? P=bnrclassfield(bnf,,1)
%80 = x^8-2*a*x^7-66*x^6+26*a*x^5+189*x^4-8*a*x^3+32*x^2+8*a*x+4
? rnfdisc(bnf,P)
%81 = [1,-1]
bnfunits allows to access the compact representation of units, see Karim talk.

? bnf = bnfinit(x^2-nextprime(2^38), 1);
? sizebyte(bnf.fu)
%83 = 193216
? sizebyte(bnfunits(bnf))
%84 = 5672
Some new GP features

Faltings height

*ellheight* can now be used to obtain the Faltings height of an elliptic curve.

```plaintext
? ellheight(ellinit([1,3]))
%85 = -0.62991512865301812208879099375776471315

? ellheight(ellinit([1,a],nfinit(a^2+1)))
%86 = -0.82141261022274297551562408240979575893
```
lfun now returns rational special values of quadratic character exactly:

? lfun(1,-7)
%87 = 1/240
? lfun(-4,-8)
%88 = 1385/2
? lfun(5,-9)
%89 = -825502/25
If \( f \) is a \( L \)-function, allow to create \( s \mapsto f(s - d) \) and \( s \mapsto f(s)f(s - d) \).

? \( L = \text{lfunshift}(1,1); \quad \backslash \backslash \ zeta(s-1) \)
? \text{lfun}(L,1)
\%91 = -0.50000000000000000000000000000000000000

? \( M = \text{lfunshift}(1,1,1); \quad \backslash \backslash \ zeta(s) \ast zeta(s-1) \)
? \text{lfun}(M,2)
\%93 = 1.6449340668482264364724151666 \ast x^\backslash -1 + O(x^0)
Some new GP features

`lfuncreate` can now handle data that depend on the precision.

```gp
? G=znstar(7,1); chi=[2]; \ cubic char of cond 7
? V=[rootsof1(3)~,3];
? r=sqrtn((-13-sqrt(-27))/14,6); \ root number
? an(V)=n->vector(n,i,chareval(G,chi,i,V));
? L=lfuncreate([an(V),1,[0],1,7,r]);
? lfuncheckfeq(L)
%99 = -126
? localbitprec(256); lfuncheckfeq(L)
%100 = -128
```
**Ifuncreate**

We create a closure that return the ldata structure with the current precision.

```plaintext
? F() = 
{
    my(V=[rootsoln(3)~,3]);
    my(r=sqrtnt((-13-sqrt(-27))/14,6));
    [an(V),1,[0],1,7,r];
}
? L = ifuncreate(F);
? ifuncheckfeq(L)
%103 = -126
? localbitprec(256); ifuncheckfeq(L)
%104 = -254
```
Some new GP features

Multiple characters in lfun

`lfun` now allow to pass multiple characters if they have \( L \)-functions with the same functional equation (different root numbers are allowed).

? G=znstar(17,1); C=[[1],[3],[5],[7]]; ? lfun([G,C],1)
%106 = [1.60101836-0.392774395*I,0.990332401+0.01073
% 0.336453687+0.304143387*I,1.02655704+0.711485755*I]
? lfunrootres([G,C])
%107 = [0,0,[0.825809120-0.563949729*I,0.988439629+
% 0.974084005-0.226186541*I,-0.139252958+0.9
Some new GP features

**Multiple characters in Ifun**

```plaintext
? default(timer,1);
? localprec(1000); lfun([G,C],1);
time = 4,188 ms.
? localprec(1000); [lfun([G,c],1)|c<-C];
time = 14,372 ms.
? default(timer,0);

Multiple Hecke characters are also supported.

? bnf = bnfinit(x^2+47); bnr = bnrinit(bnf,1);
? lfun([bnr,[[1],[2]]],1)
%113 = [0.64666083128645259893546663,
      0.450660220947390529728481755]
```
Some new GP features

**mfisetaquo**

*mfisetaquo* try to write modular forms as eta quotients.

```gp
? find(a,b)=
{
    forell(e,a,b,
        my(E=ellinit(e[1]));
        my(F=mffromell(E)[2]);
        my(Q=mfisetaquo(F));
        if(Q,print(e[1],":",Q)),1);
}

? find(1,100)
% 11a1:[1,2;11,2]
% 14a1:[1,1;2,1;7,1;14,1]
% 15a1:[1,1;3,1;5,1;15,1]
```
Some new GP features

**Miscellaneous**

```plaintext
? print(strtime(12345678))
%3h, 25min, 45,678 ms

? derivn((x*(1-x))^4,4)
%117 = 1680*x^4-3360*x^3+2160*x^2-480*x+24

? arity((x,y)->x^2+y^2)
%118 = 2

? arity(sin)
%119 = 1

? L=List([1,2,3]);L[1..2]
%120 = List([1,2])

? L=List([1,2,3]);L[^1]
%121 = List([2,3])

? svg=parploithexport("svg",x=1,10,1/gamma(x));
```