Introduction to PARI/GP

B. Allombert

IMB
CNRS/Université de Bordeaux

08/04/2019

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

Introduction

- PARI is a C library, allowing fast computations.
- GP is an easy-to-use interactive shell giving access to the PARI functions.
- GP is the name of gp's scripting language.
- GP2C, the GP$\rightarrow$ PARI compiler allows to convert GP scripts to C.

Basic objects

```
? 57!
%1 = 40526919504877216755680601905432...
? 2 / 6
%2 = 1/3
? (1+I)^2
%3 = 2*I
? (x+1)^(-2)
%4 = 1/(x^2+2*x+1)
? Mod(2,5)^3
%5 = Mod(3,5)
? Mod(x, x^2+x+1)^3
%6 = Mod(1,x^2+x+1)
? w = ffgen([3,5],'w); w^12 \\ in F_3^5
%7 = 2*w^4+2*w^3+2
```

Basic objects

```
? Pi
%8 = 3.1415926535897932384626433832795028842
? log(2)
%9 = 0.69314718055994530941723212145817656807
? \p100
? log(2)
%11 = 0.6931471805599453094172321214581765680755001
? exp(%)
%12 = 2.000000000000000000000000000000000000000000
? log(1+x)
%13 = x-1/2*x^2+1/3*x^3-1/4*x^4+1/5*x^5-...
? exp(%12)
%14 = 1+x+O(x^16)
```

Functions

```
? ?

  1: PROGRAMMING under GP
  2: Standard monadic or dyadic OPERATORS
  3: CONVERSIONS and similar elementary functions
  4: functions related to COMBINATORICS
  5: NUMBER THEORETICAL functions
  6: POLYNOMIALS and power series
  7: Vectors, matrices, LINEAR ALGEBRA and sets
  8: TRANSCENDENTAL functions
  9: SUMS, products, integrals and similar functions
 10: General NUMBER FIELDS
 11: Associative and central simple ALGEBRAS
 12: ELLIPTIC CURVES
 13: L-FUNCTIONS
 14: MODULAR FORMS
```

Help

```
? ?4
? ?atan

atan(x): arc tangent of x.


? ??atan

atan(x):

   Principal branch of
   tan^{-1}(x) = log ((1+ix)/(1-ix)) / 2i.
```

```
? ??
? ??refcard
? ??refcard-nf
? ??tutorial
? ???determinant

algdisc             bnfsunit            charker
ellpadicregulator   forsubgroup         matdet
mathnfmod           matrixqz            mspolygon
polresultant        rnfdet

See also:
  Finite abelian groups
  Pseudo-bases, determinant
```

Vectors and matrices

```
? V = [1,2,3];
? W = [4,5,6]~;
? M = [1,2,3;4,5,6]
%3 =
[1 2 3]
[4 5 6]
? V*W
%27 = 32
? M*W
%28 = [32,77]~
? U = [1..10]
%29 = [1,2,3,4,5,6,7,8,9,10]
```

Components

```
? V[2]
%30 = 2
? W[1..2]
%31 = [4,5]~
? M[2,2]
%32 = 5
? M[1,]
%33 = [1,2,3]
? M[,2]
%34 = [2,5]~
? M[1..2,1..2]
%12 =
[1 2]
[4 5]
```

Polymorphism

```
? \o0
? factor(91)
%37 = [7,1;13,1]
? factor(x^4+4)
%38 = [x^2-2*x+2,1;x^2+2*x+2,1]
? factor((x^4+1)*Mod(1,a^2-2))
%39 = [x^2+Mod(-a,a^2-2)*x+1,1;x^2+Mod(a,a^2-2)*x+1
? factor((x^4+4)*Mod(1,13))
%40 = [Mod(1,13)*x+Mod(4,13),1;Mod(1,13)*x+Mod(6,13
? factor(x^4+1,Mod(1,a^2-2))
%41 = [x^2+Mod(-a,a^2-2)*x+1,1;x^2+Mod(a,a^2-2)*x+1
? factor(x^4+1,Mod(1,13))
%42 = [Mod(1,13)*x^2+Mod(5,13),1;Mod(1,13)*x^2+Mod(
```

Numerical summation

```
? intnum(x=0,1,1/(1+x^2))/Pi
%43 = 0.25000000000000000000000000000000000000
? sumnum(n=1,1/n^2)/Pi^2
%44 = 0.16666666666666666666666666666666666667
? sumalt(n=0,(-1)^n/(2*n+1))*4
%45 = 3.1415926535897932384626433832795028842
? sumalt(n=1,(-1)^n*log(n)) \\ diverging!
%46 = 0.22579135264472743236309761494744107198
? 2*exp(2*%)
%47 = 3.1415926535897932384626433832795028854
```

Comprehension

```
? [n^2|n<-[1..10]]
%48 = [1,4,9,16,25,36,49,64,81,100]
? [n^2|n<-[1..10],isprime(n)]
%49 = [4,9,25,49]
? [n^2|n<-primes([1,10])]
%50 = [4,9,25,49]
? [a,b] = [1,2];
? print("a=",a," b=",b)
% a=1 b=2
```

Control structures

- `if(cond,expr_true{,expr_false})`
- `while(cond, expr)`
- `for(var=start,end,expr(var))`
- `forstep(var=start,end,step,expr(var))`
- `forprime(var=start,end,expr(var))`
- `fordiv(N,var,expr(var))`

To configure the memory used by PARI, In the file `.gprc` (or `gprc.txt` under windows) add

```
parisizemax=1G
```

or do

```
default(parisizemax,"1G");
```

if the message 'the PARI stack overflows !' appears.