Introduction to PARI/GP

B. Allombert

IMB
CNRS/Université de Bordeaux

12/01/2022

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 = 405269195048772167556806019054 32...
? 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)
%10 = 0.69314718055994530941723212145817656807550013
? exp(%)
%11 = 2.00000000000000000000000000000000000000000000
? log(1+x)
%12 = x-1/2*x^2+1/3*x^3-1/4*x^4+1/5*x^5-...
? exp(%12)
%13 = 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
%17 = 32
? M*W
%18 = [32,77]~
? U = [1..10]
%19 = [1,2,3,4,5,6,7,8,9,10]
```

Components

```
? V[2]
%20 = 2
? W[1..2]
%21 = [4,5]~
? M[2,2]
%22 = 5
? M[1,]
%23 = [1,2,3]
? M[,2]
%24 = [2,5]~
? M[1..2,1..2]
%12 =
[1 2]
[4 5]
```

Constructors

```
? V=vector(10,i,1/i)
%26 = [1,1/2,1/3,1/4,1/5,1/6,1/7,1/8,1/9,1/10]
? W=vectorv(10,i,1/i)
%27 = [1,1/2,1/3,1/4,1/5,1/6,1/7,1/8,1/9,1/10]~
? [1/i | i<-[1..10]]
%28 = [1,1/2,1/3,1/4,1/5,1/6,1/7,1/8,1/9,1/10]
? [1/i | i<-[1..10]]~
%29 = [1,1/2,1/3,1/4,1/5,1/6,1/7,1/8,1/9,1/10]~
? M=matrix(4,4,i,j,i*j)
%30 = [1,2,3,4;2,4,6,8;3,6,9,12;4,8,12,16]
? matdiagonal([1,2,3,4])
%31 = [1,0,0,0;0,2,0,0;0,0,3,0;0,0,0,4]
```

building matrices

Matrices internally are lines of columns.

```
? C1 = [1,2,3]~; C2=[4,5,6]~;
? concat(C1,C2)
%33 = [1,2,3,4,5,6]~
? matconcat([C1,C2])
 [1,4;2,5;3,6]
? matid(5)
? matconcat([C1,C2])
```

linear algebra

Linear algebra follows French convention, matrices act on column vectors on the left.

```
? matdet([1,2;3,4])
%37 = -2
? M = [1,2,3;4,5,6];
? M~
%39 = [1,4;2,5;3,6]
? matsize(M)
%40 = [2,3]
? matrank(M)
%41 = 2
? K=matker(M)
%42 = [1;-2;1]
? M*K
%43 = [0;0]
```

LLL

```
?  V = vector(10,i,random(10^10));
?  M = matconcat([matid(10),V]~);
?  T = qflll(M)
%47 = [2,2,3,3,-2,-2,1,5,-1,3;...]
?  B = qflll(M,3)
%48 = [2,2,3,3,-2,-2,1,5,-1,3;...]
?  M*T==B
%49 = 1
?  Q = M~*M;
?  U = qflllgram(Q)
%51 = [2,2,3,3,-2,-2,1,5,-1,3;...]
?  T == U
%52 = 1
```

Integer lattices

For definite positive matrix

```
? Q = matkerint(Mat(concat([2..24],-70)));
? Q = Q~*Q; Q[23,23]-=2; Q
%54 = [3,1,1,1,-1,0,1,1,1,0,0,0,0,1,1,-1,-1,1,-1,0,
? qfsign(Q)
%55 = [23,0]
? L = qfminim(Q); L[1..2]
%56 = [4600,3]
? V = L[3][,1]
%57 = [14,19,24,11,-32,38,-39,-30,-38,-12,57,-14,-1
? qfeval(Q,V)
%58 = 3
? G=qfauto(Q); G[1]
%59 = 84610842624000
```

Polymorphism

```
? \o0
? factor(91)
%61 = [7,1;13,1]
? factor(x^4+4)
%62 = [x^2-2*x+2,1;x^2+2*x+2,1]
? factor((x^4+1)*Mod(1,a^2-2))
%63 = [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))
%64 = [Mod(1,13)*x+Mod(4,13),1;Mod(1,13)*x+Mod(6,13
? factor(x^4+1,Mod(1,a^2-2))
%65 = [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))
%66 = [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
%67 = 0.2500000000000000000000000000000000000000
? sumnum(n=1,1/n^2)/Pi^2
%68 = 0.1666666666666666666666666666666666666667
? sumalt(n=0,(-1)^n/(2*n+1))*4
%69 = 3.1415926535897932384626433832795028842
? sumalt(n=1,(-1)^n*log(n)) \\ diverging!
%70 = 0.22579135264472743236309761494744107198
? 2*exp(2*%)
%71 = 3.1415926535897932384626433832795028854
```

Comprehension

```
? [n^2|n<-[1..10]]
%72 = [1,4,9,16,25,36,49,64,81,100]
? [n^2|n<-[1..10],isprime(n)]
%73 = [4,9,25,49]
? [n^2|n<-primes([1,10])]
%74 = [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.