

Pari-GP reference card

(PARI-GP version 2.17.0)

Note: optional arguments are surrounded by braces {}.

To start the calculator, type its name in the terminal: **gp**

To exit **gp**, type **quit**, **\q**, or **<C-D>** at prompt.

Help

| | |
|--|----------------|
| describe function | ?function |
| extended description | ??keyword |
| list of relevant help topics | ???pattern |
| name of GP-1.39 function f in GP-2.* | whatnow(f) |

Input/Output

| | |
|--|------------------------|
| previous result, the result before | %, %`, %`` , etc. |
| n -th result since startup | % n |
| separate multiple statements on line | ; |
| extend statement on additional lines | \ |
| extend statements on several lines | { seq_1 ; seq_2 ;} |
| comment | /* ... */ |
| one-line comment, rest of line ignored | \\ ... |

Metacommands & Defaults

| | |
|--|----------------------------|
| set default d to val | default({ d },{ val }) |
| toggle timer on/off | # |
| print time for last result | ## |
| print defaults | \d |
| set debug level to n | \g n |
| set memory debug level to n | \gm n |
| set n significant digits / bits | \p n , \pb n |
| set n terms in series | \ps n |
| quit GP | \q |
| print the list of PARI types | \t |
| print the list of user-defined functions | \u |
| read file into GP | \r $filename$ |
| set debuglevel for domain D to n | setdebug(D,n) |

Debugger / break loop

| | |
|--|---------------------------|
| get out of break loop | break or <C-D> |
| go up/down n frames | dbg_up({ n }), dbg_down |
| set break point | breakpoint() |
| examine object o | dbg_x(o) |
| current error data | dbg_err() |
| number of objects on heap and their size | getheap() |
| total size of objects on PARI stack | getstack() |

PARI Types & Input Formats

| | |
|---|---------------------------------------|
| t_INT . Integers; hex, binary | ± 31 ; $\pm 0x1F$, $\pm 0b101$ |
| t_REAL . Reals | ± 3.14 , 6.022 E23 |
| t_INTMOD . Integers modulo m | Mod(n,m) |
| t_FRAC . Rational Numbers | n/m |
| t_FFELT . Elt in finite field \mathbf{F}_q | ffgen(q , 't) |
| t_COMPLEX . Complex Numbers | $x + y * I$ |
| t_PADIC . p -adic Numbers | $x + O(p^k)$ |
| t_QUAD . Quadratic Numbers | $x + y * \text{quadgen}(D, \{ 'w \})$ |
| t_POLMOD . Polynomials modulo g | Mod(f,g) |
| t_POL . Polynomials | $a * x^n + \dots + b$ |
| t_SER . Power Series | $f + O(x^k)$ |
| t_RFRAC . Rational Functions | f/g |
| t_QFB . Binary quadratic form | Qfb(a,b,c) |
| t_VEC/t_COL . Row/Column Vectors | [x,y,z], [x,y,z]~ |
| t_VEC integer range | [1..10] |

| | |
|--|-----------------------|
| t_VECSMALL . Vector of small ints | Vecsmall([x,y,z]) |
| t_MAT . Matrices | [$a,b;c,d$] |
| t_LIST . Lists | List([x,y,z]) |
| t_STR . Strings | "abc" |
| t_INFINITY . $\pm\infty$ | +oo, -oo |

Reserved Variable Names

| | |
|---|-----------------------|
| $\pi \approx 3.14$, $\gamma \approx 0.57$, $C \approx 0.91$, $I = \sqrt{-1}$ | Pi, Euler, Catalan, I |
| Landau's big-oh notation | O |

Information about an Object, Precision

| | |
|---------------------------------------|---------------------------------------|
| PARI type of object x | type(x) |
| length of x / size of x in memory | # x , sizebyte(x) |
| real precision / bit precision of x | precision(x), bitprecision(x) |
| p -adic, series prec. of x | padicprec(x,p), serprec(x,v) |
| current dynamic precision | getlocalprec, getlocalbitprec |

Operators

| | |
|--|--|
| basic operations | +, -, *, /, ^, sqr |
| $i \leftarrow i+1$, $i \leftarrow i-1$, $i \leftarrow i*j$, ... | i++, i--, i*=j,... |
| Euclidean quotient, remainder | $x \backslash y$, $x \backslash y$, $x \% y$, divrem(x,y) |
| shift x left or right n bits | $x << n$, $x >> n$ or shift($x, \pm n$) |
| multiply by 2^n | shiftmul(x,n) |
| comparison operators | <=, <, >=, >, ==, !=, ==, lex, cmp |
| boolean operators (or, and, not) | , &&, ! |
| bit operations | bitand, bitneg, bitor, bitxor, bitnegimply |
| maximum/minimum of x and y | max(x,y), min(x,y) |
| sign of x (gives $-1, 0, 1$) | sign(x) |
| binary exponent of x | exponent(x) |
| derivative of f , 2nd derivative, etc. | f' , f'' , ... |
| differential operator | diffop($f,v,d,\{n=1\}$) |
| quote operator (formal variable) | 'x |
| assignment | x = value |
| simultaneous assignment $x \leftarrow v[1]$, $y \leftarrow v[2]$ | [x,y] = v |

Select Components

| | |
|---|--------------------------------------|
| <i>Caveat</i> : components start at index $n = 1$. | |
| n -th component of x | component(x,n) |
| n -th component of vector/list x | $x[n]$ |
| components $a, a+1, \dots, b$ of vector x | $x[a..b]$ |
| (m,n) -th component of matrix x | $x[m,n]$ |
| row m or column n of matrix x | $x[m,]$, $x[,n]$ |
| numerator/denominator of x | numerator(x), denominator(x) |

Random Numbers

| | |
|---------------------------------|-----------------------------------|
| random integer/prime in $[0,N[$ | random(N), randomprime(N) |
| get/set random seed | getrand, setrand(s) |

Conversions

| | |
|--|------------------------|
| to vector, matrix, vec. of small ints | Col/Vec, Mat, Vecsmall |
| to list, set, map, string | List, Set, Map, Str |
| create $(x \bmod y)$ | Mod(x,y) |
| make x a polynomial of v | Pol($x,\{v\}$) |
| variants of Pol <i>et al.</i> , in reverse order | Polrev, Vecrev, Colrev |
| make x a power series of v | Ser($x,\{v\}$) |
| convert x to simplest possible type | simplify(x) |
| object x with real precision n | precision(x,n) |
| object x with bit precision n | bitprecision(x,n) |
| set precision to p digits in dynamic scope | localprec(p) |
| set precision to p bits in dynamic scope | localbitprec(p) |

Character strings

| | |
|---|-------------------------|
| convert to TeX representation | strtex(x) |
| string from bytes / from format+args | strchr, sprintf |
| split string / join strings | strsplit, strjoin |
| convert time t ms. to h, m, s, ms format | strtime(t) |
| Conjugates and Lifts | |
| conjugate of a number x | conj(x) |
| norm of x , product with conjugate | norm(x) |
| L^p norm of x (L^∞ if no p) | normlp($x,\{p\}$) |
| square of L^2 norm of x | norml2(x) |
| lift of x from Mods and p -adics | lift, centerlift(x) |
| recursive lift | liftall |
| lift all t_INT and t_PADIC (\rightarrow t_INT) | liftint |
| lift all t_POLMOD (\rightarrow t_POL) | liftpol |

Lists, Sets & Maps

| | |
|--|-----------------------------|
| Sets (= row vector with strictly increasing entries w.r.t. cmp) | |
| intersection of sets x and y | setintersect(x,y) |
| set of elements in x not belonging to y | setminus(x,y) |
| symmetric difference $x \Delta y$ | setdelta(x,y) |
| union of sets x and y | setunion(x,y) |
| does y belong to the set x | setsearch($x,y,\{flag\}$) |
| set of all $f(x,y)$, $x \in X$, $y \in Y$ | setbinop(f,X,Y) |
| is x a set ? | setisset(x) |

| | |
|---|-------------------------------|
| Lists . create empty list: $L = \text{List}()$ | |
| append x to list L | listput($L,x,\{i\}$) |
| remove i -th component from list L | listpop($L,\{i\}$) |
| insert x in list L at position i | listinsert(L,x,i) |
| sort the list L in place | listsort($L,\{flag\}$) |
| Maps . create empty dictionary: $M = \text{Map}()$ | |
| attach value v to key k | mapput(M,k,v) |
| recover value attach to key k or error | mapget(M,k) |
| is key k in the dict? (set v to $M(k)$) | mapisdefined($M,k,\{\&v\}$) |
| evaluate f at $M(k)$ | mapapply(M,k,f) |
| remove k from map domain | mapdelete(M,k) |

GP Programming

User functions and closures

x,y are formal parameters; y defaults to Pi if parameter omitted;
 z,t are local variables (lexical scope), z initialized to 1.

| | |
|--|---------------------------------|
| fun (x, y=Pi) = my(z=1, t); seq | |
| fun = (x, y=Pi) -> my(z=1, t); seq | |
| attach help message h to s | addhelp(s,h) |
| undefine symbol s (also kills help) | kill(s) |
| Control Statements (X : formal parameter in expression seq) | |
| if $a \neq 0$, evaluate seq_1 , else seq_2 | if($a,\{seq_1\},\{seq_2\}$) |
| eval. seq for $a \leq X \leq b$ | for($X = a,b,seq$) |
| ...for $X \in v$ | foreach(v,X,seq) |
| ...for primes $a \leq X \leq b$ | forprime($X = a,b,seq$) |
| ...for primes $\equiv a \pmod{q}$ | forprimestep($X = a,b,q,seq$) |
| ...for composites $a \leq X \leq b$ | forcomposite($X = a,b,seq$) |
| ...for $a \leq X \leq b$ stepping s | forstep($X = a,b,s,seq$) |
| ...for X dividing n | fordiv(n,X,seq) |
| ... $X = [n, factor(n)]$, $a \leq n \leq b$ | forfactored($X = a,b,seq$) |
| ...as above, n squarefree | forsquarefree($X = a,b,seq$) |
| ... $X = [d, factor(d)]$, $d \mid n$ | fordivfactored(n,X,seq) |
| multivariable for , lex ordering | forvec($X = v,seq$) |

loop over partitions of n
... permutations of S
... subsets of $\{1, \dots, n\}$
... k -subsets of $\{1, \dots, n\}$
... vectors $v, q(v) \leq B$; $q > 0$
... $H < G$ finite abelian group
evaluate seq until $a \neq 0$
while $a \neq 0$, evaluate seq
exit n innermost enclosing loops
start new iteration of n -th enclosing loop
return x from current subroutine

Exceptions, warnings
raise an exception / warning
type of error message E
try seq_1 , evaluate seq_2 on error

Functions with closure arguments / results
number of arguments of f
select from v according to f
apply f to all entries in v
evaluate $f(a_1, \dots, a_n)$
evaluate $f(\dots f(f(a_1, a_2), a_3) \dots, a_n)$
calling function as closure

Sums & Products
sum $X = a$ to $X = b$, initialized at x
sum entries of vector v
product of all vector entries
sum $expr$ over divisors of n
... assuming $expr$ multiplicative
product $a \leq X \leq b$, initialized at x
product over primes $a \leq X \leq b$

Sorting
sort x by k -th component
min. m of x ($m = x[i]$), max.
does y belong to x , sorted wrt. f
 $\prod g^x \rightarrow$ factorization (\Rightarrow sorted, unique g)

Input/Output
print with/without $\backslash n$, TeX format
pretty print matrix
print fields with separator
formatted printing
write $args$ to file
write x in binary format
read file into GP
... return as vector of lines
... return as vector of strings
read a string from keyboard

Files and file descriptors
File descriptors allow efficient small consecutive reads or writes from or to a given file. The argument n below is always a descriptor, attached to a file in **r**(ead), **w**(rite) or **a**(ppend) mode.
get descriptor n for file $path$ in given $mode$
... from shell cmd output (pipe)

close descriptor
commit pending write operations
read logical line from file
... raw line from file
write $s \backslash n$ to file
... write s to file

forpart($p = n, seq$)
forperm(S, p, seq)
forsubset(n, p, seq)
forsubset($[n, k], p, seq$)
forqvec(v, q, b, seq)
forsubgroup($H = G$)
until(a, seq)
while(a, seq)
break($\{n\}$)
next($\{n\}$)
return($\{x\}$)

error(), warning()
errname(E)
iferr(seq_1, E, seq_2)

arity(f)
select(f, v)
apply(f, v)
call(f, a)
fold(f, a)
self()

sum($X = a, b, expr, \{x\}$)
vecsum(v)
vecprod(v)
sumdiv($n, X, expr$)
sumdivmult($n, X, expr$)
prod($X = a, b, expr, \{x\}$)
prodeuler($X = a, b, expr$)

vecsort($x, \{k\}, \{fl = 0\}$)
vecmin($x, \{\&i\}$), vecmax
vecsearch($x, y, \{f\}$)
matreduce(m)

print, print1, printtex
printp
printsep(sep, \dots), printsep1
printf()

write, write1, writetex($file, args$)
writebin($file, x$)
read($\{file\}$)
readvec($\{file\}$)
readstr($\{file\}$)
input()

Pari-GP reference card

(PARI-GP version 2.17.0)

Timers
CPU time in ms and reset timer
CPU time in ms since gp startup
time in ms since UNIX Epoch
timeout command after s seconds

Interface with system
allocates a new stack of s bytes
alias old to new
install function from library
execute system command a
... and feed result to GP
... returning GP string
get \$VAR from environment
expand env. variable in string

gettime()
getabstime()
getwalltime()
alarm($s, expr$)

allocatemem($\{s\}$)
alias(new, old)
install($f, code, \{gpf\}, \{lib\}$)
system(a)
extern(a)
externstr(a)
getenv("VAR")
strexpend(x)

Parallel evaluation

These functions evaluate their arguments in parallel (pthreads or MPI); args. must not access global variables (use export for this) and must be free of side effects. Enabled if threading engine is not *single* in gp header.

evaluate f on $x[1], \dots, x[n]$
evaluate closures $f[1], \dots, f[n]$
as select
as sum
as vector
eval f for $i = a, \dots, b$
... for each element x in v
... for p prime in $[a, b]$
... for $p = a \bmod q$
... for $i = a, a + s, \dots, b$
... multivariate
export x to parallel world
... all dynamic variables
frees exported value x
... all exported values

parapply(f, x)
pareval(f)
parselect($f, A, \{flag\}$)
parsum($i = a, b, expr$)
parvector($n, i, \{expr\}$)
parfor($i = a, \{b\}, f, \{r\}, \{f_2\}$)
parforeach($v, x, f, \{r\}, \{f_2\}$)
parforprime($p = a, \{b\}, f, \{r\}, \{f_2\}$)
parforprimestep($p = a, \{b\}, q, f, \{r\}, \{f_2\}$)
parforstep($i = a, \{b\}, s, f, \{r\}, \{f_2\}$)
parforvec($X = v, f, \{r\}, \{f_2\}, \{flag\}$)

export(x)
exportall()
unexport(x)
unexportall()

Linear Algebra

dimensions of matrix x
multiply two matrices
... assuming result is diagonal
concatenation of x and y
extract components of x
transpose of vector or matrix x
adjoint of the matrix x
eigenvectors/values of matrix x
characteristic/minimal polynomial of x
trace/determinant of matrix x
permanent of matrix x
Frobenius form of x
QR decomposition
apply matqr's transform to v

matsize(x)
 $x * y$
matmultodiagonal(x, y)
concat($x, \{y\}$)
vecextract($x, y, \{z\}$)
 $x \sim$, mattranspose(x)
matadjoin(x)
mateigen(x)
charpoly(x), minpoly(x)
trace(x), matdet(x)
matpermanent(x)
matfrobenius(x)
matqr(x)
mathouseholder(Q, v)

Constructors & Special Matrices

$\{g(x) : x \in v \text{ s.t. } f(x)\}$
 $\{x : x \in v \text{ s.t. } f(x)\}$
 $\{g(x) : x \in v\}$
row vec. of $expr$ eval'd at $1 \leq i \leq n$
col. vec. of $expr$ eval'd at $1 \leq i \leq n$
vector of small ints

[g(x) | x <- v, f(x)]
[x | x <- v, f(x)]
[g(x) | x <- v]
vector($n, \{i\}, \{expr\}$)
vectorv($n, \{i\}, \{expr\}$)
vectorsmall($n, \{i\}, \{expr\}$)

$[c, c \cdot x, \dots, c \cdot x^n]$
 $[1, 2^x, \dots, n^x]$
matrix $1 \leq i \leq m, 1 \leq j \leq n$
define matrix by blocks
diagonal matrix with diagonal x
is x diagonal?
 $x \cdot \text{matdiagonal}(d)$
 $n \times n$ identity matrix
Hessenberg form of square matrix x
 $n \times n$ Hilbert matrix $H_{ij} = (i + j - 1)^{-1}$
 $n \times n$ Pascal triangle
companion matrix to polynomial x
Sylvester matrix of x and y

powers($x, n, \{c = 1\}$)
dirpowers(n, x)
matrix($m, n, \{i\}, \{j\}, \{expr\}$)
matconcat(B)
matdiagonal(x)
matisdiagonal(x)
matmuldiagonal(x, d)
matid(n)
mathess(x)
mathilbert(n)
matpascal($n - 1$)
matcompanion(x)
polsylvestermatrix(x, y)

Gaussian elimination

kernel of matrix x
intersection of column spaces of x and y
solve $MX = B$ (M invertible)
one sol of $M * X = B$
basis for image of matrix x
columns of x not in **matimage**
supplement columns of x to get basis
rows, cols to extract invertible matrix
rank of the matrix x
solve $MX = B \bmod D$
image mod D
kernel mod D
inverse mod D
determinant mod D

matker($x, \{flag\}$)
matintersect(x, y)
matsolve(M, B)
matinverseimage(M, B)
matimage(x)
matimagecompl(x)
matsupplement(x)
matindexrank(x)
matrank(x)
matsolvemod(M, D, B)
matimagemod(M, D)
matkermod(M, D)
matinvmod(M, D)
matdetmod(M, D)

Lattices & Quadratic Forms

Quadratic forms

evaluate ${}^t x Q y$
evaluate ${}^t x Q x$
signature of quad form ${}^t y * x * y$
decomp into squares of ${}^t y * x * y$
eigenvalues/vectors for real symmetric x
Cholesky decomposition of x

qfeval($\{Q = id\}, x, y$)
qfeval($\{Q = id\}, x$)
qfsign(x)
qfgaussred(x)
qfjacobi(x)
qfcholesky(x)

HNF and SNF

upper triangular Hermite Normal Form
HNF of x where d is a multiple of $\det(x)$
multiple of $\det(x)$
HNF of $(x \mid \text{diagonal}(D))$
elementary divisors of x
 q -rank from elementary divisors
elementary divisors of $\mathbf{Z}[a]/(f'(a))$
integer kernel of x
Z-module \leftrightarrow **Q**-vector space

mathnf(x)
mathnfmod(x, d)
matdetint(x)
mathnfmodid(x, D)
matsnf(x)
snfrank(v, q)
poldiscduced(f)
matkerint(x)
matrixqz(x, p)

Lattices

LLL-algorithm applied to columns of x
... for Gram matrix of lattice
find up to m sols of $qfeval(x, y) \leq b$
... up to m closest vectors to t
 $v, v[i] :=$ number of y s.t. $qfeval(x, y) = i$
perfection rank of x
find isomorphism between q and Q

qflll($x, \{flag\}$)
qflllgram($x, \{flag\}$)
qfminim(x, b, m)
qfcvp(x, t, m)
qfrep($x, B, \{flag\}$)
qfperfection(x)
qfisom(q, Q)

| | |
|--|--|
| precompute for isomorphism test with q | <code>qfisominit(q)</code> |
| automorphism group of q | <code>qfauto(q)</code> |
| convert <code>qfauto</code> for GAP/Magma | <code>qfautoexport($G, \{flag\}$)</code> |
| orbits of V under $G \subset \mathrm{GL}(V)$ | <code>qforbits(G, V)</code> |
| Polynomials & Rational Functions | |
| all defined polynomial variables | <code>variables()</code> |
| get var. of highest priority (higher than v) | <code>varhigher($name, \{v\}$)</code> |
| ... of lowest priority (lower than v) | <code>varlower($name, \{v\}$)</code> |
| Coefficients, variables and basic operators | |
| degree of f | <code>poldegree(f)</code> |
| coef. of degree n of f , leading coef. | <code>polcoef(f, n), pollead</code> |
| main variable / all variables in f | <code>variable(f), variables(f)</code> |
| replace x by y in f | <code>subst(f, x, y)</code> |
| evaluate f replacing vars by their value | <code>eval(f)</code> |
| replace polynomial expr. $T(x)$ by y in f | <code>substpol(f, T, y)</code> |
| replace x_1, \dots, x_n by y_1, \dots, y_n in f | <code>substvec(f, x, y)</code> |
| $f \in A[x]$; reciprocal polynomial $x^{\deg f} f\left(\frac{1}{x}\right)$ | <code>polrecip(f)</code> |
| gcd of coefficients of f | <code>content(f)</code> |
| derivative of f w.r.t. x | <code>deriv($f, \{x\}$)</code> |
| ... n -th derivative of f | <code>derivn($f, n, \{x\}$)</code> |
| formal integral of f w.r.t. x | <code>intformal($f, \{x\}$)</code> |
| formal sum of f w.r.t. x | <code>sumformal($f, \{x\}$)</code> |
| Constructors & Special Polynomials | |
| interpolation polynomial at $(x[1], y[1]), \dots, (x[n], y[n])$, evaluated at t , with error estimate e | <code>polinterpolate($x, \{y\}, \{t\}, \{&e\}$)</code> |
| monic polynomial from roots r | <code>polfromroots(r)</code> |
| $T_n/U_n, H_n$ | <code>polchebyshev(n), polhermite(n)</code> |
| $P_n, L_n^{(\alpha)}$ | <code>pollegendre(n), pollaguerre(n, a)</code> |
| n -th cyclotomic polynomial Φ_n | <code>polcyclo(n)</code> |
| return n if $f = \Phi_n$, else 0 | <code>poliscyclo(f)</code> |
| is f a product of cyclotomic polynomials? | <code>poliscycloprod(f)</code> |
| Zagier's polynomial of index (n, m) | <code>polzagier(n, m)</code> |
| Resultant, elimination | |
| discriminant of polynomial f | <code>poldisc(f)</code> |
| find factors of <code>poldisc(f)</code> | <code>poldiscfactors(f)</code> |
| resultant $R = \mathrm{Res}_v(f, g)$ | <code>polresultant($f, g, \{v\}$)</code> |
| $[u, v, R], xu + yv = \mathrm{Res}_v(f, g)$ | <code>polresultantext($x, y, \{v\}$)</code> |
| solve Thue equation $f(x, y) = a$ | <code>thue($t, a, \{sol\}$)</code> |
| initialize t for Thue equation solver | <code>thueinit(f)</code> |
| Roots and Factorization (Complex/Real) | |
| complex roots of f | <code>polroots(f)</code> |
| bound complex roots of f | <code>polrootsbound(f)</code> |
| number of real roots of f (in $[a, b]$) | <code>polsturm($f, \{[a, b]\}$)</code> |
| real roots of f (in $[a, b]$) | <code>polrootsreal($f, \{[a, b]\}$)</code> |
| complex embeddings of <code>t.POLMOD z</code> | <code>conjvec(z)</code> |
| Roots and Factorization (Finite fields) | |
| factor f mod p , roots | <code>factormod(f, p), polrootsmod</code> |
| factor f over $\mathbf{F}_p[x]/(T)$, roots | <code>factormod($f, [T, p]$), polrootsmod</code> |
| squarefree factorization of f in $\mathbf{F}_q[x]$ | <code>factormodSQF($f, \{D\}$)</code> |
| distinct degree factorization of f in $\mathbf{F}_q[x]$ | <code>factormodDDF($f, \{D\}$)</code> |
| factor n -th cyclotomic pol. Φ_n mod p | <code>factormodcyclo(n, p)</code> |
| Roots and Factorization (p-adic fields) | |
| factor f over \mathbf{Q}_p , roots | <code>factorpadic(f, p, r), polrootspadic</code> |
| p -adic root of f congruent to a mod p | <code>padicappr(f, a)</code> |
| Newton polygon of f for prime p | <code>newtonpoly(f, p)</code> |
| Hensel lift $A/lc(A) = \prod_i B[i] \bmod p^e$ | <code>polhensellift(A, B, p, e)</code> |

| | |
|---|--|
| Pari-GP reference card | |
| (PARI-GP version 2.17.0) | |
| $T = \prod (x - z_i) \mapsto \prod (x - \omega(z_i)) \in \mathbf{Z}_p[x]$ | <code>polteichmuller(T, p, e)</code> |
| extensions of \mathbf{Q}_p of degree N | <code>padicfields(p, N)</code> |
| Roots and Factorization (Miscellaneous) | |
| symmetric powers of roots of f up to n | <code>polsym(f, n)</code> |
| Graeffe transform of $f, g(x^2) = f(x)f(-x)$ | <code>polgraeffe(f)</code> |
| factor f over coefficient field | <code>factor(f)</code> |
| cyclotomic factors of $f \in \mathbf{Q}[X]$ | <code>polcyclofactors(f)</code> |
| Finite Fields | |
| A finite field is encoded by any element (<code>t_FFELT</code>). | |
| find irreducible $T \in \mathbf{F}_p[x]$, $\deg T = n$ | <code>ffinit($p, n, \{x\}$)</code> |
| Create t in $\mathbf{F}_q \simeq \mathbf{F}_p[t]/(T)$ | <code>t = ffgen($T, 't$)</code> |
| ... indirectly, with implicit T | <code>t = ffgen($q, 't$); T = t.mod</code> |
| map m from $\mathbf{F}_q \ni a$ to $\mathbf{F}_{q^k} \ni b$ | <code>m = ffebed(a, b)</code> |
| build $K = \mathbf{F}_q[x]/(P)$ extending $\mathbf{F}_q \ni a$, | <code>ffextend(a, P)</code> |
| evaluate map m on x | <code>ffmap(m, x)</code> |
| inverse map of m | <code>ffinvmap(m)</code> |
| compose maps $m \circ n$ | <code>ffcompomap(m, n)</code> |
| x as polmod over codomain of map m | <code>ffmaprel(m, x)</code> |
| F^n over $\mathbf{F}_q \ni a$ | <code>fffrobenius(a, n)</code> |
| $\#\{\text{monic irred. } T \in \mathbf{F}_q[x], \deg T = n\}$ | <code>ffnbirred(q, n)</code> |
| Formal & p-adic Series | |
| truncate power series or p -adic number | <code>truncate(x)</code> |
| valuation of x at p | <code>valuation(x, p)</code> |
| Dirichlet and Power Series | |
| Taylor expansion around 0 of f w.r.t. x | <code>taylor(f, x)</code> |
| Laurent series of closure F up to x^k | <code>laurentseries(f, k)</code> |
| $\sum a_k b_k t^k$ from $\sum a_k t^k$ and $\sum b_k t^k$ | <code>serconvol(a, b)</code> |
| $f = \sum a_k t^k$ from $\sum (a_k/k!) t^k$ | <code>serlaplace(f)</code> |
| reverse power series F so $F(f(x)) = x$ | <code>serreverse(f)</code> |
| remove terms of degree $< n$ in f | <code>serchop(f, n)</code> |
| Dirichlet series multiplication / division | <code>dirmul, dirdiv(x, y)</code> |
| Dirichlet Euler product (b terms) | <code>direuler($p = a, b, expr$)</code> |
| Transcendental and p-adic Functions | |
| real, imaginary part of x | <code>real(x), imag(x)</code> |
| absolute value, argument of x | <code>abs(x), arg(x)</code> |
| square/ n th root of x | <code>sqrt(x), sqrtsn($x, n, \{&z\}$)</code> |
| all n -th roots of 1 | <code>rootsof1(n)</code> |
| FFT of $[f_0, \dots, f_{n-1}]$ | <code>w = fftinit(n), fft/fftinw(w, f)</code> |
| trig functions | <code>sin, cos, tan, cotan, sinc</code> |
| inverse trig functions | <code>asin, acos, atan</code> |
| hyperbolic functions | <code>sinh, cosh, tanh, cotanh</code> |
| inverse hyperbolic functions | <code>asinh, acosh, atanh</code> |
| $\log(x)$, $\log(1+x)$, e^x , $e^x - 1$ | <code>log, loglp, exp, expm1</code> |
| Euler Γ function, $\log \Gamma$, Γ'/Γ | <code>gamma, lngamma, psi</code> |
| half-integer gamma function $\Gamma(n+1/2)$ | <code>gammah(n)</code> |
| Riemann's zeta $\zeta(s) = \sum n^{-s}$ | <code>zeta(s)</code> |
| $\sum_{1 \leq n \leq N} n^s$ | <code>dirpowerssum(N, s)</code> |
| Hurwitz's $\zeta(s, x) = \sum (n+x)^{-s}$ | <code>zetahurwitz(s, x)</code> |
| Lerch $\Phi(z, s, x) = \sum z^n (n+x)^{-s}$ | <code>lerchphi(z, s, x)</code> |
| Lerch $L(s, x, t) = \Phi(e^{2i\pi t}, s, x)$ | <code>lerchzeta(s, x, t)</code> |
| multiple zeta value (MZV), $\zeta(s_1, \dots, s_k)$ | <code>zetamult($s, \{T\}$)</code> |
| all MZVs for weight $\sum s_i = n$ | <code>zetamultall(n)</code> |
| convert MZV id to $[s_1, \dots, s_k]$ | <code>zetamultconvert($f, \{flag\}$)</code> |
| MZV dual sequence | <code>zetamultdual(s)</code> |
| multiple polylog $Li_{s_1, \dots, s_k}(z_1, \dots, z_k)$ | <code>polylogmult(s, z)</code> |

| | |
|---|--|
| incomplete Γ function ($y = \Gamma(s)$) | <code>incgam($s, x, \{y\}$)</code> |
| complementary incomplete Γ | <code>incgamc(s, x)</code> |
| $\int_x^\infty e^{-t} dt/t$, $(2/\sqrt{\pi}) \int_x^\infty e^{-t^2} dt$ | <code>eint1, erfc</code> |
| elliptic integral of 1st and 2nd kind | <code>ellK(k), ellE(k)</code> |
| dilogarithm of x | <code>dilog(x)</code> |
| m -th polylogarithm of x | <code>polylog($m, x, \{flag\}$)</code> |
| U -confluent hypergeometric function | <code>hyperu(a, b, u)</code> |
| Hypergeometric ${}_pF_q(A, B; z)$ | <code>hypergeom(A, B, z)</code> |
| Bessel $J_n(x)$, $J_{n+1/2}(x)$ | <code>besselj(n, x), besseljh(n, x)</code> |
| Bessel $I_\nu, K_\nu, H_\nu^1, H_\nu^2, Y_\nu$ | <code>(bessel)i, k, h1, h2, y</code> |
| k -th zero of $J_\nu(x)$ | <code>besseljzero($nu, \{k = 1\}$)</code> |
| k -th zero of $Y_\nu(x)$ | <code>besselyzero($nu, \{k = 1\}$)</code> |
| Airy functions $A_i(x)$, $B_i(x)$ | <code>airy(x)</code> |
| Lambert W : x s.t. $xe^x = y$ | <code>lambertw(y)</code> |
| Teichmuller character of p -adic x | <code>teichmuller(x)</code> |
| Iterations, Sums & Products | |
| Numerical integration for meromorphic functions | |
| Behaviour at endpoint for Double Exponential (DE) methods: either a scalar ($a \in \mathbf{C}$, regular) or $\pm\infty$ (decreasing at least as x^{-2}) or | |
| $(x-a)^{-\alpha}$ singularity | <code>[a, α]</code> |
| exponential decrease $e^{-\alpha x }$ | <code>[\pm\infty, \alpha], \alpha > 0</code> |
| slow decrease $ x ^\alpha$ | <code>... \alpha < -1</code> |
| oscillating as $\cos(kx)$ | <code>\alpha = kI, k > 0</code> |
| oscillating as $\sin(kx)$ | <code>\alpha = -kI, k > 0</code> |
| numerical integration | <code>intnum($x = a, b, f, \{T\}$)</code> |
| weights T for intnum | <code>intnuminit($a, b, \{m\}$)</code> |
| weights T incl. kernel K | <code>intfuncinit($t = a, b, K, \{m\}$)</code> |
| integrate $(2i\pi)^{-1} f$ on circle $ z-a = R$ | <code>intcirc($x = a, R, f, \{T\}$)</code> |
| Other integration methods | |
| n -point Gauss-Legendre | <code>intnumgauss($x = a, b, f, \{n\}$)</code> |
| weights for n -point Gauss-Legendre | <code>intnumgaussinit($\{n\}$)</code> |
| quasi-periodic function, period $2H$ | <code>intnumosc($x = a, f, H$)</code> |
| Romberg (low accuracy) | <code>intnumromb($x = a, b, f, \{flag\}$)</code> |
| Numerical summation | |
| sum of series $f(n)$, $n \geq a$ (low accuracy) | <code>suminf($n = a, expr$)</code> |
| sum of alternating/positive series | <code>sumalt, sumpos</code> |
| sum of series using Euler-Maclaurin | <code>sumnum($n = a, f, \{T\}$)</code> |
| ... Sidi summation | <code>sumnumsidi($n = a, f$)</code> |
| $\sum_{n \geq a} F(n)$, F rational function | <code>sumnumrat(F, a)</code> |
| $\dots \sum_{p \geq a} F(p^s)$ | <code>sumeulerrat($F, \{s = 1\}, \{a = 2\}$)</code> |
| weights for sumnum, a as in DE | <code>sumnuminit($\{\infty, a\}$)</code> |
| sum of series by Monien summation | <code>sumnummonien($n = a, f, \{T\}$)</code> |
| weights for sumnummonien | <code>sumnummonieninit($\{\infty, a\}$)</code> |
| sum of series using Abel-Plana | <code>sumnumap($n = a, f, \{T\}$)</code> |
| weights for sumnumap, a as in DE | <code>sumnumapinit($\{\infty, a\}$)</code> |
| sum of series using Lagrange | <code>sumnumlagrange($n = a, f, \{T\}$)</code> |
| weights for sumnumlagrange | <code>sumnumlagrangeinit</code> |
| Products | |
| product $a \leq X \leq b$, initialized at x | <code>prod($X = a, b, expr, \{x\}$)</code> |
| product over primes $a \leq X \leq b$ | <code>prodeuler($X = a, b, expr$)</code> |
| infinite product $a \leq X \leq \infty$ | <code>prodinf($X = a, expr$)</code> |
| $\prod_{n \geq a} F(n)$, F rational function | <code>prodnumrat(F, a)</code> |
| $\prod_{p \geq a} F(p^s)$ | <code>prodeulerrat($F, \{s = 1\}, \{a = 2\}$)</code> |

Other numerical methods

| | |
|---|---|
| real root of f in $[a, b]$; bracketed root | <code>solve($X = a, b, f$)</code> |
| ...interval splitting, step s | <code>solvestep($X = a, b, s, f, \{flag = 0\}$)</code> |
| limit of $f(t)$, $t \rightarrow \infty$ | <code>limitnum($f, \{\alpha\}$)</code> |
| asymptotic expansion of f (rational) | <code>asypnum($f, \{\alpha\}$)</code> |
| ... $N + 1$ terms as floats | <code>asypnumraw($f, N, \{\alpha\}$)</code> |
| numerical derivation w.r.t x : $f'(a)$ | <code>derivnum($x = a, f$)</code> |
| evaluate continued fraction F at t | <code>contfraceval($F, t, \{L\}$)</code> |
| power series to cont. fraction (L terms) | <code>contfracinit($S, \{L\}$)</code> |
| Padé approximant (deg. denom. $\leq B$) | <code>bestapprPade($S, \{B\}$)</code> |

Elementary Arithmetic Functions

| | |
|--|---|
| vector of binary digits of $ x $ | <code>binary(x)</code> |
| bit number n of integer x | <code>bittest(x, n)</code> |
| Hamming weight of integer x | <code>hammingweight(x)</code> |
| digits of integer x in base B | <code>digits($x, \{B = 10\}$)</code> |
| sum of digits of integer x in base B | <code>sumdigits($x, \{B = 10\}$)</code> |
| integer from digits | <code>fromdigits($v, \{B = 10\}$)</code> |
| ceiling/floor/fractional part | <code>ceil, floor, frac</code> |
| round x to nearest integer | <code>round($x, \{\&e\}$)</code> |
| truncate x | <code>truncate($x, \{\&e\}$)</code> |
| gcd/LCM of x and y | <code>gcd(x, y), lcm(x, y)</code> |
| gcd of entries of a vector/matrix | <code>content(x)</code> |

Primes and Factorization

| | |
|--|---|
| extra prime table | <code>addprimes()</code> |
| add primes in v to prime table | <code>addprimes(v)</code> |
| remove primes from prime table | <code>removeprimes(v)</code> |
| Chebyshev $\pi(x)$, n -th prime p_n | <code>primepi(x), prime(n)</code> |
| vector of first n primes | <code>primes(n)</code> |
| smallest prime $\geq x$ | <code>nextprime(x)</code> |
| largest prime $\leq x$ | <code>precprime(x)</code> |
| factorization of x | <code>factor($x, \{lim\}$)</code> |
| ...selecting specific algorithms | <code>factorint($x, \{flag = 0\}$)</code> |
| $n = df^2$, d squarefree/fundamental | <code>core($n, \{fl\}$), coredisc</code> |
| certificate for (prime) N | <code>primecert(N)</code> |
| verifies a certificate c | <code>primecertisvalid(c)</code> |
| convert certificate to Magma/PRIMO | <code>primecertexport</code> |
| recover x from its factorization | <code>factorback($f, \{e\}$)</code> |
| $x \in \mathbf{Z}$, $ x \leq X$, $\gcd(N, P(x)) \geq N$ | <code>zncoppersmith($P, N, X, \{B\}$)</code> |
| divisors of N in residue class r mod s | <code>divisorslensstra(N, r, s)</code> |

Divisors and multiplicative functions

| | |
|--|---|
| number of prime divisors $\omega(n)$ / $\Omega(n)$ | <code>omega(n), bigomega</code> |
| divisors of n / number of divisors $\tau(n)$ | <code>divisors(n), numdiv</code> |
| sum of (k -th powers of) divisors of n | <code>sigma($n, \{k\}$)</code> |
| Möbius μ -function | <code>moebius(x)</code> |
| Ramanujan's τ -function | <code>ramanujantau(x)</code> |

Combinatorics

| | |
|---|---|
| factorial of x | <code>x!</code> or <code>factorial(x)</code> |
| binomial coefficient $\binom{x}{k}$ | <code>binomial($x, \{k\}$)</code> |
| Bernoulli number B_n as real/rational | <code>bernreal(n), bernfrac</code> |
| $[B_0, B_2, \dots B_{2k}]$ | <code>bernvec(k)</code> |
| Bernoulli polynomial $B_n(x)$ | <code>bernpol($n, \{x\}$)</code> |
| Euler numbers | <code>eulerfrac, eulerreal, eulervec</code> |
| Euler polynomial $E_n(x)$ | <code>eulerpol($n, \{x\}$)</code> |
| Eulerian polynomial $A_n(x)$ | <code>eulerianpol</code> |
| Fibonacci number F_n | <code>fibonacci(n)</code> |
| Harmonic number $H_{n,r} = 1^{-r} + \dots + n^{-r}$ | <code>harmonic(n, r)</code> |
| Stirling numbers $s(n, k)$ and $S(n, k)$ | <code>stirling($n, k, \{flag\}$)</code> |

Pari-GP reference card

(PARI-GP version 2.17.0)

| | |
|---|---|
| number of partitions of n | <code>numbpart(n)</code> |
| k -th permutation on n letters | <code>numtoperm(n, k)</code> |
| ...index k of permutation v | <code>permtotnum(v)</code> |
| order of permutation p | <code>permorder(p)</code> |
| signature of permutation p | <code>permsign(p)</code> |
| cyclic decomposition of permutation p | <code>permcycles(p)</code> |

Multiplicative groups $(\mathbf{Z}/N\mathbf{Z})^*$, \mathbf{F}_q^*

| | |
|---|--|
| Euler ϕ -function | <code>eulerphi(x)</code> |
| multiplicative order of x (divides ϕ) | <code>znorder($x, \{o\}$), fforder</code> |
| primitive root mod q / x .mod | <code>znprimroot(q), ffpriroot(x)</code> |
| structure of $(\mathbf{Z}/n\mathbf{Z})^*$ | <code>znstar(n)</code> |
| discrete logarithm of x in base g | <code>znlog($x, g, \{o\}$), ffflog</code> |
| Kronecker-Legendre symbol $(\frac{x}{y})$ | <code>kronecker(x, y)</code> |
| quadratic Hilbert symbol (at p) | <code>hilbert($x, y, \{p\}$)</code> |

Euclidean algorithm, continued fractions

| | |
|--|---|
| CRT: solve $z \equiv x$ and $z \equiv y$ | <code>chinese(x, y)</code> |
| minimal u, v so $xu + yv = \gcd(x, y)$ | <code>gcdext(x, y)</code> |
| half-gcd algorithm | <code>halfgcd(x, y)</code> |
| continued fraction of x | <code>confrac($x, \{b\}, \{lmax\}$)</code> |
| last convergent of continued fraction x | <code>confracpnqn(x)</code> |
| rational approximation to x (den. $\leq B$) | <code>bestappr($x, \{B\}$)</code> |
| recognize $x \in \mathbf{C}$ as polmod mod $T \in \mathbf{Z}[X]$ | <code>bestapprnf(x, T)</code> |

Miscellaneous

| | |
|---|---|
| integer square / n -th root of x | <code>sqrtnint(x, n)</code> |
| largest integer e s.t. $b^e \leq x$, $e = \lfloor \log_b(x) \rfloor$ | <code>logint($x, b, \{\&z\}$)</code> |

Characters

Let $\chi = [d_1, \dots, d_k]$ represent an abelian group $G = \oplus (\mathbf{Z}/d_j\mathbf{Z}) \cdot g_j$ or any structure G affording a `.cyc` method; e.g. `znstar($q, 1$)` for Dirichlet characters. A character χ is coded by $[c_1, \dots, c_k]$ such that $\chi(g_j) = e(n_j/d_j)$.
 $\chi \cdot \psi$; χ^{-1} ; $\chi \cdot \psi^{-1}$; χ^k `charmul, charconj, chardiv, charpow`
order of χ `charorder(cyc, χ)`
kernel of χ `charker(cyc, χ)`
 $\chi(x)$, G a GP group structure `chareval($G, \chi, x, \{z\}$)`
Galois orbits of characters `chargalois(G)`

Dirichlet Characters

initialize $G = (\mathbf{Z}/q\mathbf{Z})^*$ `G = znstar($q, 1$)`
convert datum D to $[G, \chi]$ `znchar(D)`
is χ odd? `zncharisodd(G, χ)`
real $\chi \rightarrow$ Kronecker symbol (D/\cdot) `znchartokronecker(G, χ)`
conductor of χ `zncharconductor(G, χ)`
 $[G_0, \chi_0]$ primitive attached to χ `znchartoprimitive(G, χ)`
induce $\chi \in \hat{G}$ to $\mathbf{Z}/N\mathbf{Z}$ `zncharinduce(G, χ, N)`
 χp `znchardecompose(G, χ, p)`
 $\prod_p |(Q, N) \chi p$ `znchardecompose(G, χ, Q)`
complex Gauss sum $G_a(\chi)$ `znchargauss(G, χ)`

Conrey labelling

Conrey label $m \in (\mathbf{Z}/q\mathbf{Z})^* \rightarrow$ character `znconreychar(G, m)`
character \rightarrow Conrey label `znconreyexp(G, χ)`
log on Conrey generators `znconreylog(G, m)`
conductor of χ (χ_0 primitive) `znconreyconductor($G, \chi, \{\chi_0\}$)`

True-False Tests

| | |
|--|--|
| is x the disc. of a quadratic field? | <code>isfundamental(x)</code> |
| is x a prime? | <code>isprime(x)</code> |
| is x a strong pseudo-prime? | <code>ispseudoprime(x)</code> |
| is x square-free? | <code>issquarefree(x)</code> |
| is x a square? | <code>issquare($x, \{\&n\}$)</code> |
| is x a perfect power? | <code>ispower($x, \{k\}, \{\&n\}$)</code> |
| is x a perfect power of a prime? ($x = p^n$) | <code>isprimepower($x, \&n$)</code> |
| ... of a pseudoprime? | <code>ispseudoprimepower($x, \&n$)</code> |
| is x powerful? | <code>ispowerful(x)</code> |
| is x a totient? ($x = \varphi(n)$) | <code>istotient($x, \{\&n\}$)</code> |
| is x a polygonal number? ($x = P(s, n)$) | <code>ispolygonal($x, s, \{\&n\}$)</code> |
| is pol irreducible? | <code>polisirreducible(pol)</code> |

Graphic Functions

crude graph of $expr$ between a and b `plot($X = a, b, expr$)`
High-resolution plot (immediate plot)
plot $expr$ between a and b `ploth($X = a, b, expr, \{flag\}, \{n\}$)`
plot points given by lists lx, ly `plothraw($lx, ly, \{flag\}$)`
terminal dimensions `plotsizes()`
Rectwindow functions
init window w , with size x, y `plotinit(w, x, y)`
erase window w `plotkill(w)`
copy w to w_2 with offset (dx, dy) `plotcopy(w, w_2, dx, dy)`
scale contents of w `plotclip(w)`
scale coordinates in w `plotscale(w, x_1, x_2, y_1, y_2)`
`ploth` in w `plotrecth($w, X = a, b, expr, \{flag\}, \{n\}$)`
`plothraw` in w `plotrecthraw($w, data, \{flag\}$)`
draw window w_1 at $(x_1, y_1), \dots$ `plotdraw($[[w_1, x_1, y_1], \dots]$)`

Low-level Rectwindow Functions

set current drawing color in w to c `plotcolor(w, c)`
current position of cursor in w `plotcursor(w)`
write s at cursor's position `plotstring(w, s)`
move cursor to (x, y) `plotmove(w, x, y)`
move cursor to $(x + dx, y + dy)$ `plotrmove(w, dx, dy)`
draw a box to (x_2, y_2) `plotbox(w, x_2, y_2)`
draw a box to $(x + dx, y + dy)$ `plotrbox(w, dx, dy)`
draw polygon `plotlines($w, lx, ly, \{flag\}$)`
draw points `plotpoints(w, lx, ly)`
draw ellipse `plotarc($w, lx, ly, \{flag\}$)`
draw line to $(x + dx, y + dy)$ `plotrline(w, dx, dy)`
draw point $(x + dx, y + dy)$ `plotrpoint(w, dx, dy)`

Convert to Postscript or Scalable Vector Graphics

The format f is either "ps" or "svg".
as `ploth` `plotlexport($f, X = a, b, expr, \{flag\}, \{n\}$)`
as `plothraw` `plotlrawexport($f, lx, ly, \{flag\}$)`
as `plotdraw` `plotexport($f, [[w_1, x_1, y_1], \dots]$)`