

Pari-GP reference card

(PARI-GP version 2.11.0)

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 |

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_QFI / t_QFR . Imag/Real binary quad. form | Qfb($a, b, c, \{d\}$) |
| t_VEC / t_COL . Row/Column Vectors | $[x, y, z]$, $[x, y, z] \sim$ |
| 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

| | |
|--|--------------------|
| $\pi = 3.14\dots$, $\gamma = 0.57\dots$, $C = 0.91\dots$ | Pi, Euler, Catalan |
| square root of -1 | I |
| Landau's big-oh notation | O |

Information about an Object

| | |
|---------------------------------------|--------------------------------|
| 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 |
| p -adic, series prec. of x | padicprec(x), serprec |

Operators

| | |
|--|---|
| basic operations | +, -, *, /, ^, sqr |
| $i=i+1$, $i=i-1$, $i=i*j$, ... | i++, i--, i*=j, ... |
| eulerian quotient, remainder | $x \backslash y$, $x \backslash y$, $x \backslash y$, 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, bitneginv |
| maximum/minimum of x and y | max, min(x, y) |
| sign of $x = -1, 0, 1$ | sign(x) |
| binary exponent of x | exponent(x) |
| derivative of 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

| | |
|---|-------------------------------|
| 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 |

Random Numbers

| | |
|----------------------------------|----------------------------|
| random integer/prime in $[0, N[$ | random(N), randomprime |
| 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 PARI object ($x \bmod y$) | Mod(x, y) |
| make x a polynomial of v | Pol($x, \{v\}$) |
| as Pol, etc., starting with constant term | Polrev, Vecrev, Colrev |
| make x a power series of v | Ser($x, \{v\}$) |
| string from bytes / from format+args | Strchr, Strprintf |
| TeX string | Strtex(x) |
| 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) |

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) | lifttpol |

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) |
| 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($M, k, \{\&v\}$) |
| 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 a help message to f | addhelp(f) |
| 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 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 | forpart($p = n, seq$) |
| ...permutations of S | forperm(S, p, seq) |
| ...subsets of $\{1, \dots, n\}$ | forsubset(n, p, seq) |
| ... k -subsets of $\{1, \dots, n\}$ | forsubset($[n, k], p, seq$) |
| ...vectors v , $q(v) \leq B$; $q > 0$ | forqfvec(v, q, b, seq) |
| ... $H < G$ finite abelian group | forsubgroup($H = G$) |

| | |
|---|-------------------|
| evaluate seq until $a \neq 0$ | until(a, seq) |
| while $a \neq 0$, evaluate seq | while(a, seq) |
| exit n innermost enclosing loops | break($\{n\}$) |
| start new iteration of n -th enclosing loop | next($\{n\}$) |
| return x from current subroutine | return($\{x\}$) |

Exceptions, warnings

raise an exception / warn `error(), warning()`
type of error message `E` `errname(E)`
try seq_1 , evaluate seq_2 on error `iferr(seq1, E, seq2)`

Functions with closure arguments / results

select from v according to f `select(f, v)`
apply f to all entries in v `apply(f, v)`
evaluate $f(a_1, \dots, a_n)$ `call(f, a)`
evaluate $f(\dots f(f(a_1, a_2), a_3) \dots, a_n)$ `fold(f, a)`
calling function as closure `self()`

Sums & Products

sum $X = a$ to $X = b$, initialized at x `sum(X = a, b, expr, {x})`
sum entries of vector v `vecsum(v)`
product of all vector entries `vecprod(v)`
sum $expr$ over divisors of n `sumdiv(n, X, expr)`
... assuming $expr$ multiplicative `sumdivmult(n, X, expr)`
product $a \leq X \leq b$, initialized at x `prod(X = a, b, expr, {x})`
product over primes $a \leq X \leq b$ `prodeuler(X = a, b, expr)`

Sorting

sort x by k -th component `vecsrt(x, {k}, {fl = 0})`
min. m of x ($m = x[i]$), max. `vecmin(x, {&i}), vecmax`
does y belong to x , sorted wrt. f `vecsearch(x, y, {f})`

Input/Output

print with/without $\backslash n$, \TeX format `print, print1, printtex`
pretty print matrix `printp`
print fields with separator `printsep(sep, ...), printsep1`
formatted printing `printf()`
write $args$ to file `write, write1, writetex(file, args)`
write x in binary format `writebin(file, x)`
read file into GP `read({file})`
... return as vector of lines `readvec({file})`
... return as vector of strings `readstr({file})`
read a string from keyboard `input()`

Files and file descriptors

File descriptors allows 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 $mode$ `fileopen(path, mode)`
... from shell cmd output (pipe) `fileextern(cmd)`

close descriptor `fileclose(n)`
commit pending write operations `fileflush(n)`
read logical line from file `fileread(n)`
... raw line from file `filereadstr(n)`
write $s \backslash n$ to file `filewrite(n, s)`
... write s to file `filewrite1(n, s)`

Timers

CPU time in ms and reset timer `gettime()`
CPU time in ms since gp startup `getabstime()`
time in ms since UNIX Epoch `getwalltime()`
timeout command after s seconds `alarm(s, expr)`

Interface with system

allocates a new stack of s bytes `allocatemem({s})`
alias old to new `alias(new, old)`
install function from library `install(f, code, {gpf}, {lib})`
execute system command a `system(a)`
... and feed result to GP `extern(a)`
... returning GP string `externstr(a)`

Pari-GP reference card

(PARI-GP version 2.11.0)

get $\$VAR$ from environment `getenv("VAR")`
expand env. variable in string `Strexpend(x)`

Parallel evaluation

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

evaluate f on $x[1], \dots, x[n]$ `parapply(f, x)`
evaluate closures $f[1], \dots, f[n]$ `pareval(f)`
as `select` `parselect(f, A, {flag})`
as `sum` `parsum(i = a, b, expr, {x})`
as `vector` `parvector(n, i, {expr})`
eval f for $i = a, \dots, b$ `parfor(i = a, {b}, f, {r}, {f2})`
... for p prime in $[a, b]$ `parforprime(p = a, {b}, f, {r}, {f2})`
... multivariate `parforvec(X = v, f, {r}, {f2}, {flag})`
declare x as inline (allows to use as global) `inline(x)`
stop inlining `uninline()`

Linear Algebra

dimensions of matrix x `matsize(x)`
multiply two matrices $x * y$
... assuming result is diagonal `matmultodiagonal(x, y)`
concatenation of x and y `concat(x, {y})`
extract components of x `vecextract(x, y, {z})`
transpose of vector or matrix x `mattranspose(x)` or $x \sim$
adjoint of the matrix x `matadjoint(x)`
eigenvectors/values of matrix x `mateigen(x)`
characteristic/minimal polynomial of x `charpoly(x), minpoly`
trace/determinant of matrix x `trace(x), matdet`
permanent of matrix x `matpermanent(x)`
Frobenius form of x `matfrobenius(x)`
QR decomposition `matqr(x)`
apply `matqr`'s transform to v `mathouseholder(Q, v)`

Constructors & Special Matrices

$\{g(x): x \in v \text{ s.t. } f(x)\}$ `[g(x) | x <- v, f(x)]`
 $\{x: x \in v \text{ s.t. } f(x)\}$ `[x | x <- v, f(x)]`
 $\{g(x): x \in v\}$ `[g(x) | x <- v]`
row vec. of $expr$ eval'ed at $1 \leq i \leq n$ `vector(n, {i}, {expr})`
col. vec. of $expr$ eval'ed at $1 \leq i \leq n$ `vectorv(n, {i}, {expr})`
vector of small ints `vectorsmall(n, {i}, {expr})`
 $[c, c \cdot x, \dots, c \cdot x^n]$ `powers(x, n, {c = 1})`
matrix $1 \leq i \leq m, 1 \leq j \leq n$ `matrix(m, n, {i}, {j}, {expr})`
define matrix by blocks `matconcat(B)`
diagonal matrix with diagonal x `matdiagonal(x)`
is x diagonal? `matisdiagonal(x)`
 $x \cdot \text{matdiagonal}(d)$ `matmuldiagonal(x, d)`
 $n \times n$ identity matrix `matid(n)`
Hessenberg form of square matrix x `mathess(x)`
 $n \times n$ Hilbert matrix $H_{ij} = (i + j - 1)^{-1}$ `mathilbert(n)`
 $n \times n$ Pascal triangle `matpascal(n - 1)`
companion matrix to polynomial x `matcompanion(x)`
Sylvester matrix of x `polsylvestermatrix(x)`

Gaussian elimination

kernel of matrix x `matker(x, {flag})`
intersection of column spaces of x and y `matintersect(x, y)`
solve $MX = B$ (M invertible) `matsolve(M, B)`
one sol of $M * X = B$ `matinverseimage(M, B)`
basis for image of matrix x `matimage(x)`
columns of x *not* in `matimage` `matimagecompl(x)`
supplement columns of x to get basis `mat supplement(x)`
rows, cols to extract invertible matrix `matindexrank(x)`
rank of the matrix x `matrank(x)`
solve $MX = B \bmod D$ `matsolvemod(M, D, B)`
image mod D `matimagemod(M, D)`
kernel mod D `matkermod(M, D)`
inverse mod D `matinvmod(M, D)`
determinant mod D `matdetmod(M, D)`

Lattices & Quadratic Forms

Quadratic forms

evaluate ${}^t x Q y$ `qfeval({Q = id}, x, y)`
evaluate ${}^t x Q x$ `qfeval({Q = id}, x)`
signature of quad form ${}^t y * x * y$ `qfsign(x)`
decomp into squares of ${}^t y * x * y$ `qfgaussred(x)`
eigenvalues/vectors for real symmetric x `qfjacobi(x)`

HNF and SNF

upper triangular Hermite Normal Form `mathnf(x)`
HNF of x where d is a multiple of $\det(x)$ `mathnfmod(x, d)`
multiple of $\det(x)$ `matdetint(x)`
HNF of $(x \mid \text{diagonal}(D))$ `mathnfmodid(x, D)`
elementary divisors of x `matsnf(x)`
elementary divisors of $\mathbf{Z}[a]/(f'(a))$ `poldiscreduced(f)`
integer kernel of x `matkerint(x)`
 \mathbf{Z} -module $\leftrightarrow \mathbf{Q}$ -vector space `matrixqz(x, p)`

Lattices

LLL-algorithm applied to columns of x `qflll(x, {flag})`
... for Gram matrix of lattice `qflllgram(x, {flag})`
find up to m sols of $\text{qfnorm}(x, y) \leq b$ `qfminim(x, b, m)`
 $v, v[i] :=$ number of y s.t. $\text{qfnorm}(x, y) = i$ `qfrep(x, B, {flag})`
perfection rank of x `qfperfection(x)`
find isomorphism between q and Q `qfisom(q, Q)`
precompute for isomorphism test with q `qfisominit(q)`
automorphism group of q `qfauto(q)`
convert `qfauto` for GAP/Magma `qfautoexport(G, {flag})`
orbits of V under $G \subset \text{GL}(V)$ `qforbits(G, V)`

Polynomials & Rational Functions

all defined polynomial variables `variables()`
get var. of highest priority (higher than v) `varhigher(name, {v})`
... of lowest priority (lower than v) `varlower(name, {v})`

Pari-GP reference card

(PARI-GP version 2.11.0)

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> |
| reciprocal polynomial $x^{\deg f} f(1/x)$ | <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> |
| 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

| | |
|---|--|
| interpolating pol. eval. at a | <code>polinterpolate(X,{Y},{a})</code> |
| $P_n, T_n/U_n, H_n$ | <code>pollegendre, polchebyshev, polhermite</code> |
| n -th cyclotomic polynomial Φ_n | <code>polcyclo(n,{v})</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 $R = \text{Res}_v(f,g)$ | <code>polresultant(f,g,{v})</code> |
| $[u,v,R], xu + yv = \text{Res}_v(f,g)$ | <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 $f \bmod p$, 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> |

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 $a \bmod p$ | <code>padicappr(f,a)</code> |
| Newton polygon of f for prime p | <code>newtonpoly(f,p)</code> |
| Hensel lift $A/\text{lc}(A) = \prod_i B[i] \bmod p^e$ | <code>polhensellift(A,B,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 (`t_FFELT`).

| | |
|---|---|
| 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 = ffembed(a,b)</code> |
| build K from $\mathbf{F}_q[x]/(P)$ 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> |
| 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 expansion around 0 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/nth root of x | <code>sqrt(x), sqrtn(x,n,{&z})</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, log1p, exp, expm1</code> |
| Euler Γ function, $\log \Gamma, \Gamma'/\Gamma$ | <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> |
| Hurwitz's $\zeta(s,x) = \sum (n+x)^{-s}$ | <code>zetahurwitz(s,x)</code> |
| multiple zeta value (MZV), $\zeta(s_1, \dots, s_k)$ | <code>zetamult(s,{T})</code> |
| ... init T for MZV with $s_1 + \dots + s_k \leq w$ | <code>zetamultinit(w)</code> |
| all MZVs for all weights $\sum s_i \leq n$ | <code>zetamultall(n)</code> |
| convert MZV id to $[s_1, \dots, s_k]$ | <code>zetamultconvert(f,{flag})</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> |
| 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> |
| Bessel $J_n(x), J_{n+1/2}(x)$ | <code>besselj(n,x), besseljh(n,x)</code> |
| Bessel $I_\nu, K_\nu, H_\nu^1, H_\nu^2, N_\nu$ | <code>(bessel)i,k,h1,h2,n</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,a]</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>$\alpha = k\mathbf{I}, k > 0$</code> |
| oscillating as $\sin(kx)$ | <code>$\alpha = -k\mathbf{I}, k > 0$</code> |
| numerical integration | <code>intnum($x=a,b,f,{T}$)</code> |
| weights T for intnum | <code>intnuminit($a,b,K,{m}$)</code> |
| weights T incl. kernel K | <code>intfuncinit($a,b,K,{m}$)</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> |
| Romberg integration (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> |
| $\sum_{n \geq a} F(n)$, F rational function | <code>sumnumrat(F,a)</code> |
| $\dots \sum_{n \geq a} (-1)^n F(n)$ | <code>sumaltrat(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>prodinfn($X=a,expr$)</code> |
| $\prod_{n \geq a} F(n)$, F rational function | <code>prodnumrat(F,a)</code> |
| $\dots \prod_{p \geq a} F(p^s)$ | <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> |
| ... by interval splitting | <code>solvestep($X=a,b,f,{flag=0}$)</code> |
| limit of $f(t)$, $t \rightarrow \infty$ | <code>limitnum(f,{k},{alpha})</code> |
| asymptotic expansion of f at ∞ | <code>asypnum(f,{k},{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>confracinit($S,{L}$)</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>preprime(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 $r \bmod s$ | <code>divisorslenstra(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>$x!$ or 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> |
| Bernoulli polynomial $B_n(x)$ | <code>bernpol($n, \{x\}$)</code> |
| n -th Fibonacci number | <code>fibonacci(n)</code> |
| Stirling numbers $s(n, k)$ and $S(n, k)$ | <code>stirling($n, k, \{flag\}$)</code> |
| number of partitions of n | <code>numbpart(n)</code> |
| k -th permutation on n letters | <code>numtoperm(n, k)</code> |
| convert permutation to (n, k) form | <code>permtotnum(v)</code> |
| order of permutation p | <code>permorder(p)</code> |
| signature of permutation p | <code>permsign(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), ffprimroot(x)</code> |
| structure of $(\mathbf{Z}/n\mathbf{Z})^*$ | <code>znstar(n)</code> |
| discrete logarithm of x in base g | <code>znlog($x, g, \{o\}$), fflag</code> |
| Kronecker-Legendre symbol $(\frac{x}{y})$ | <code>kronecker(x, y)</code> |
| quadratic Hilbert symbol (at p) | <code>hilbert($x, y, \{p\}$)</code> |

| | |
|---|--|
| Miscellaneous | |
| integer square / n -th root of x | <code>sqrntint(x), sqrtnint(x, n)</code> |
| largest integer e s.t. $b^e \leq b$, $e = \lfloor \log_b(x) \rfloor$ | <code>logint($x, b, \{\&z\}$)</code> |
| 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> |
| continued fraction of x | <code>contfrac($x, \{b\}, \{lmax\}$)</code> |
| last convergent of continued fraction x | <code>contfracpnqn(x)</code> |
| rational approximation to x (den. $\leq B$) | <code>bestappr($x, \{B\}k$)</code> |
| recognize $x \in \mathbf{C}$ as polmod mod $T \in \mathbf{Z}[X]$ | <code>bestapprnf(x, T)</code> |

Characters

| | |
|--|---|
| Let $cyc = [d_1, \dots, d_k]$ 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. <code>znstar($q, 1$)</code> 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 | <code>charmul, charconj, chardiv,, charpow</code> |
| order of χ | <code>charorder(cyc, χ)</code> |
| kernel of χ | <code>charker(cyc, χ)</code> |
| $\chi(x)$, G a GP group structure | <code>chareval($G, \chi, x, \{z\}$)</code> |
| Galois orbits of characters | <code>chargalois(G)</code> |

Dirichlet Characters

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

Conrey labelling

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

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 | <code>plot($X = a, b, expr$)</code> |
| High-resolution plot (immediate plot) | |
| plot $expr$ between a and b | <code>plotth($X = a, b, expr, \{flag\}, \{n\}$)</code> |
| plot points given by lists lx, ly | <code>plotthraw($lx, ly, \{flag\}$)</code> |
| terminal dimensions | <code>plotsizes()</code> |

Rectwindow functions

| | |
|--|---|
| init window w , with size x, y | <code>plotinit(w, x, y)</code> |
| erase window w | <code>plotkill(w)</code> |
| copy w to w_2 with offset (dx, dy) | <code>plotcopy(w, w_2, dx, dy)</code> |
| clips contents of w | <code>plotclip(w)</code> |
| scale coordinates in w | <code>plotscale(w, x_1, x_2, y_1, y_2)</code> |
| plotth in w | <code>plotrecth($w, X = a, b, expr, \{flag\}, \{n\}$)</code> |
| plotthraw in w | <code>plotrecthraw($w, data, \{flag\}$)</code> |
| draw window w_1 at $(x_1, y_1), \dots$ | <code>plotdraw($[[w_1, x_1, y_1], \dots]$)</code> |

Low-level Rectwindow Functions

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

Convert to Postscript or Scalable Vector Graphics

| | |
|---|--|
| The format f is either "ps" or "svg". | |
| as plotth | <code>plotthexport($f, X = a, b, expr, \{flag\}, \{n\}$)</code> |
| as plotthraw | <code>plotthrawexport($f, lx, ly, \{flag\}$)</code> |
| as plotdraw | <code>plotexport($f, [[w_1, x_1, y_1], \dots]$)</code> |

Based on an earlier version by Joseph H. Silverman
July 2018 v2.35. Copyright © 2018 K. Belabas
Permission is granted to make and distribute copies of this card provided the copyright and this permission notice are preserved on all copies.
Send comments and corrections to (Karim.Belabas@math.u-bordeaux.fr)