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; ±0x1F, ±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	$x + y *$ quadgen(D,'w)
t_POLMOD. Polynomials modulo g	Mod(f, g)
t_POL. Polynomials	$a * x$^$n + \cdots + b$
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]~
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

$\pi = 3.14\ldots$, $\gamma = 0.57\ldots$, $C = 0.91\ldots$	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,...
euclidean quotient, remainder	x\/y, x\\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, 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 $a, a+1, \ldots, b$ 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 mod 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 ($\to$t_INT)	liftint
lift all t_POLMOD ($\to$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)
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 = 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 dictionnary: M = 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 opitted;
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 $a \le X \le b$	for($X = a, b, seq$)
...for primes $a \le X \le b$	forprime($X = a, b, seq$)
...for primes $\equiv a \pmod q$	forprimestep($X = a, b, q, seq$)
...for composites $a \le X \le b$	forcomposite($X = a, b, seq$)
...for $a \le X \le b$ stepping s	forstep($X = a, b, s, seq$)
...for X dividing n	fordiv(n, X, seq)
...$X = [n, factor(n)]$, $a \le n \le b$	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 $\{1, \ldots, n\}$	forsubset(n, p, seq)
...k-subsets of $\{1, \ldots, n\}$	forsubset([n, k], p, seq)
...vectors v, $q(v) \le B$; $q > 0$	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 $f(a_1, \ldots, a_n)$	`call(f, a)`
evaluate $f(\ldots f(f(a_1, a_2), a_3) \ldots, a_n)$	`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	`vecsort(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 `\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`\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	`Strexpand(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 $x[1], \ldots, x[n]$	`parapply(f, x)`
evaluate closures $f[1], \ldots, f[n]$	`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 $i = a, \ldots, b$	`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~`
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})`
$[c, c \cdot x, \ldots, c \cdot x^n]$	`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)`
$x \cdot$ `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	`matsupplement(x)`
rows, cols to extract invertible matrix	`matindexrank(x)`
rank of the matrix x	`matrank(x)`
solve $MX = B$ mod 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 ${}^t xQy$	`qfeval({Q = id}, x, y)`
evaluate ${}^t xQx$	`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 | `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 `qfnorm(x, y)` $\leq b$	`qfminim(x, b, m)`
v, $v[i]$:=number of y s.t. `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 $G \subset \mathrm{GL}(V)$	`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	poldegree(f)
coef. of degree n of f, leading coef.	polcoef(f,n), pollead
main variable / all variables in f	variable(f), variables(f)
replace x by y in f	subst(f,x,y)
evaluate f replacing vars by their value	eval(f)
replace polynomial expr. $T(x)$ by y in f	substpol(f,T,y)
replace $x_1,\dots,x_n$ by $y_1,\dots,y_n$ in f	substvec(f,x,y)
reciprocal polynomial $x^{\deg f} f(1/x)$	polrecip(f)
gcd of coefficients of f	content(f)
derivative of f w.r.t. x	deriv($f,\{x\}$)
formal integral of f w.r.t. x	intformal($f,\{x\}$)
formal sum of f w.r.t. x	sumformal($f,\{x\}$)

Constructors & Special Polynomials

interpolating pol. eval. at a	polinterpolate($X,\{Y\},\{a\}$)
P_n, T_n/U_n, H_n	pollegendre, polchebyshev, polhermite
n-th cyclotomic polynomial Φ_n	polcyclo($n,\{v\}$)
return n if $f=\Phi_n$, else 0	poliscyclo(f)
is f a product of cyclotomic polynomials?	poliscycloprod(f)
Zagier's polynomial of index (n,m)	polzagier(n,m)

Resultant, elimination

discriminant of polynomial f	poldisc(f)
find factors of poldisc(f)	poldiscfactors(f)
resultant $R = \mathrm{Res}_v(f,g)$	polresultant($f,g,\{v\}$)
$[u,v,R]$, $xu+yv=\mathrm{Res}_v(f,g)$	polresultantext($x,y,\{v\}$)
solve Thue equation $f(x,y)=a$	thue($t,a,\{sol\}$)
initialize t for Thue equation solver	thueinit(f)

Roots and Factorization (Complex/Real)

complex roots of f	polroots(f)
bound complex roots of f	polrootsbound(f)
number of real roots of f (in $[a,b]$)	polsturm($f,\{[a,b]\}$)
real roots of f (in $[a,b]$)	polrootsreal($f,\{[a,b]\}$)
complex embeddings of t_POLMOD z	conjvec(z)

Roots and Factorization (Finite fields)

factor f mod p, roots	factormod(f,p), polrootsmod
factor f over $\mathbf{F}_p[x]/(T)$, roots	factormod($f,[T,p]$), polrootsmod
squarefree factorization of f in $\mathbf{F}_q[x]$	factormodSQF($f,\{D\}$)
distinct degree factorization of f in $\mathbf{F}_q[x]$	factormodDDF($f,\{D\}$)

Roots and Factorization (p-adic fields)

factor f over $\mathbf{Q}_p$, roots	factorpadic(f,p,r), polrootspadic
p-adic root of f congruent to a mod p	padicappr(f,a)
Newton polygon of f for prime p	newtonpoly(f,p)
Hensel lift $A/\mathrm{lc}(A)=\prod_i B[i]$ mod p^e	polhensellift(A,B,p,e)
extensions of $\mathbf{Q}_p$ of degree N	padicfields(p,N)

Roots and Factorization (Miscellaneous)

symmetric powers of roots of f up to n	polsym(f,n)
Graeffe transform of f, $g(x^2)=f(x)f(-x)$	polgraeffe(f)
factor f over coefficient field	factor(f)
cyclotomic factors of $f\in\mathbf{Q}[X]$	polcyclofactors(f)

Finite Fields

A finite field is encoded by any element (t_FFELT).

find irreducible $T\in\mathbf{F}_p[x]$, $\deg T = n$	ffinit($p,n,\{x\}$)
Create t in $\mathbf{F}_q\simeq\mathbf{F}_p[t]/(T)$	t = ffgen(T,'t)
...indirectly, with implicit T	t = ffgen(q,'t); T = t.mod
map m from $\mathbf{F}_q\ni a$ to $\mathbf{F}_{q^k}\ni b$	m = ffembed(a,b)
build $K=\mathbf{F}_q[x]/(P)$ extending $\mathbf{F}_q\ni a$,	ffextend(a,P)
evaluate map m on x	ffmap(m,x)
inverse map of m	ffinvmap(m)
compose maps $m\circ n$	ffcompomap(m,n)
F^n over $\mathbf{F}_q\ni a$	fffrobenius(a,n)
$\#\{$monic irred. $T\in\mathbf{F}_q[x], \deg T = n\}$	ffnbirred(q,n)

Formal & p-adic Series

truncate power series or p-adic number	truncate(x)
valuation of x at p	valuation(x,p)

Dirichlet and Power Series

Taylor expansion around 0 of f w.r.t. x	taylor(f,x)
Laurent series expansion around 0 up to x^k	laurentseries(f,k)
$\sum a_k b_k t^k$ from $\sum a_k t^k$ and $\sum b_k t^k$	serconvol(a,b)
$f=\sum a_k t^k$ from $\sum (a_k/k!) t^k$	serlaplace(f)
reverse power series F so $F(f(x))=x$	serreverse(f)
remove terms of degree $<n$ in f	serchop(f,n)
Dirichlet series multiplication / division	dirmul, dirdiv(x,y)
Dirichlet Euler product (b terms)	direuler($p=a,b,expr$)

Transcendental and p-adic Functions

real, imaginary part of x	real(x), imag(x)
absolute value, argument of x	abs(x), arg(x)
square/nth root of x	sqrt(x), sqrtn($x,n,\{\&z\}$)
trig functions	sin, cos, tan, cotan, sinc
inverse trig functions	asin, acos, atan
hyperbolic functions	sinh, cosh, tanh, cotanh
inverse hyperbolic functions	asinh, acosh, atanh
$\log(x)$, $\log(1+x)$, e^x, e^x-1	log, log1p, exp, expm1
Euler Γ function, $\log\Gamma$, Γ'/Γ	gamma, lngamma, psi
half-integer gamma function $\Gamma(n+1/2)$	gammah(n)
Riemann's zeta $\zeta(s)=\sum n^{-s}$	zeta(s)
Hurwitz's $\zeta(s,x)=\sum(n+x)^{-s}$	zetahurwitz(s,x)
multiple zeta value (MZV), $\zeta(s_1,\dots,s_k)$	zetamult($s,\{T\}$)
... init T for MZV with $s_1+\ldots+s_k\le w$	zetamultinit(w)
all MZVs for all weights $\sum s_i\le n$	zetamultall(n)
convert MZV id to $[s_1,\dots,s_k]$	zetamultconvert($f,\{flag\}$)
incomplete Γ function ($y=\Gamma(s)$)	incgam($s,x,\{y\}$)
complementary incomplete Γ	incgamc(s,x)
$\int_x^\infty e^{-t}\,dt/t$, $\quad (2/\sqrt{\pi})\int_x^\infty e^{-t^2}\,dt$	eint1, erfc
dilogarithm of x	dilog(x)
m-th polylogarithm of x	polylog($m,x,\{flag\}$)
U-confluent hypergeometric function	hyperu(a,b,u)
Bessel $J_n(x)$, $J_{n+1/2}(x)$	besselj(n,x), besseljh(n,x)
Bessel I_ν, K_ν, H^1_ν, H^2_ν, N_ν	(bessel)i, k, h1, h2, n
Lambert W: x s.t. $xe^x=y$	lambertw(y)
Teichmuller character of p-adic x	teichmuller(x)

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 $\pm$oo (decreasing at least as x^{-2}) or

$(x-a)^{-\alpha}$ singularity	$[a,\alpha]$
exponential decrease $e^{-\alpha\lvert x\rvert}$	$[\pm\infty,\alpha]$, $\alpha>0$
slow decrease $\lvert x\rvert^\alpha$	$\ldots\alpha<-1$
oscillating as $\cos(kx)$)	$\alpha=k$I, $k>0$
oscillating as $\sin(kx)$)	$\alpha=-k$I, $k>0$
numerical integration	intnum($x=a,b,f,\{T\}$)
weights T for intnum	intnuminit($a,b,\{m\}$)
weights T incl. kernel K	intfuncinit($a,b,K,\{m\}$)
integrate $(2i\pi)^{-1}f$ on circle $\lvert z-a\rvert=R$	intcirc($x=a,R,f,\{T\}$)

Other integration methods

n-point Gauss-Legendre	intnumgauss($x=a,b,f,\{n\}$)
weights for n-point Gauss-Legendre	intnumgaussinit($\{n\}$)
Romberg integration (low accuracy)	intnumromb($x=a,b,f,\{flag\}$)

Numerical summation

sum of series $f(n)$, $n\ge a$ (low accuracy)	suminf($n=a,expr$)
sum of alternating/positive series	sumalt, sumpos
sum of series using Euler-Maclaurin	sumnum($n=a,f,\{T\}$)
$\sum_{n\ge a}F(n)$, F rational function	sumnumrat(F,a)
$\ldots\sum_{n\ge a}(-1)^n F(n)$	sumaltrat(F,a)
$\ldots\sum_{p\ge a}F(p^s)$	sumeulerrat($F,\{s=1\},\{a=2\}$)
weights for sumnum, a as in DE	sumnuminit($\{\infty,a\}$)
sum of series by Monien summation	sumnummonien($n=a,f,\{T\}$)
weights for sumnummonien	sumnummonieninit($\{\infty,a\}$)
sum of series using Abel-Plana	sumnumap($n=a,f,\{T\}$)
weights for sumnumap, a as in DE	sumnumapinit($\{\infty,a\}$)
sum of series using Lagrange	sumnumlagrange($n=a,f,\{T\}$)
weights for sumnumlagrange	sumnumlagrangeinit

Products

product $a\le X\le b$, initialized at x	prod($X=a,b,expr,\{x\}$)
product over primes $a\le X\le b$	prodeuler($X=a,b,expr$)
infinite product $a\le X\le\infty$	prodinf($X=a,expr$)
$\prod_{n\ge a}F(n)$, F rational function	prodnumrat(F,a)
$\ldots\prod_{p\ge a}F(p^s)$	prodeulerrat($F,\{s=1\},\{a=2\}$)

Other numerical methods

real root of f in $[a,b]$; bracketed root	solve($X=a,b,f$)
...by interval splitting	solvestep($X=a,b,f,\{flag=0\}$)
limit of $f(t)$, $t\to\infty$	limitnum(f, {k}, {alpha})
asymptotic expansion of f at ∞	asympnum(f, {k}, {alpha})
numerical derivation w.r.t x: $f'(a)$	derivnum($x=a,f$)
evaluate continued fraction F at t	contfraceval($F,t,\{L\}$)
power series to cont. fraction (L terms)	contfracinit($S,\{L\}$)
Padé approximant (deg. denom. $\le B$)	bestapprPade($S,\{B\}$)

Elementary Arithmetic Functions

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

Primes and Factorization

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

Divisors and multiplicative functions

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

Combinatorics

factorial of x	`x!` or `factorial`(x)
binomial coefficient $\binom{x}{k}$	`binomial`$(x, \{k\})$
Bernoulli number B_n as real/rational	`bernreal`(n), `bernfrac`
Bernoulli polynomial $B_n(x)$	`bernpol`$(n, \{x\})$
n-th Fibonacci number	`fibonacci`(n)
Stirling numbers $s(n, k)$ and $S(n, k)$	`stirling`$(n, k, \{flag\})$
number of partitions of n	`numbpart`(n)
k-th permutation on n letters	`numtoperm`(n, k)
convert permutation to (n, k) form	`permtonum`(v)
order of permutation p	`permorder`(p)
signature of permutation p	`permsign`(p)

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

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

Miscellaneous

integer square / n-th root of x	`sqrtint`(x), `sqrtnint`(x, n)
largest integer e s.t. $b^e \leq b$, $e = \lfloor \log_b(x) \rfloor$	`logint`$(x, b, \{\&z\})$
CRT: solve $z \equiv x$ and $z \equiv y$	`chinese`(x, y)
minimal u, v so $xu + yv = \gcd(x, y)$	`gcdext`(x, y)
continued fraction of x	`contfrac`$(x, \{b\}, \{lmax\})$
last convergent of continued fraction x	`contfracpnqn`(x)
rational approximation to x (den. $\leq B$)	`bestappr`$(x, \{B\}k)$
recognize $x \in \mathbf{C}$ as polmod mod $T \in \mathbf{Z}[X]$	`bestapprnf`(x, T)

Characters

Let $cyc = [d_1, \ldots, d_k]$ 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 $[c_1, \ldots, c_k]$ 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 $\chi \to$ Kronecker symbol $(D/.)$	`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)
χ_p	`znchardecompose`(G, χ, p)
$\prod_{p \mid (Q, N)} \chi_p$	`znchardecompose`(G, χ, Q)
complex Gauss sum $G_a(\chi)$	`znchargauss`(G, χ)

Conrey labelling

Conrey label $m \in (\mathbf{Z}/q\mathbf{Z})^* \to$ character	`znconreychar`(G, m)
character $\to$ 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?	`isfundamental`(x)
is x a prime?	`isprime`(x)
is x a strong pseudo-prime?	`ispseudoprime`(x)
is x square-free?	`issquarefree`(x)
is x a square?	`issquare`$(x, \{\&n\})$
is x a perfect power?	`ispower`$(x, \{k\}, \{\&n\})$
is x a perfect power of a prime? ($x = p^n$)	`isprimepower`$(x, \&n\})$
...of a pseudoprime?	`ispseudoprimepower`$(x, \&n\})$
is x powerful?	`ispowerful`(x)
is x a totient? ($x = \varphi(n)$)	`istotient`$(x, \{\&n\})$
is x a polygonal number? ($x = P(s, n)$)	`ispolygonal`$(x, s, \{\&n\})$
is pol irreducible?	`polisirreducible`(pol)

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	`plothsizes()`

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)
clips 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), ...	`plotdraw`$([[w_1, x_1, y_1], \ldots])$

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 line to $(x + dx, y + dy)$	`plotrline`(w, dx, dy)
draw point $(x + dx, y + dy)$	`plotrpoint`(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`	`plothexport`$(f, X = a, b, expr, \{flag\}, \{n\})$
as `plothraw`	`plothrawexport`$(f, lx, ly, \{flag\})$
as `plotdraw`	`plotexport`$(f, [[w_1, x_1, y_1], \ldots])$

Send comments and corrections to ⟨Karim.Belabas@math.u-bordeaux.fr⟩