

Pari-GP reference card

(PARI-GP version 2.13.3)

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	? <i>function</i>
extended description	?? <i>keyword</i>
list of relevant help topics	??? <i>pattern</i>
name of GP-1.39 function <i>f</i> in GP-2.*	whatnow(<i>f</i>)

Input/Output

previous result, the result before	%, %`, %``, etc.
<i>n</i> -th result since startup	% <i>n</i>
separate multiple statements on line	;
extend statement on additional lines	\
extend statements on several lines	{ <i>seq</i> ₁ ; <i>seq</i> ₂ ;}
comment	/* ... */
one-line comment, rest of line ignored	\\ ...

Metacommands & Defaults

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

Debugger / break loop

get out of break loop	break or <C-D>
go up/down <i>n</i> frames	dbg_up({ <i>n</i> }), dbg_down
set break point	breakpoint()
examine object <i>o</i>	dbg_x(<i>o</i>)
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 <i>m</i>	Mod(<i>n</i> , <i>m</i>)
t_FRAC. Rational Numbers	<i>n</i> / <i>m</i>
t_FFELT. Elt in finite field F _{<i>q</i>}	ffgen(<i>q</i> , 't)
t_COMPLEX. Complex Numbers	<i>x</i> + <i>y</i> * I
t_PADIC. <i>p</i> -adic Numbers	<i>x</i> + 0(<i>p</i> ^{<i>k</i>})
t_QUAD. Quadratic Numbers	<i>x</i> + <i>y</i> * quadgen(<i>D</i> , {'w'})
t_POLMOD. Polynomials modulo <i>g</i>	Mod(<i>f</i> , <i>g</i>)
t_POL. Polynomials	<i>a</i> * <i>x</i> ^{<i>n</i>} + ... + <i>b</i>
t_SER. Power Series	<i>f</i> + 0(<i>x</i> ^{<i>k</i>})
t_RFRAC. Rational Functions	<i>f</i> / <i>g</i>
t_QFI/t_QFR. Imag/Real binary quad. form	Qfb(<i>a</i> , <i>b</i> , <i>c</i> , { <i>d</i> })
t_VEC/t_COL. Row/Column Vectors	[<i>x</i> , <i>y</i> , <i>z</i>], [<i>x</i> , <i>y</i> , <i>z</i>]~
t_VEC integer range	[1..10]

t_VECSMALL. Vector of small ints	Vecsmall([<i>x</i> , <i>y</i> , <i>z</i>])
t_MAT. Matrices	[<i>a</i> , <i>b</i> ; <i>c</i> , <i>d</i>]
t_LIST. Lists	List([<i>x</i> , <i>y</i> , <i>z</i>])
t_STR. Strings	"abc"
t_INFINITY. ±∞	+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 <i>x</i>	type(<i>x</i>)
length of <i>x</i> / size of <i>x</i> in memory	# <i>x</i> , sizebyte(<i>x</i>)
real precision / bit precision of <i>x</i>	precision(<i>x</i>), bitprecision(<i>x</i>)
<i>p</i> -adic, series prec. of <i>x</i>	padicprec(<i>x</i> , <i>p</i>), serprec(<i>x</i> , <i>v</i>)
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	<i>x</i> \/ <i>y</i> , <i>x</i> \/ <i>y</i> , <i>x</i> % <i>y</i> , divrem(<i>x</i> , <i>y</i>)
shift <i>x</i> left or right <i>n</i> bits	<i>x</i> << <i>n</i> , <i>x</i> >> <i>n</i> or shift(<i>x</i> ,± <i>n</i>)
multiply by 2 ^{<i>n</i>}	shiftmul(<i>x</i> , <i>n</i>)
comparison operators	<=, <, >=, >, ==, !=, ==, lex, cmp
boolean operators (or, and, not)	, &&, !
bit operations	bitand, bitneg, bitor, bitxor, bitnegimply
maximum/minimum of <i>x</i> and <i>y</i>	max(<i>x</i> , <i>y</i>), min(<i>x</i> , <i>y</i>)
sign of <i>x</i> (gives −1, 0, 1)	sign(<i>x</i>)
binary exponent of <i>x</i>	exponent(<i>x</i>)
derivative of <i>f</i> , 2nd derivative, etc.	<i>f</i> ' , <i>f</i> '', ...
differential operator	diffop(<i>f</i> , <i>v</i> , <i>d</i> , { <i>n</i> = 1})
quote operator (formal variable)	'x
assignment	x = <i>value</i>
simultaneous assignment $x \leftarrow v[1]$, $y \leftarrow v[2]$	[x,y] = v

Select Components

Caveat: components start at index *n* = 1.

<i>n</i> -th component of <i>x</i>	component(<i>x</i> , <i>n</i>)
<i>n</i> -th component of vector/list <i>x</i>	<i>x</i> [<i>n</i>]
components <i>a</i> , <i>a</i> + 1, ..., <i>b</i> of vector <i>x</i>	<i>x</i> [<i>a</i> .. <i>b</i>]
(<i>m</i> , <i>n</i>)-th component of matrix <i>x</i>	<i>x</i> [<i>m</i> , <i>n</i>]
row <i>m</i> or column <i>n</i> of matrix <i>x</i>	<i>x</i> [<i>m</i> ,], <i>x</i> [, <i>n</i>]
numerator/denominator of <i>x</i>	numerator(<i>x</i>), denominator(<i>x</i>)

Random Numbers

random integer/prime in [0, <i>N</i> [random(<i>N</i>), randomprime(<i>N</i>)
get/set random seed	getrand, setrand(<i>s</i>)

Conversions

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

Character strings

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

Lists, Sets & Maps

Sets (= row vector with strictly increasing entries w.r.t. cmp)

intersection of sets <i>x</i> and <i>y</i>	setintersect(<i>x</i> , <i>y</i>)
set of elements in <i>x</i> not belonging to <i>y</i>	setminus(<i>x</i> , <i>y</i>)
union of sets <i>x</i> and <i>y</i>	setunion(<i>x</i> , <i>y</i>)
does <i>y</i> belong to the set <i>x</i>	setsearch(<i>x</i> , <i>y</i> , { <i>flag</i> })
set of all <i>f</i> (<i>x</i> , <i>y</i>), $x \in X$, $y \in Y$	setbinop(<i>f</i> , <i>X</i> , <i>Y</i>)
is <i>x</i> a set ?	setisset(<i>x</i>)

Lists. create empty list: *L* = List()

append <i>x</i> to list <i>L</i>	listput(<i>L</i> , <i>x</i> , { <i>i</i> })
remove <i>i</i> -th component from list <i>L</i>	listpop(<i>L</i> , { <i>i</i> })
insert <i>x</i> in list <i>L</i> at position <i>i</i>	listinsert(<i>L</i> , <i>x</i> , <i>i</i>)
sort the list <i>L</i> in place	listsort(<i>L</i> , { <i>flag</i> })

Maps. create empty dictionary: *M* = Map()

attach value <i>v</i> to key <i>k</i>	mapput(<i>M</i> , <i>k</i> , <i>v</i>)
recover value attach to key <i>k</i> or error	mapget(<i>M</i> , <i>k</i>)
is key <i>k</i> in the dict? (set <i>v</i> to <i>M</i> (<i>k</i>))	mapisdefined(<i>M</i> , <i>k</i> , {& <i>v</i> })
remove <i>k</i> from map domain	mapdelete(<i>M</i> , <i>k</i>)

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 <i>h</i> to <i>s</i>	addhelp(<i>s</i> , <i>h</i>)
undefine symbol <i>s</i> (also kills help)	kill(<i>s</i>)
Control Statements (<i>X</i> : formal parameter in expression <i>seq</i>)	
if $a \neq 0$, evaluate <i>seq</i> ₁ , else <i>seq</i> ₂	if(<i>a</i> , { <i>seq</i> ₁ }, { <i>seq</i> ₂ })
eval. <i>seq</i> for $a \leq X \leq b$	for(<i>X</i> = <i>a</i> , <i>b</i> , <i>seq</i>)
...for $X \in v$	foreach(<i>v</i> , <i>X</i> , <i>seq</i>)
...for primes $a \leq X \leq b$	forprime(<i>X</i> = <i>a</i> , <i>b</i> , <i>seq</i>)
...for primes $\equiv a \pmod{q}$	forprimestep(<i>X</i> = <i>a</i> , <i>b</i> , <i>q</i> , <i>seq</i>)
...for composites $a \leq X \leq b$	forcomposite(<i>X</i> = <i>a</i> , <i>b</i> , <i>seq</i>)
...for $a \leq X \leq b$ stepping <i>s</i>	forstep(<i>X</i> = <i>a</i> , <i>b</i> , <i>s</i> , <i>seq</i>)
...for <i>X</i> dividing <i>n</i>	fordiv(<i>n</i> , <i>X</i> , <i>seq</i>)
... $X = [n, factor(n)]$, $a \leq n \leq b$	forfactored(<i>X</i> = <i>a</i> , <i>b</i> , <i>seq</i>)
...as above, <i>n</i> squarefree	forsquarefree(<i>X</i> = <i>a</i> , <i>b</i> , <i>seq</i>)
... $X = [d, factor(d)]$, $d \mid n$	fordivfactored(<i>n</i> , <i>X</i> , <i>seq</i>)
multivariable for, lex ordering	forvec(<i>X</i> = <i>v</i> , <i>seq</i>)

```

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  $\backslash n$ ,  $\text{\TeX}$  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( $[n, k], p, seq$ )
forqfvec( $v, q, b, seq$ )
forsubgroup( $H = G$ )
until( $a, seq$ )
while( $a, seq$ )
break( $\{n\}$ )
next( $\{n\}$ )
return( $\{x\}$ )

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

Results
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( $X = a, b, expr$ )

vecsrt( $x, \{k\}, \{fl = 0\}$ )
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()

fileopen( $path, mode$ )
fileextern( $cmd$ )
fileclose( $n$ )
fileflush( $n$ )
fileread( $n$ )
filereadstr( $n$ )
filewrite( $n, s$ )
filewrite1( $n, s$ )

```

Pari-GP reference card

(PARI-GP version 2.13.3)

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

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 $p = a \bmod q$
... multivariate
export x to parallel world
... all dynamic variables
frees exported value x
... all exported values

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

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'ed at $1 \leq i \leq n$
col. vec. of $expr$ eval'ed at $1 \leq i \leq n$
vector of small ints

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)

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($p = a, \{b\}, q, f, \{r\}, \{f_2\}$)
parforvec($X = v, f, \{r\}, \{f_2\}, \{flag\}$)
export(x)
exportall()
unexport(x)
unexportall()

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

$[g(x) \mid x \leftarrow v, f(x)]$
 $[x \mid x \leftarrow v, f(x)]$
 $[g(x) \mid x \leftarrow v]$
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

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

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

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
elementary divisors of $\mathbf{Z}[a]/(f'(a))$
integer kernel of x
 \mathbf{Z} -module \leftrightarrow \mathbf{Q} -vector space

Lattices

LLL-algorithm applied to columns of x
... for Gram matrix of lattice
find up to m sols of **qfnorm**($x, y) \leq b$
 $v, v[i] :=$ number of y s.t. **qfnorm**($x, y) = i$
perfection rank of x
find isomorphism between q and Q
precompute for isomorphism test with q
automorphism group of 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($name, \{v\}$)`

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)`

$f \in A[x]$; reciprocal polynomial $x^{\deg f} f\left(\frac{1}{x}\right)$ `polrecip(f)`
gcd of coefficients of f `content(f)`
derivative of f w.r.t. x `deriv($f, \{x\}$)`
... n -th derivative of f `derivn($f, n, \{x\}$)`
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

interpolation polynomial at $(x[1], y[1]), \dots, (x[n], y[n])$, evaluated at t , with error estimate e `polinterpolate($x, \{y\}, \{t\}, \{&e\}$)`
 $T_n/U_n, H_n$ `polchebyshev(n), polhermite(n)`
 $P_n, L_n^{(\alpha)}$ `pollegendre(n), pollaguerre(n, a)`
 n -th cyclotomic polynomial Φ_n `polcyclo(n)`
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)` `polresultant($f, g, \{v\}$)`
resultant $R = \mathrm{Res}_v(f, g)$ `polresultanttext($x, y, \{v\}$)`
 $[u, v, R], xu + yv = \mathrm{Res}_v(f, g)$
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 $A/\mathrm{lc}(A) = \prod_i B[i] \bmod p^e$ `polhensellift(A, B, p, e)`
 $T = \prod (x - z_i) \mapsto \prod (x - \omega(z_i)) \in \mathbf{Z}_p[x]$ `polteichmuller(T, p, e)`
extensions of \mathbf{Q}_p of degree N `padicfields(p, N)`

Pari-GP reference card
(PARI-GP version 2.13.3)

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 = ffggen($T, 't$)`
... indirectly, with implicit T `t = ffggen($q, 't$); 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)`
 x as polmod over codomain of map m `ffmaprel(m, x)`
 F^n over $\mathbf{F}_q \ni a$ `fffrobenius(a, n)`
 $\#\{\text{monic irred. } T \in \mathbf{F}_q[x], \deg T = n\}$ `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 of closure F 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($x, n, \{&z\}$)`
all n -th roots of 1 `rootsof1(n)`
FFT of $[f_0, \dots, f_{n-1}]$ `w = fftinit(n), fft/fftinw(w, f)`
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, loglp, 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)`
 $\sum_{n \leq N} n^s$ `dirpowerssum(N, s)`
Hurwitz's $\zeta(s, x) = \sum (n+x)^{-s}$ `zetahurwitz(s, x)`
multiple zeta value (MZV), $\zeta(s_1, \dots, s_k)$ `zetamult($s, \{T\}$)`
all MZVs for weight $\sum s_i = n$ `zetamultall(n)`
convert MZV id to $[s_1, \dots, s_k]$ `zetamultconvert($f, \{flag\}$)`
MZV dual sequence `zetamultdual(s)`
multiple polylog $Li_{s_1, \dots, s_k}(z_1, \dots, z_k)$ `polylogmult(s, z)`

incomplete Γ function ($y = \Gamma(s)$) `incgam($s, x, \{y\}$)`
complementary incomplete Γ `incgamc(s, x)`
 $\int_x^\infty e^{-t} dt/t$, $(2/\sqrt{\pi}) \int_x^\infty e^{-t^2} dt$ `eint1, erfc`
elliptic integral of 1st and 2nd kind `ellK(k), ellE(k)`
dilogarithm of x `dilog(x)`
 m -th polylogarithm of x `polylog($m, x, \{flag\}$)`
 U -confluent hypergeometric function `hyperu(a, b, u)`
Hypergeometric ${}_pF_q(A, B; z)$ `hypergeom(A, B, z)`
Bessel $J_n(x)$, $J_{n+1/2}(x)$ `besselj(n, x), besseljh(n, x)`
Bessel $I_\nu, K_\nu, H_\nu^1, H_\nu^2, Y_\nu$ `(bessel)i, k, h1, h2, y`
Airy functions $A_i(x)$, $B_i(x)$ `airy(x)`
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 $\pm\infty$ (decreasing at least as x^{-2}) or $(x-a)^{-\alpha}$ singularity `[a, a]`
exponential decrease $e^{-\alpha|x|}$ `[$\pm\infty, a$], $\alpha > 0$`
slow decrease $|x|^\alpha$ `... $\alpha < -1$`
oscillating as $\cos(kx)$ `$\alpha = k\mathbf{I}$, $k > 0$`
oscillating as $\sin(kx)$ `$\alpha = -k\mathbf{I}$, $k > 0$`
numerical integration `intnum($x = a, b, f, \{T\}$)`
weights T for `intnum` `intnuminit($a, b, \{m\}$)`
weights T incl. kernel K `intfuncinit($t = a, b, K, \{m\}$)`
integrate $(2i\pi)^{-1} f$ on circle $|z-a|=R$ `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 $f(n)$, $n \geq a$ (low accuracy) `suminf($n = a, expr$)`
sum of alternating/positive series `sumalt, sumpos`
sum of series using Euler-Maclaurin `sumnum($n = a, f, \{T\}$)`
 $\sum_{n \geq a} F(n)$, F rational function `sumnumrat(F, a)`
 $\dots \sum_{p \geq a} F(p^s)$ `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 $a \leq X \leq b$, initialized at x `prod($X = a, b, expr, \{x\}$)`
product over primes $a \leq X \leq b$ `prodeuler($X = a, b, expr$)`
infinite product $a \leq X \leq \infty$ `prodinf($X = a, expr$)`
 $\prod_{n \geq a} F(n)$, F rational function `prodnunrat(F, a)`
 $\prod_{p \geq a} F(p^s)$ `prodeulerrat($F, \{s = 1\}, \{a = 2\}$)`

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>divisorslensra(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, eulervec</code>
Euler polynomials $E_n(x)$	<code>eulerpol($n, \{x\}$)</code>
Eulerian polynomials $A_n(x)$	<code>eulerianpol</code>
Fibonacci number F_n	<code>fibonacci(n)</code>
Stirling numbers $s(n, k)$ and $S(n, k)$	<code>stirling($n, k, \{flag\}$)</code>

Pari-GP reference card

(PARI-GP version 2.13.3)

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($x, \{o\}$), fford</code>
primitive root mod q / x .mod	<code>znprimroot(q), fprimroot(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>

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>contfrac($x, \{b\}, \{lmax\}$)</code>
last convergent of continued fraction x	<code>contfracpnqn(x)</code>
rational approximation to x (den. $\leq B$)	<code>bestappr($x, \{B\}$)</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	<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>
χp	<code>znchardecompose(G, χ, p)</code>
$\prod_p (Q, N) \chi p$	<code>znchardecompose(G, χ, 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>plotoh($X = a, b, expr, \{flag\}, \{n\}$)</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>
scale contents of w	<code>plotclip(w)</code>
scale coordinates in w	<code>plotscale(w, x_1, x_2, y_1, y_2)</code>
plotoh in w	<code>plotrecth($w, X = a, b, expr, \{flag\}, \{n\}$)</code>
plotthraw in w	<code>plotrectthraw($w, data, \{flag\}$)</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>

Convert to Postscript or Scalable Vector Graphics

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