PariTwine

Edition 0.1.1
August 2022

Andreas Enge, Fredrik Johansson

This manual is for PariTwine, a library to convert between multiprecision types of PARI/GP and external libraries, and to wrap functions from these libraries for use in GP, version 0.1.1 of August 2022.

firstname.name@inria.fr

Table of Contents

PariTwine Copying Conditions

1 Introduction to PariTwine

PariTwine is a glue library between the system for computer algebra and number theory PARI/GP and a number of other mathematics libraries, currently GMP (`https://gmplib.org/`), GNU MPFR (`http://www.mpfr.org/`), GNU MPC (`http://www.multiprecision.org/mpc/`), FLINT (`http://www.flintlib.org/`), ARB (`http://arblib.org/`) and CMH (`http://cmh.gforge.inria.fr/`).

PariTwine provides C functions to convert back and forth between basic types of PARI/GP and the other libraries, and it wraps a number of functions from the external libraries to be called from the PARI library with arguments of PARI type (otherwise said, '`GEN`'). Finally PariTwine makes these wrapped functions available to GP scripts by installing them into the interpreter.

2 Installing PariTwine

To build PariTwine, you first have to install PARI/GP and all the desired libraries that you wish to wrap (at least GMP, and at your choice any of GNU MPFR, GNU MPC, FLINT, ARB and CMH) on your computer. You need a C compiler and a standard Unix '`make`' program, plus some other standard Unix utility programs.

2.1 Basic installation instructions

Here are the steps needed to install the library on Unix systems:

1. '`tar xzf paritwine-0.1.1.tar.gz`'
2. '`cd paritwine-0.1.1`'
3. '`./configure`'

   if dependencies are installed into standard directories, that is, directories that are searched by default by the compiler and the linking tools.

   '`./configure --with-gmp=DIR`'

   is used to indicate a different location where GMP is installed.

   '`./configure --with-pari=DIR`'

   is used to indicate a different location where PARI/GP is installed.

   '`./configure --with-mpfr=DIR`'

   is used to indicate a different location where MPFR is installed.

   '`./configure --with-mpc=DIR`'

   is used to indicate a different location where MPC is installed.

   '`./configure --with-flint=DIR`'

   is used to indicate a different location where FLINT is installed.

   '`./configure --with-arb=DIR`'

   is used to indicate a different location where ARB is installed.

   '`./configure --with-cmh=DIR`'

   is used to indicate a different location where CMH is installed.

   For each package, separate search paths for the header and library files can be specified as follows:

   '`./configure --with-gmp-include=DIR`'

   '`./configure --with-gmp-lib=DIR`'

   and analogously for the other packages.

   Another useful parameter is '`--prefix`', which can be used to specify an alternative installation location instead of `/usr/local`; see '`make install`' below.

   Use '`./configure --help`' for an exhaustive list of parameters.
4. '`make`'

   This compiles PariTwine in the working directory.
5. '`make check`'

   This executes a number of tests on the compiled project. If you get error messages, please report them to the authors.
6. '`make install`'

   This copies the header files `paritwine-config.h` and `paritwine.h` into the directory `/usr/local/include`, the static library `libparitwine.a` and the dynamic library libparitwine.so into the directory /usr/local/lib, the GP script paritwine.gp into the directory /usr/local/share/paritwine and the manual paritwine.info into the directory /usr/local/share/info. If you passed the '--prefix' option to 'configure', the prefix directory given as argument to '--prefix' is used instead of /usr/local. Note that you need write permissions on the prefix directory and its subdirectories.

2.2 Other 'make' Targets

There are some other useful make targets:

- 'pdf'

  This creates a PDF version of the manual in doc/paritwine.pdf.
- 'html'

  This creates an HTML version of the manual, in several pages in the directory doc/paritwine.html; if you want only one output HTML file, then type 'makeinfo --html --no-split paritwine.texi' instead.
- 'clean'

  This deletes all object files and archive files, but not the configuration files.
- 'distclean'

  This has the same effect as 'make clean', but it additionally deletes the configuration files created by './configure'.
- 'uninstall'

  This deletes all files copied by 'make install'.

3 Using PariTwine

PariTwine consists of essentially three parts:

- C functions for converting between the basic PARI types and the types of external libraries;
- C functions for wrapping functions of external libraries to be called with arguments of PARI types;
- a GP script to call these functions from within the GP command interpreter.

The following three sections describe these functionalities in order.

3.1 Conversion functions

Basically, for each external type `foo_t`, we provide two functions:

`void foo_set_GEN` (*foo_t* `z`, *GEN* `x`) [Function]
Set the value of *z* from the PARI variable *x*, which needs to be of compatible type; otherwise, a PARI error is raised.

`GEN foo_get_GEN` (*foo_t* `z`) [Function]
Create from *z* a PARI object (of C type `GEN`) of suitable PARI type on the PARI stack and return it.

Functions operating on floating point numbers may take as additional argument a rounding mode and return an integer indicating the effective direction of rounding.

3.1.1 Conversion functions for scalar types

`void mpz_set_GEN` (*mpz_t* `z`, *GEN* `x`) [Function]
`void fmpz_set_GEN` (*fmpz_t* `z`, *GEN* `x`) [Function]
Set the GMP or FLINT integer variable *z* to the value of *x*, which must be of PARI type `t_INT`.

`GEN mpz_get_GEN` (*mpz_t* `z`) [Function]
`GEN fmpz_get_GEN` (*fmpz_t* `z`) [Function]
From the GMP or FLINT integer *z* create a variable of PARI type `t_INT` on the PARI stack and return it.

`void mpq_set_GEN` (*mpq_t* `z`, *GEN* `x`) [Function]
`void fmpq_set_GEN` (*fmpq_t* `z`, *GEN* `x`) [Function]
Set the GMP or FLINT rational variable *z* to the value of *x*, which must be of PARI type `t_INT` or `t_FRAC`.

`GEN mpq_get_GEN` (*mpq_t* `z`) [Function]
`GEN fmpq_get_GEN` (*fmpq_t* `z`) [Function]
From the GMP or FLINT rational *z* create a variable of PARI type `t_FRAC` on the PARI stack and return it.

`int mpfr_set_GEN` (*mpfr_t* `z`, *GEN* `x`, *mpfr_rnd_t* `rnd`) [Function]
Set the MPFR floating point variable *z* to the value of *x*, which must be of PARI type `t_INT`, `t_FRAC` or `t_REAL`. The variable *z* must have been initialised to a given precision before, and the assigned value is the value of *x* rounded according to the rounding mode *rnd*; one possible choice is to use the constant `MPFR_RNDN` for rounding to nearest. The return value has the usual semantics of MPFR functions and indicates the effective direction of rounding: 0 if the result is exactly represented without rounding, a positive integer if the result is larger than the exact value and a negative integer if the result is smaller than the exact value.

`GEN mpfr_get_GEN` (*mpfr_t* `z`) [Function]

From the MPFR floating point number *z* create a variable of PARI type `t_REAL` on the PARI stack and return it. The precision of the created variable is the minimal possible precision in PARI (a multiple of the word size) that is at least the bit precision of *z*.

`int mpc_set_GEN` (*mpc_t* `z`, *GEN* `x`, *mpfr_rnd_t* `rnd`) [Function]

Set the MPC floating point variable *z* to the value of *x*, which must be of PARI type `t_INT`, `t_FRAC`, `t_REAL` or `t_COMPLEX`. The variable *z* must have been initialised to a given precision before, and the assigned value is the value of *x* rounded according to the rounding mode *rnd*; one possible choice is to use the constant `MPC_RNDNN` for rounding both the real and the imaginary part to nearest. The return value has the usual semantics of MPC functions and indicates the effective direction of rounding for the real and the imaginary part; for more details, see the MPC documentation.

`GEN mpc_get_GEN` (*mpc_t* `z`) [Function]

From the MPC floating point number *z* create a variable of PARI type `t_COMPLEX` on the PARI stack and return it. The real and imaginary parts of the result are created using `mpfr_get_GEN`. So in particular their precisions are determined separately as the minimal possible precisions in PARI (multiples of the word size) that are at least the bit precisions of the corresponding parts of *z*.

3.1.2 Conversion functions for ball types

`void arf_set_GEN` (*arf_t* `z`, *GEN* `x`) [Function]
`void mag_set_GEN` (*mag_t* `z`, *GEN* `x`) [Function]

ARB implements two real floating point types, `arf_t` for holding the centre point of a real interval in ball representation at arbitrary precision, and `mag_t` for holding the radius of the interval (the "magnitude of the error") at small fixed precision. These two functions set the ARB floating point variable *z* to the value of *x*, which must be of PARI type `t_INT` or `t_REAL`. In the case of *z* of type `arf_t`, its precision is chosen minimal such that *x* can be stored exactly without rounding. In the case of *z* of type `mag_t`, the value is rounded up if necessary.

`GEN arf_get_GEN` (*arf_t* `z`, *long* `prec`) [Function]
`GEN mag_get_GEN` (*mag_t* `z`) [Function]

From the ARB floating point number *z* create a PARI variable on the PARI stack and return it. If the value of *z* is 0, then the return value is of PARI type `t_INT`. Otherwise it is of PARI type `t_REAL`, and in the case of `arf_get_GEN`, the result is rounded to at least a precision of *prec* bits (precisely, to the next multiple of the word size); in the case of `mag_get_GEN`, a `t_REAL` of the minimal precision to hold the exact value of *z* is returned.

`void arb_set_GEN` (*arb_t* `z`, *GEN* `x`, *long* `prec`) [Function]

Set the ARB real ball variable *z* to the value of *x*, which can be of PARI type `t_INT`, `t_FRAC`, `t_REAL` or `t_VEC`. If *x* is not of PARI type `t_VEC`, then the interval has as centre *x* rounded to precision *prec* and is taken of minimal size to handle the rounding error.

If *x* is of PARI type `t_VEC`, it is supposed to represent an interval itself, that is, it contains two elements representing the centre and the radius. These are transformed into an `arb_t` interval by calls to `arf_set_GEN` and `mag_set_GEN`, respectively, on the two components. This transformation does not use the parameter *prec* and thus preserves exactly the centre, and it potentially rounds up the radius.

`GEN arb_get_GEN` (*arb_t* `z`, *long* `prec`) [Function]

From the ARB real ball *z*, create a PARI variable on the PARI stack and return it. The result is of PARI type `t_VEC` with two elements and contains the interval in *z* as obtained by calls to `arf_get_GEN` and `mag_get_GEN`, respectively.

Notice that in a sequence of alternating calls to `arb_get_GEN` and `arb_set_GEN`, starting with either of them, and with the same value of *prec*, the first call may result in rounding if the value of *prec* is not large enough, or if the sequence starts with the conversion of a `t_FRAC`. In this case, the rounded ball always contains the input value. The subsequent calls will be lossless.

`void acb_set_GEN` (*acb_t* **z**, *GEN* **x**, *long* **prec**) [Function]

In ARB, complex "balls" of type `acb_t` are implemented as a pair of real intervals of type `arb_t` representing the real and imaginary parts; so they are in fact rectangles in the complex plane. The same complex rectangle is represented in PARI in a "transposed" form, as a `t_VEC` with two elements of type `t_COMPLEX`, the first of which represents the centre of the rectangle, and the second of which represents the two radii in the real and the imaginary direction.

If *x* is of PARI type `t_COMPLEX`, the resulting value of *z* is the smallest complex ball with centre *x* at precision *prec*.

If *x* is of PARI type `t_INT`, `t_FRAC` or `t_REAL`, the imaginary part of *z* is set to an exact 0, while its real part is set to a real ball by a call to `arb_set_GEN (z, x, prec)`.

If *x* is of PARI type `t_VEC`, it is interpreted as a complex rectangle that is transformed into the corresponding `acb_t` rectangle *z* by calls to `arb_set_GEN`. This operation does not use the parameter *prec*, so that it preserves exactly the centre of the complex ball, while the real and imaginary radii may be rounded up.

`GEN acb_get_GEN` (*acb_t* **z**, *long* **prec**) [Function]

From the ARB complex ball *z*, create a PARI variable on the PARI stack and return it. The result is of PARI type `t_VEC` with two elements, each of which are of type `t_COMPLEX`; it represents the same complex rectangle as *z*, with the real and imaginary part of its centre rounded through calls to `arf_get_GEN`.

Notice that as for real balls of type `arb_t`, in a sequence of alternating calls to `acb_set_GEN` and `acb_get_GEN` with the same precision, only the first call may lead to rounding (in which case the output ball contains the input ball), while all following ones are lossless.

3.1.3 Initialisation on the PARI stack

For the GNU MPFR and GNU MPC libraries, the following initialisation functions, the names of which start with the prefix `pari_`, have a special behaviour: Exactly like their counterparts without the `pari_` prefix from the respective libraries, they initialise a variable of type `mpfr_t` or `mpc_t`, but they allocate their mantissae on the PARI stack. So they should not be freed with calls to `mpfr_clear` or `mpc_clear`, but with the usual PARI stack handling (also known as "avma magic"). They are used internally inside the wrappers for functions from MPFR and MPC, but they may also be more efficient for use in C code relying on `libpari` and requiring handling of the PARI stack anyway.

`void pari_mpfr_init2` (*mpfr_t z, mpfr_prec_t prec*) [Function]
`void pari_mpc_init2` (*mpc_t z, mpfr_prec_t prec*) [Function]
`void pari_mpc_init3` (*mpc_t z, mpfr_prec_t prec_re, mpfr_prec_t prec_im*) [Function]

These are the counterparts of `mpfr_init2`, `mpc_init2` and `mpc_init3`. All of them take additional arguments to determine the bit precisions of the numbers (the `init2` variants) or of the real and imaginary part separately (`mpc_init3`).

There are also functions combining initialisations of MPFR or MPC numbers on the PARI stack with assignments of PARI numbers.

`void pari_mpfr_init_set_GEN` (*mpfr_t z, GEN x, mpfr_prec_t default_prec*) [Function]

Conceptually, this function combines a call to `pari_mpfr_init2` and `mpfr_set_GEN`. However, the precision handling is special and depends on the type of *x*: If *x* is of the floating point type `t_REAL`, the precision used for initialising *z* is the same as that of *x*, so that the result fits without rounding. If *x* is of an exact type (`t_INT` or `t_FRAC`), however, the value of *default_prec* is used for initialising *z*.

`void pari_mpc_init_set_GEN` (*mpc_t z, GEN x, mpfr_prec_t default_prec*) [Function]

This function calls `pari_mpfr_init_set_GEN` to initialise the real part of *z* and to assign the real part of *x* to it, and to separately initialise the complex part of *z* and to assign the complex part of *x* to it. Notice that the real and complex parts of *x* may have as types arbitrary combinations of `t_INT`, `t_FRAC` and `t_REAL`, and that the precision is determined by `pari_mpfr_init_set_GEN` independently for each part.

3.2 Wrapped library functions

Besides providing functions to convert between PARI types and types of external libraries, a goal of PariTwine is to wrap functions from the external libraries so that they can be called directly from PARI with PARI type arguments, returning a PARI type result.

Roughly speaking, if `void lib_func (lib_t z, lib_t x, lib_t y, ...)` is a function from the library *lib* computing the mathematical function *func* in the arguments *x*, *y*, ... and assigning the result to *z*, where all these variables are of some type *lib_t* defined in *lib*, we wrap it to obtain a function `GEN pari_lib_func (GEN x, GEN y, ..., long prec)` that uses `lib_func` to compute *func* on the PARI type arguments *x*, *y*, ... and that returns the result as a PARI object. The additional parameter *prec* determines the working precision (in bits) used in the external library and also the precision of the result. Usually functions in GNU MPFR and GNU MPC take as an additional parameter a rounding mode; this parameter is dropped in the wrapped function, where rounding to nearest is used. Functions in MPFR and MPC also usually have an `int` return value, which indicates the effective rounding direction of the result; this is discarded. For instance, the MPFR function computing the Riemann zeta function, `int mpfr_zeta (mpfr_t z, mpfr_t x, mpfr_rnd_t rnd)` is wrapped to become `GEN pari_mpfr_zeta (GEN x, GEN y, long prec)`.

Currently, the following wrapped functions are available in PariTwine; see Chapter 4 [Extending PariTwine], page 13, for instructions on how to add more functions.

`GEN pari_mpfr_add` (*GEN x, GEN y, long prec*) [Function]
`GEN pari_mpfr_sub` (*GEN x, GEN y, long prec*) [Function]
`GEN pari_mpfr_mul` (*GEN x, GEN y, long prec*) [Function]
`GEN pari_mpfr_sqr` (*GEN x, long prec*) [Function]
`GEN pari_mpfr_div` (*GEN x, GEN y, long prec*) [Function]
`GEN pari_mpfr_sqrt` (*GEN x, long prec*) [Function]
`GEN pari_mpfr_rec_sqrt` (*GEN x, long prec*) [Function]
`GEN pari_mpfr_cbrt` (*GEN x, long prec*) [Function]
`GEN pari_mpfr_pow` (*GEN x, GEN y, long prec*) [Function]
`GEN pari_mpfr_log` (*GEN x, long prec*) [Function]
`GEN pari_mpfr_log2` (*GEN x, long prec*) [Function]
`GEN pari_mpfr_log10` (*GEN x, long prec*) [Function]
`GEN pari_mpfr_exp` (*GEN x, long prec*) [Function]
`GEN pari_mpfr_exp2` (*GEN x, long prec*) [Function]
`GEN pari_mpfr_exp10` (*GEN x, long prec*) [Function]
`GEN pari_mpfr_sin` (*GEN x, long prec*) [Function]
`GEN pari_mpfr_cos` (*GEN x, long prec*) [Function]
`GEN pari_mpfr_tan` (*GEN x, long prec*) [Function]
`GEN pari_mpfr_sec` (*GEN x, long prec*) [Function]

GEN `pari_mpfr_csc` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_cot` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_acos` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_asin` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_atan` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_cosh` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_sinh` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_tanh` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_sech` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_csch` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_coth` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_acosh` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_asinh` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_atanh` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_log1p` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_expm1` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_eint` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_li2` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_gamma` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_lngamma` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_digamma` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_zeta` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_erf` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_erfc` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_j0` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_j1` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_y0` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_y1` (*GEN x, long prec*) [Function]
GEN `pari_mpfr_fma` (*GEN x, GEN y, GEN z, long prec*) [Function]
GEN `pari_mpfr_fms` (*GEN x, GEN y, GEN z, long prec*) [Function]
GEN `pari_mpfr_agm` (*GEN x, GEN y, long prec*) [Function]
GEN `pari_mpfr_hypot` (*GEN x, GEN y, long prec*) [Function]
GEN `pari_mpfr_ai` (*GEN x, long prec*) [Function]

These functions take arguments of types `t_INT`, `t_FRAC` or `t_REAL` and use GNU MPFR to return a result of type `t_REAL`.

GEN `pari_mpfr_fac_ui` (*unsigned long int t, long prec*) [Function]

This function takes as argument a small unsigned integer and returns its factorial as a number of type `t_REAL`.

GEN `pari_mpfr_jn` (*long int i, GEN x, long prec*) [Function]
GEN `pari_mpfr_yn` (*long int i, GEN x, long prec*) [Function]

These functions take as arguments a small integer and a number of type `t_INT`, `t_FRAC` or `t_REAL` and return a Bessel function of the given order of the first or second kind in the argument.

GEN `pari_mpc_add` (*GEN x, GEN y, long prec*) [Function]
GEN `pari_mpc_sub` (*GEN x, GEN y, long prec*) [Function]
GEN `pari_mpc_mul` (*GEN x, GEN y, long prec*) [Function]
GEN `pari_mpc_sqr` (*GEN x, long prec*) [Function]
GEN `pari_mpc_fma` (*GEN x, GEN y, GEN z, long prec*) [Function]
GEN `pari_mpc_div` (*GEN x, GEN y, long prec*) [Function]
GEN `pari_mpc_sqrt` (*GEN x, long prec*) [Function]

GEN pari_mpc_pow (*GEN x, GEN y, long prec*) [Function]
GEN pari_mpc_exp (*GEN x, long prec*) [Function]
GEN pari_mpc_log (*GEN x, long prec*) [Function]
GEN pari_mpc_log10 (*GEN x, long prec*) [Function]
GEN pari_mpc_sin (*GEN x, long prec*) [Function]
GEN pari_mpc_cos (*GEN x, long prec*) [Function]
GEN pari_mpc_tan (*GEN x, long prec*) [Function]
GEN pari_mpc_sinh (*GEN x, long prec*) [Function]
GEN pari_mpc_cosh (*GEN x, long prec*) [Function]
GEN pari_mpc_tanh (*GEN x, long prec*) [Function]
GEN pari_mpc_asin (*GEN x, long prec*) [Function]
GEN pari_mpc_acos (*GEN x, long prec*) [Function]
GEN pari_mpc_atan (*GEN x, long prec*) [Function]
GEN pari_mpc_asinh (*GEN x, long prec*) [Function]
GEN pari_mpc_acosh (*GEN x, long prec*) [Function]
GEN pari_mpc_atanh (*GEN x, long prec*) [Function]

These functions take arguments of types t_INT, t_FRAC, t_REAL or t_COMPLEX and use GNU MPC to return a result of type t_COMPLEX.

GEN pari_mpc_abs (*GEN x, long prec*) [Function]
GEN pari_mpc_norm (*GEN x, long prec*) [Function]

These functions take arguments of types t_INT, t_FRAC, t_REAL or t_COMPLEX and use MPC to return a result of type t_REAL.

GEN pari_cmh_I2I4I6I10 (*GEN tau, long prec*) [Function]
GEN pari_cmh_4theta (*GEN tau, long prec*) [Function]
GEN pari_cmh_10theta2 (*GEN tau, long prec*) [Function]

These functions do not completely fit the generic description above and might change in the future. They take as input a 2x2-matrix *tau* of type t_MAT with entries of type t_COMPLEX, which is supposed to be an element of the Siegel half space; in particular, *tau* is symmetric, and its lower left entry is not used. They use the CMH library to compute and return a vector of type t_VEC, containing four or ten elements of type t_COMPLEX. The first function computes the Igusa-Clebsch invariants I_2, I_4, I_6 and I_{10}. The second function computes the first four theta constants. The third function computes the squares of the ten non-zero theta constants.

GEN pari_acb_add (*GEN x, GEN y, long prec*) [Function]
GEN pari_acb_sub (*GEN x, GEN y, long prec*) [Function]
GEN pari_acb_mul (*GEN x, GEN y, long prec*) [Function]
GEN pari_acb_div (*GEN x, GEN y, long prec*) [Function]
GEN pari_acb_neg (*GEN x, long prec*) [Function]
GEN pari_acb_conj (*GEN x, long prec*) [Function]
GEN pari_acb_exp (*GEN x, long prec*) [Function]
GEN pari_acb_sqrt (*GEN x, long prec*) [Function]
GEN pari_acb_log (*GEN x, long prec*) [Function]
GEN pari_acb_pow (*GEN x, GEN y, long prec*) [Function]
GEN pari_acb_atan (*GEN x, long prec*) [Function]
GEN pari_acb_sin (*GEN x, long prec*) [Function]
GEN pari_acb_cos (*GEN x, long prec*) [Function]
GEN pari_acb_sinh (*GEN x, long prec*) [Function]
GEN pari_acb_cosh (*GEN x, long prec*) [Function]
GEN pari_acb_agm (*GEN a, GEN b, long prec*) [Function]
GEN pari_acb_elliptic_k (*GEN x, long prec*) [Function]

GEN `pari_acb_elliptic_e` (*GEN x, long prec*) [Function]
GEN `pari_acb_elliptic_pi` (*GEN x, long prec*) [Function]
GEN `pari_acb_gamma` (*GEN x, long prec*) [Function]
GEN `pari_acb_digamma` (*GEN x, long prec*) [Function]
GEN `pari_acb_zeta` (*GEN s, long prec*) [Function]
GEN `pari_acb_hurwitz_zeta` (*GEN s, GEN z, long prec*) [Function]
GEN `pari_acb_modular_eta` (*GEN tau, long prec*) [Function]
GEN `pari_acb_modular_j` (*GEN tau, long prec*) [Function]
GEN `pari_acb_modular_delta` (*GEN tau, long prec*) [Function]
GEN `pari_acb_elliptic_p` (*GEN z, GEN tau, long prec*) [Function]
GEN `pari_acb_elliptic_inv_p` (*GEN z, GEN tau, long prec*) [Function]
GEN `pari_acb_elliptic_zeta` (*GEN z, GEN tau, long prec*) [Function]
GEN `pari_acb_elliptic_sigma` (*GEN z, GEN tau, long prec*) [Function]
GEN `pari_acb_hypgeom_2f1` (*GEN a, GEN b, GEN c, GEN z, long flags, long prec*) [Function]

These functions take `GEN` arguments of types `t_INT`, `t_FRAC`, `t_REAL`, `t_COMPLEX` or `t_VEC` and return a result of type `t_VEC`. Here the `t_VEC` are vectors with two complex components, representing the centre and the radius of a complex rectangle.

Notice that unless the default precision is changed in between, it is safe to compose these functions operating on complex rectangles, since the conversion back and forth between GP and ARB is then lossless. In this way it is possible to build more complicated expressions using interval arithmetic all along, such that the final result contains the exact mathematical value.

GEN `pari_acb_modular_theta` (*GEN z, GEN tau, long prec*) [Function]

The function takes the same type of arguments as the previous ones, but instead of returning one result, it returns a `t_VEC` with four entries (each of which is a `t_VEC` representing a complex rectangle). It computes the four Jacobi theta functions θ_1, θ_2, θ_3 and θ_4 (in arb notation), which correspond to $-i\theta_{1,1}$, $\theta_{1,0}$, $\theta_{0,0}$ and $\theta_{0,1}$ (in notation using half-integral characteristics).

GEN `pari_acb_modular_eisenstein` (*GEN tau, long n, long prec*) [Function]

As the previous function, this one returns a `t_VEC`, but this time of length n, of complex rectangles. The vector contains the n first Eisenstein series G_4, G_6, G_8,...

int `pari_acb_overlaps` (*GEN x, GEN y, long prec*) [Function]
int `pari_acb_contains` (*GEN x, GEN y, long prec*) [Function]

The first function returns 1 or 0 depending on whether the complex rectangles given by x and y overlap or not; in the first case, they may represent the same real number, in the second case they represent distinct real numbers. As other functions operating on complex rectangles, these can be given as `t_INT`, `t_FRAC`, `t_REAL`, `t_COMPLEX` or `t_VEC`. The second function checks whether y is contained in x; if y is of scalar type, the two functions have the same semantics.

GEN `pari_fmpz_numbpart` (*GEN x*) [Function]
GEN `pari_arb_numbpart` (*GEN x, long prec*) [Function]

These functions take an argument x of type `t_INT` and compute the partition number of x. The first one uses FLINT to return the exact `t_INT`, the second one uses ARB to return a real ball of type `t_VEC` at the given precision.

3.3 Calling wrapped functions from GP

PariTwine provides a GP snippet, `paritwine.gp`, which can be used to integrate the wrapped functions from the external libraries into the GP command interpreter. This file is copied by `make install` into the subdirectory `share/paritwine` of the installation prefix (`/usr/local`, unless specified otherwise). To use it, issue the command

```
\r /usr/local/share/paritwine/paritwine.gp
```

Roughly speaking, if `void` *`lib_func`* `(`*`lib_t z`*`,` *`lib_t x`*`,` *`lib_t y`*`, ...)` is a function from the library *lib*, wrapped as the C library function `GEN pari_`*`lib_func`* `(GEN` *`x`*`, GEN` *`y`*`, ..., long` *`prec`*`)` on PARI types, inclusion of the above GP snippet makes a GP function available that can be called as *`lib_func`* `(x, y, ...)`. The parameter *prec* is omitted and replaced by the current default bit precision. For instance, `mpfr_zeta (2)` uses GNU MPFR to compute the Riemann zeta function in the argument 2 at the current default precision.

4 Extending PariTwine

For wrapping a new function from an external library, one needs to add the wrapper function to one of the C files, a process that shall be illustrated with the function `int mpfr_zeta (mpfr_t z, mpfr_t x, mpfr_rnd_t rnd)` (which already exists in PariTwine). The wrapper function could look like this:

```
GEN pari_mpfr_zeta (GEN x, long prec)
{
   pari_sp ltop = avma;
   mpfr_prec_t p = prec;
   mpfr_t z, z1;

   pari_mpfr_init2 (z, p);
   pari_mpfr_init_set_GEN (z1, x, p);

   mpfr_zeta (z, z1, MPFR_RNDN);

   return gerepileuptoleaf (ltop, mpfr_get_GEN (z));
}
```

The first line memorises the state of the PARI stack in the variable `ltop`. The second line casts the PARI precision of type `long` into an MPFR precision (which could be dropped, since in general the latter is also `long`). The next line declares two variables, `z1` to hold the MPFR version of the argument `x`, and `z` to hold the result of the computation. Then `z` is initialised on the PARI stack with the desired precision, and `z1` is initialised and set to `x`. Hereby if `x` is of type `t_INT` or `t_FRAC`, the precision `prec` is used; if it is of type `t_REAL`, its own precision is used, which may be different from `prec`. Then the function `mpfr_zeta` is called with rounding to nearest (`MPFR_RNDN`), which puts the result of the computation into `z`. The subexpression `mpfr_get_GEN (z)` adds an object of type `t_REAL` to the PARI stack with the same value as `z`. The surrounding call to `gerepileupto` deletes everything between `ltop` and this result on the PARI stack and returns a pointer to the result. So the effect of the function on the PARI stack is exactly to have added this result.

The modified version of PariTwine is compiled and installed using `make install`.

The next (optional) step is to make this new library function available in the GP command interpreter. This can be done issuing the command

```
install ("pari_mpfr_zeta", "Gb", "mpfr_zeta", "/usr/local/lib/libparitwine.so");
```

It takes the function `pari_mpfr_zeta` from the shared library `/usr/local/lib/libparitwine.so` and installs it under the name of `mpfr_zeta`. The code `Gb` indicates that the function takes one argument of type `GEN` and also the current default bit precision of the GP environment; the latter is added automatically and need not be specified in the function call. The return value of type `GEN` is also understood. So now it is possible to call

```
Pisquareoversix = mpfr_zeta (2);
```

in the GP interpreter.

If you have extended PariTwine by wrapping more functions or adding a new external library, you may wish to contact the authors to have your modifications included into a future release.

Appendix A GNU Free Documentation License

0. PREAMBLE

   The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

   This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

   We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

   This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

   A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

   A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

   The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

   The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “publisher” means any person or entity that distributes copies of the Document to the public.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its

Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

    "Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

    "CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

    "Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

    An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

    The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.