PariTwine
A bridge between PARI/GP and the GNU MP universe

Andreas Enge

LFANT project-team
INRIA Bordeaux–Sud-Ouest
andreas.enge@inria.fr
http://www.math.u-bordeaux.fr/~aenge

(Software written together with Fredrik Johansson)
MPFR/MPC/iRRAM Workshop, Trier, 21 November 2018
Motivation

- Use C code for evaluating two-dimensional \( \vartheta \)-functions from inside GP.
Motivation

- Use C code for evaluating two-dimensional $\vartheta$-functions from inside GP.
- Use C code from the GNU multiprecision universe inside GP:
  - GMP
  - MPFR
  - MPC
  - MPFRCX, CM, CMH
  - FPLLL
  - ...
Use C code for evaluating two-dimensional \( \varphi \)-functions from inside GP.

Use C code from the GNU multiprecision universe inside GP:

- GMP
- MPFR
- MPC
- MPFRCX, CM, CMH
- FPLLL
- ...

Create a bridge between the GNU MP universe and PARI/GP.
Motivation

- Use C code for evaluating two-dimensional \( \theta \)-functions from inside GP.
- Use C code from the GNU multiprecision universe inside GP:
  - GMP
  - MPFR
  - MPC
  - MPFRCX, CM, CMH
  - FPLLL
  - ...
- Create a bridge between the GNU MP universe and PARI/GP.
- Create a FFI for the GNU MP libraries in PARI/GP.
- Software for algebra and number theory
- Written since 1985 in Bordeaux
- Number theory backend of SageMath
- C library (PARI) and REPL/command interpreter (GP)

Mini-demo GP
1. Installation

2. Conversion of numbers

3. Wrapped library functions

4. Calling functions from GP
Installation

- Version 0.0.1 of 2014 — outdated, do not use.
- Development version
  - git clone
    https://scm.gforge.inria.fr/anonscm/git/pari-gnump/pari-gnump.git
  - autoreconf -vfi
  - ./configure
  - make
  - make check
  - make install
  - make pdf; xpdf doc/paritwine.pdf
1. Installation

2. Conversion of numbers

3. Wrapped library functions

4. Calling functions from GP
Memory management

- **PARI**: `t_INT, t_FRAC, t_REAL, t_COMPLEX`
  - stores numbers on the PARI stack
  - allocates sort of automatically:
    ```c
    GEN c;
    c = gadd (a, b);
    ```
  - frees by moving the stack pointer (`avma, gerepile`)

- **GMP, MPFR, MPC**: `mpz_t, mpq_t, mpfr_t, mpc_t`
  - store numbers on the heap
  - require explicit allocation (`mpz_init, mpc_init2 → malloc`):
    ```c
    mpz_t c;
    mpz_init (c);
    mpz_add (c, a, b);
    ```
  - require explicit freeing (`mpz_clear, mpc_clear → free`):
    ```c
    mpz_clear (c);
    ```
Precision

- **PARI**
  - has a global precision for the creation of variables
  - each variable implicitly has a given precision
  - works on a best-effort basis for rounding

- **MPFR, MPC**
  - assign a separate precision to each variable
    ```c
    mpc_init2 (c, 200);
    ```
  - accept a rounding mode per operation and guarantee the result
    ```c
    mpc_mul (c, a, b, MPC_RNDND);
    ```
Edianness

- Both store numbers as arrays of unsigned long int.
- \texttt{t\_INT} and \texttt{mpz\_t} have the same endianness.
- \texttt{t\_REAL} has the other endianness.

Conversion functions provided by Karim Belabas
Conversion functions: paritwine.h

- From PARI to MP*
  - void mpz_set_GEN (mpz_ptr z, GEN x);
  - void mpq_set_GEN (mpq_ptr q, GEN x);
  - int mpfr_set_GEN (mpfr_ptr f, GEN x, mpfr_rnd_t rnd);
  - int mpc_set_GEN (mpc_ptr c, GEN x, mpc_rnd_t rnd);

  x of type t_INT, t_FRAC, t_REAL, t_COMPLEX, as suitable
  Semantics: consider x as exact, round and return inexact value
Conversion functions: paritwine.h

- **From PARI to MP**
  - void mpz_set_GEN (mpz_ptr z, GEN x);
  - void mpq_set_GEN (mpq_ptr q, GEN x);
  - int mpfr_set_GEN (mpfr_ptr f, GEN x, mpfr_rnd_t rnd);
  - int mpc_set_GEN (mpc_ptr c, GEN x, mpc_rnd_t rnd);

  x of type t_INT, t_FRAC, t_REAL, t_COMPLEX, as suitable
  Semantics: consider x as exact, round and return inexact value

- **From MP** to PARI
  - GEN mpz_get_GEN (mpz_srcptr z);
  - GEN mpq_get_GEN (mpq_srcptr q);
  - GEN mpfr_get_GEN (mpfr_srcptr f);
  - GEN mpc_get_GEN (mpc_srcptr c);

  Semantics: Create t_REAL or t_COMPLEX with the minimal precision
  to store f or c without loss
Use the PARI heap for MP*: paritwine.h

- Allocate mpfr and mpc numbers on the PARI heap; do not free!
  - void pari_mpfr_init2 (mpfr_ptr f, mpfr_prec_t prec);
  - void pari_mpc_init2 (mpc_ptr c, mpfr_prec_t prec);
  - void pari_mpc_init3 (mpc_ptr c, mpfr_prec_t prec_re, mpfr_prec_t prec_im);

- Emulate PARI precision handling
  - void pari_mpfr_init_set_GEN (mpfr_ptr f, GEN x, mpfr_prec_t default_prec);
  - void pari_mpc_init_set_GEN (mpc_ptr c, GEN x, mpfr_prec_t default_prec);

Allocate on the PARI heap.
For t_REAL components, use their own precision.
For t_INT and t_FRAC components, use default_prec.
1. Installation
2. Conversion of numbers
3. Wrapped library functions
4. Calling functions from GP
#include "paritwine.h"

- **MPC**
  - GEN pari_mpc_mul (GEN x, GEN y, long prec);
  - all others!

- **MPFR**
  - GEN pari_mpfr_zeta (GEN x, long prec);
  - all others!

- **ARB**
  - GEN pari_acb_zeta (GEN s, long prec);
  - some more, ask Fredrik!

- **CMH**
  - GEN pari_cmos_I2I4I6I10 (GEN tau, long prec);
  - GEN pari_cmos_4theta (GEN tau, long prec);
  - GEN pari_cmos_10theta2 (GEN tau, long prec);
GEN pari_mpfr_zeta (GEN x, long prec)
{
    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 mpfr_get_GEN (z);
}
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));
}
pari_mpfr_zeta

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));
}

In reality: macro generated
1. Installation

2. Conversion of numbers

3. Wrapped library functions

4. Calling functions from GP
Use the Foreign Function Interface of GP.

LIBPARITWINESO="PREFIX/lib/libparitwine.so";
install ("pari_mpfr_zeta", "Gb", "mpfr_zeta", LIBPARITWINESO);

- Takes our new function pari_mpfr_zeta
- from the installed libparitwine.so,
- with one argument of type GEN and the default bit precision;
- and calls it mpfr_zeta inside GP.
read ("PREFIX/share/paritwine/paritwine.gp");
mpfr_zeta (3)

Mini-demo GP
Plans for the future

- Finish wrapping ARB.
- Wrap CM for use in ECPP.
- Wrap FPLLL to test our LLL implementation (requires vectors and matrices).
- Wrap your favourite library.