Recent and future developments of GNU MPC

Andreas Enge

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

iRRAM/MPFR/MPC Workshop, Dagstuhl, April 18–20, 2018
MPC in one slide

- **URL**: http://mpc.multiprecision.org/
- **Authors**: AE, Paul Zimmermann; Mickaël Gastineau, Philippe Théveny
- **Institutions**: INRIA LFANT and CARAMBA, CNRS
- **Licence**: LGPL 3+
- **Current version**: 1.1.0 “Gladiolus palustris”
- **Language**: C99
- **Dependencies**: GNU MP $\geq 5.0.0$, GNU MPFR $\geq 3.0.0$ (both of 2010)
- **Installation**
  - ./configure; make; make install
<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>2002</td>
<td>Version 0.1, AE and PZ</td>
</tr>
<tr>
<td>2005</td>
<td>Inclusion into Magma</td>
</tr>
<tr>
<td>2007</td>
<td>Inclusion into Trips, Windows support (MG)</td>
</tr>
<tr>
<td>2008</td>
<td>Debian packages for squeeze (Laurent Fousse)</td>
</tr>
<tr>
<td>2007–2009</td>
<td>PT:</td>
</tr>
<tr>
<td></td>
<td>Trigonometric functions</td>
</tr>
<tr>
<td></td>
<td>Testing framework</td>
</tr>
<tr>
<td>2009</td>
<td>Summer school on GNU MPFR and GNU MPC in Nancy</td>
</tr>
<tr>
<td>2010</td>
<td>Mandatory for GCC 4.5</td>
</tr>
<tr>
<td>2011</td>
<td>Version 0.9, complete with all functions of the C standard</td>
</tr>
<tr>
<td>2011</td>
<td>First joint GNU MPFR and GNU MPC developers’ meeting</td>
</tr>
<tr>
<td>2011</td>
<td>GNU project, license change</td>
</tr>
<tr>
<td>2012</td>
<td>Version 1.0</td>
</tr>
<tr>
<td>2018</td>
<td>Version 1.1.0</td>
</tr>
</tbody>
</table>
typedef struct {
    mpfr_t re;
    mpfr_t im;
} __mpc_struct;
typedef __mpc_struct mpc_t[1];

- Each coordinate carries its own precision.
- Coordinate access
  - mpc_realref (z)
  - mpc_imagref (z)
Usage

- **Initialisation**
  - mpc_init2 (z, prec)
  - mpc_init3 (z, prec_re, prec_im)

- **Assignment**
  - mpc_set
  - mpc_set_fr, mpc_set_ui, ...
  - mpc_set_fr_fr, mpc_set_ui_ui, ...

- **Computation**

- **Freeing**
  - mpc_clear (z)
Semantics

- **Rounding modes**
  - Separate for each coordinate
  - \( \text{MPC}_\text{RNDNU} = \text{MPFR}_\text{RNDN} + i \text{MPFR}_\text{RNDU} \)

- **Correct rounding** of each coordinate for atomic operation

- **Nonary return value**
  - Result exact/rounded down/rounded up by coordinate
  - \( \text{MPC}_\text{INEX}_\text{RE} \) \((\text{ret})\) \(\in\) \{0, −1, 1\}
  - \( \text{MPC}_\text{INEX}_\text{IM} \) \((\text{ret})\) \(\in\) \{0, −1, 1\}

- **Comparison**
  - \( \text{mpc}_\text{cmp} \) \((z1, z2)\)
  - Returns nonary value
  - Value == 0 iff \( z1 == z2 \)
All functions in the C99 standard since version 0.9
1.1.0 “Gladiolus palustris”

Minimally required library versions:
Gnu Mp 5.0.0 and Gnu Mpfr 3.0.0
(of 2010)

Fixed issues with Gnu Mpfr 4.0.0

Rewrite of the testing framework

New mpcbench tool, used with make bench

New functions:

- mpc_rootofunity
- mpc_cmp_abs
  (paving the way for mpc_agm)
Small things to do

See also TODO in the distribution or git.

- Simplify code for GNU MPFR $\geq 3$ (?)
- Improve test coverage (used to be 100%, now is 99.4%...)
- Move hard-coded tests to data files
  (for instance, tests/tcos.c)
Engineering type things to do

- Publish benchmarks:
  scripts and web page comparing timings with different systems, as done for GNU MPFR at
  http://www.mpfr.org/mpfr-3.1.2/timings.html
- Support the MPFR_RNDA rounding mode
Algorithmic things to do

- Karatsuba multiplication:  
  with 3 multiplications at precision around $p$,  
  instead of 2 at $2p$ and 1 at $p$

- Alternative formulæ for (inverse) trigonometric functions,  
  depending on (relative) argument sizes

- Correct handling of NaN and infinities in the case of intermediate overflows while the result may fit
New functions to implement

- Functions with reserved names in C99 and C11
  - `mpc_erf`, `mpc_erfc`
  - `mpc_exp2`, `mpc_expm1`
  - `mpc_log1p`, `mpc_log2`
  - `mpc_lgamma`, `mpc_tgamma`

- Number theoretic functions (see Arb)
  - Hurwitz zeta
New number type to implement

Complex ball arithmetic

- Used internally to replace hand calculations of error bounds
- As a tool to implement Taylor and Laurent series
- As a building block for polynomials, class polynomials, etc.
- Only four basic arithmetic operations and square root?
- Need for real ball arithmetic? interface with MPFI?
- Representation as rectangles? as balls?