MPFR – Main Features

C multiple-precision library based on GMP.

- MPFR number x: floating-point number in base 2, with precision $p_x \in [p_{\min}, p_{\max}]$ (where $p_{\min} = 2$) written

$$x = s_x \times m_x \times 2^{e_x} \quad (x \neq 0)$$

  with sign $s_x = \pm 1$, p_x-bit mantissa $m_x \in [1/2, 1[$, exponent $e_x \in [e_{\min}, e_{\max}]$ (exponent range fixed by the user).

- Supported functions: arithmetic operations and math functions of ISO C99.

- Portable, completely specified results: exact rounding (4 standard rounding modes), overflows, underflows, NaN. (*still in dev.*)

Consequences

- Reproducible and accurate results (exact rounding).
- Proofs of algorithms or bounds on results. Interval arithmetic (directed rounding modes), e.g. with MPFI.
- Test other implementations / emulate other arithmetics (except subnormals). See example.

Efficiency:

- The precision can be chosen by the user (e.g. increased dynamically).
- Very fast routines from GMP.

Example 1: Test of Double-Precision Impl.

Something like:

```
mpfr_t x, y, z;
double r;

mpfr_init2 (x, 53);
mpfr_init2 (y, 53);
mpfr_init2 (z, 53);
mpfr_exp (y, x, rnd);
r = mpfr_get_d1 (x);
r = exp (r);
mpfr_set_d (z, r, rnd);
```

Surprising...

```
#include <stdio.h>
#include <math.h>
#include <fenv.h>
int main (void)
{ double x, y;
  fesetround (FE_TOWARDZERO);
  x = 1.06337362123585; y = exp (x);
  printf ("exp(%.17g) = %.17g\n", x, y);
  return 0; }
```

gives (on my machine – PowerBook G4 under Linux, libm 2.2.5)

```
exp(1.06337362123585) = 2.0022568270516627
```

Example 2: Rump's Polynomial

$$f(a,b) = 333.75b^6 + a^2\left(11a^2b^2 - b^6 - 121b^4 - 2\right) + 5.5b^8 + \frac{a}{2b}$$

with $a = 77617.0$ and $b = 33096.0$.

On an IBM 370:

Single precision:	1.172603
Double precision:	1.1726039400531
Extended precision:	1.172603940053178

Example 3 (Jean-Michel Muller)

$$\begin{cases} u_0 & = & 2 \\ u_1 & = & -4 \\ u_{n+1} & = & 111 - \dfrac{1130}{u_n} + \dfrac{3000}{u_n u_{n-1}} \end{cases}$$

- u_n converges to 6 $\quad \left(u_n = \dfrac{4 \times 5^{n+1} - 3 \times 6^{n+1}}{4 \times 5^n - 3 \times 6^n} \right)$
- on *any machine* u_n seems to converge to 100 very quickly.

Example 4 (Jean-Michel Muller)

$$\begin{cases} x_0 & = & 1.510005072136258 \\ x_n & = & f(x_{n-1}) \quad \text{with} \quad f(x) = \dfrac{3x^4 - 20x^3 + 35x^2 - 24}{4x^3 - 30x^2 + 70x - 50} \end{cases}$$

MPFR web page:

`http://www.loria.fr/projets/mpfr/`

or

`http://www.mpfr.org/`

E-mail: `mpfr@loria.fr`