FLINT: Fast Library for Number Theory

Fredrik Johansson

Global Virtual Sage Days 112.358
June 2022

Topics

- About FLINT and related projects (Antic, Arb, Calcium)
- Development status: present, future
- Interfacing with FLINT

What is FLINT? (https://flintlib.org/)

```
sage: R.<x> = ZZ["x"]

sage: f = (x + 1)^10000

sage: %timeit f * f
523 ms ± 1.85 ms per loop (...)

sage: len(str(f * f))
87035463
```

FLINT programming

```
#include "flint/fmpz_poly.h"

int main()
{
    fmpz_poly_t f, g;

    fmpz_poly_init(f);
    fmpz_poly_init(g);

    fmpz_poly_set_coeff_si(f, 0, 1);  // f = 1
    fmpz_poly_set_coeff_si(f, 1, 1);  // f = 1 + x
    fmpz_poly_pow(f, f, 10000);       // f = f ^ 10000
    fmpz_poly_mul(g, f, f);           // g = f * f

    fmpz_poly_clear(f);
    fmpz_poly_clear(g);
}
```

FLINT feature highlights

- Base rings: $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}/n\mathbb{Z}$, $\mathbb{F}_q$, $\mathbb{Q}_p$
- Matrices (dense)
- Polynomials (dense univariate, sparse multivariate)
- Integer factorization, primality proving (best-of-breed algorithms including quadratic sieve, ECM and APRCL)
- Number-theoretic functions
- Many matrix operations: LU, FFLU, HNF, SNF, LLL, ...
- Special polynomial and power series functions
- Polynomial GCD and factorization (univariate and multivariate), multivariate ideal reduction
- Many fast multiplication algorithms (+ reduction of various operations to multiplication)

FLINT design philosophy

- Written in plain C
- Extensively tested
- Extensively documented
- Same public/private API
- Developer friendly
- Builds on top of GMP and MPFR
- Optimized for large operands (asymptotically fast algorithms)
- Optimized for small operands
- Multithreaded
- Verbose, fully-featured (currently 600,000 lines of code . . .)

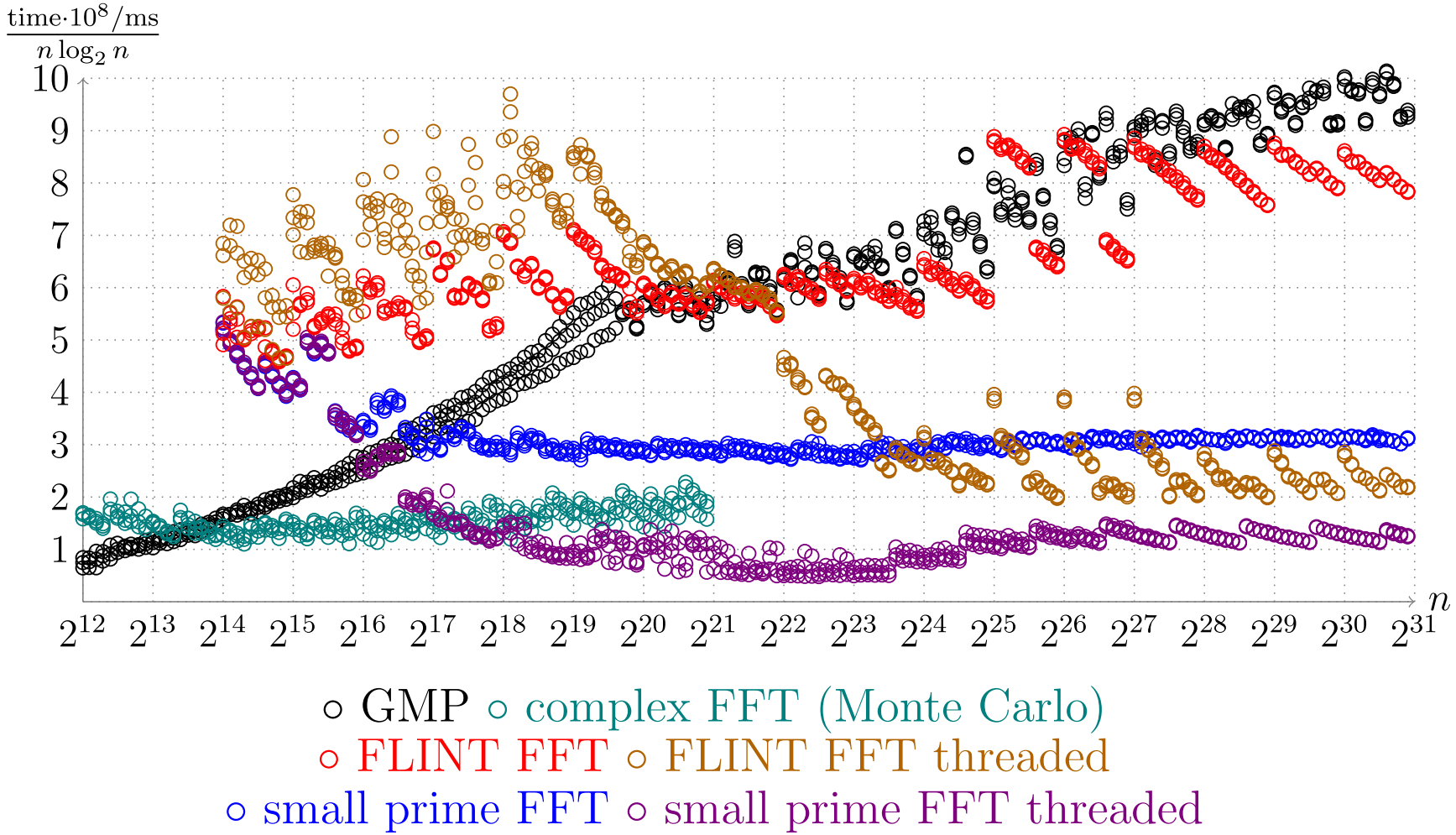
FLINT and GMP

Why our own integer and rational types (`fmpz`, `fmpq`)?

- An `fmpz` is only one word: values up to $\pm(2^{62} - 1)$ stored inline, otherwise becomes pointer to a GMP `mpz`. This is **significantly** faster for typical workloads.
- GMP still provides most of the low-level multi-word integer arithmetic.
- We provide far more functions on top of basic arithmetic, well beyond the scope of GMP.
- FLINT's Schönhage-Strassen FFT: slightly faster than GMP on one core, and multithreaded ($\approx$ 5x speedup on 8 cores)

FFT timings (courtesy of Daniel Schultz)

FIGURE 1. n bit integer product on desktop Coffee Lake



Antic, Arb and Calcium

Antic - `https://github.com/wbhart/antic/`

- Algebraic number fields
- Binary quadratic forms

Arb - `https://arblib.org/`

- Real and complex ball arithmetic
- Special functions, numerical analysis tools (integration, etc.)
- Additional number theory functionality (e.g. Dirichlet characters)

Calcium - `https://fredrikj.net/calcium/`

- Exact algebraic and transcendental numbers
- Extra FLINT utilities (symbolic expressions, multivariate rational functions, (slow) Gröbner bases)

What's new in FLINT 2.9 (upcoming)

Partial list:

- Optimized some `fmpz` functions for small inputs
- Speedups to `nmod` arithmetic
- Improvements to `fq_default` (use `nmod` where optimal)
- n-th derivative for $\mathbb{Z}[x]$, $\mathbb{Q}[x]$
- Eulerian polynomials; speedups for Stirling and Bell numbers
- Square root functions for various rings
- Solving for non-square/singular matrices over $\mathbb{Q}$
- Support "multivariate" polynomials with zero variables
- FFT matrix multiplication
- Parallel programming helpers

Currently active FLINT developers

Bill Hart
(maintainer, 2007-2022)

Me (2010-)

Daniel Schultz
(2017-2022)

Albin Ahlbäck (2021-)

Past authors

- David Harvey (polynomial multiplication)
- Andy Novocin (LLL and polynomial factorization)
- Sebastian Pancratz (polynomials, p-adics, matrices)
- Mike Hansen, Andres Goens (finite fields)
- Abhinav Baid, Curtis Bright (LLL)
- Alex Best (HNF, SNF, linear algebra improvements)
- Martin Lee, Lina Kulakova (polynomial factorization)
- Tom Bachmann (C++ interface)
- Luca De Feo, Edouard Rousseau (finite field embeddings)
- Kushagra Singh (ECM), Vladimir Glazachev (APRCL)

Plus dozens of other contributors listed on
`http://flintlib.org/authors.html`

Future FLINT development and funding?

- With Bill and Dan stepping down this year, I will take over maintenance and direction of FLINT.
- Near term: no direct backing by a major grant like OpenDreamKit, OSCAR. I have tried to apply for small grants to help develop Arb & Calcium without success.
- We've had 7-8 very good GSoC students (2012-2015). None continued working on FLINT after the project. Crucial to have good mentors.
- Inria can potentially allocate "research engineers" for targeted short-term projects. I'm not sure how useful this is.
- Future of MPFR is also uncertain. MPFR has two active developers: Paul Zimmermann and Vincent Lefèvre, and Paul is retiring in a few years. GMP has three active developers.

Development plans

- Performance optimization
  - Better algorithms and implementations
  - Multithreading, improvements for modern hardware
- More math functions
- Code cleanup
- Possible merger of FLINT/Antic/Arb/Calcium?
- Generics
- Interfaces (including Sage/Python)

More multithreading

FLINT has been threadsafe since 2009, extensively multithreaded (polynomial multiplication, integer factorization, etc.) since 2020.

`flint_set_num_threads(N)` makes FLINT up to N times faster

Note: it looks like Sage does not yet wrap this function

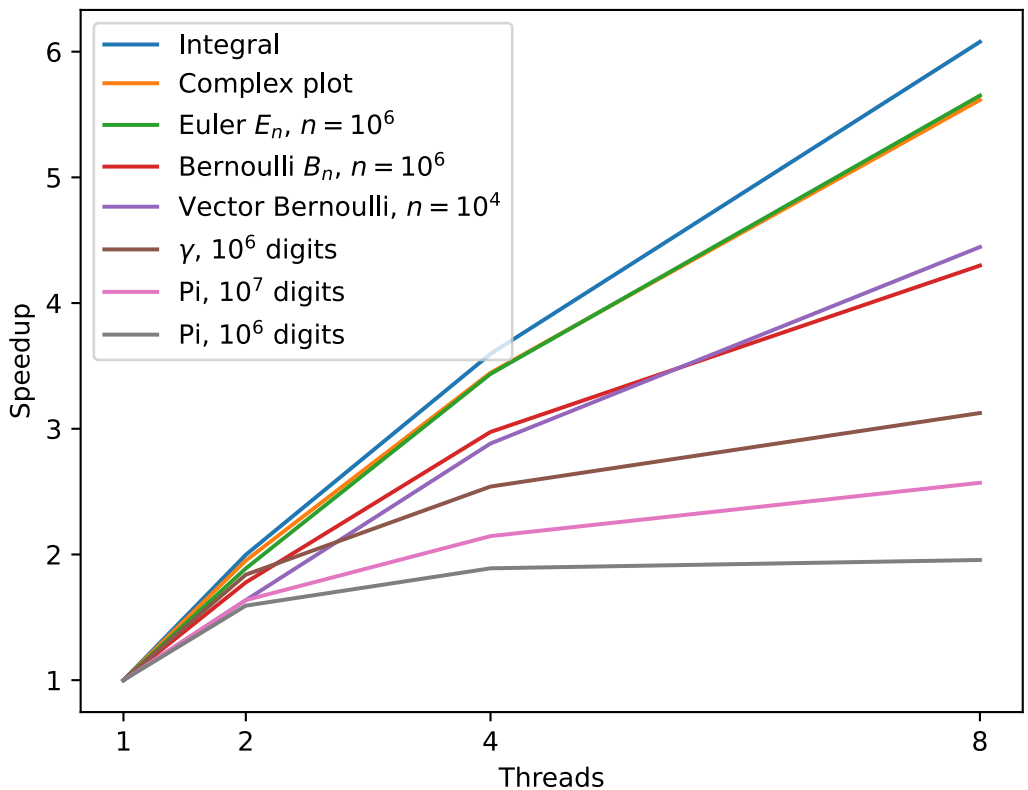
New parallel programming helpers:

- `flint_parallel_do`
- `flint_parallel_binary_splitting`

Upcoming parallelization in Arb

Timing examples (on my 8-core laptop, Zen3):

- Integration of $\int_0^8 \sin(x + e^x)dx$ to 1000 digits: 2.4 s $\rightarrow$ 0.4 s
- 10^6-th Bernoulli number: 20.5 s $\rightarrow$ 4.8 s
- $\exp(x)$ with 10 million digits: 20 s $\rightarrow$ 4.5 s



Vectorization, "extended-precision" arithmetic

- Arbitrary-precision types are not suited for vector processing (SIMD, GPU)
- Convert to vector-friendly representations
  - Already done in FLINT e.g. for BLAS matrix multiplication
  - NTL uses vectorized modular arithmetic (AVX2) and beats FLINT in some ranges: see Victor Shoup's comparison `https://libntl.org/benchmarks.pdf`
- Idea for Arb: double and double-double ball arithmetic

Generic rings in C (experimental)

https://github.com/fredrik-johansson/generic-rings

Parent (context object) + element (void pointer) model

```
gr_ctx_t ZZ, ZZx;
gr_ptr f;
int status;

gr_ctx_init_fmpz(ZZ);
gr_ctx_init_polynomial(ZZx, ZZ);
GR_TMP_INIT(f, ZZx);

status = gr_set_si(f, 3, ZZx);
status |= gr_pow_ui(f, f, 10, ZZx);
gr_print(f, ZZx);
```

Goals and benefits

- Wrap existing FLINT/Antic/Arb/Calcium types
- Complement existing specialized types with generic recursively constructed matrices, polynomials, multivariate polynomials, fraction fields, power series...
- Support all unusual cases in FLINT/Antic/Arb/Calcium (error handling, inexact rings, noncomputable rings, context objects) **mathematically correctly** and **with a uniform interface**
- Plain C, similar programming model to existing FLINT code. Small code size, fast compilation. Allows streamlining FLINT?
- Possible to pack data efficiently (down to 1 byte / element)

Interfacing with FLINT/Antic/Arb/Calcium

- FLINT wrappers in Sage (Cython)
  - Highly incomplete
- Python-FLINT (Cython)
  - Also incomplete, in need of basic maintenance
- Nemo + AbstractAlgebra (Julia)
  - Most complete and most efficient wrapper
  - Status of Python-Julia interoperability?
- Other wrappers
- C generics (interface to FLINT in FLINT) potentially helpful?