(Fast special function computations with) FLINT

Fredrik Johansson

RISC

2012-05-08

Contents

- What is FLINT?
- What does FLINT do?
- Some example computations
- What does FLINT not do (but in the near future, perhaps might)?

Overview of FLINT (Fast Library for Number Theory)

- C library with about 130,000 lines of code
- Main authors: Bill Hart, Sebastian Pancratz, Andy Novocin, myself, David Harvey (past author)
- Arithmetic over various base rings: $\mathbb{Z}$, $\mathbb{Z}/n\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}_p$
- Efficient polynomial arithmetic over all base rings
- Linear algebra over all base rings (experimental)

History of FLINT

Started in 2007 by Bill Hart and David Harvey as a successor to Victor Shoup's NTL.

NTL was no longer maintained, did not support parallel computing well, and many of its algorithms were getting out of date.

In particular, Magma was several times faster than NTL for multiplication in $\mathbb{Z}[x]$.

The first goal (beating Magma on polynomial multiplication) took several months of work.

More features: modular arithmetic, integer factorization, LLL and polynomial factorization (Andy Novocin), modular arithmetic

History of FLINT 2

In 2009-2010, Bill Hart rewrote the core functionality of FLINT from scratch (creating FLINT 2).

Much simpler code, in many cases better performance.

Sebastian Pancratz added support for $\mathbb{Q}[x]$, $\mathbb{Q}(x)$, $(\mathbb{Z}/n\mathbb{Z})[x]$ for large n, and $\mathbb{Z}_p$.

I added power series, miscellaneous polynomial functionality, linear algebra, special functions.

Software using FLINT

Sage uses FLINT 1.x for $\mathbb{Z}[x]$ and $(\mathbb{Z}/n\mathbb{Z})[x]$

```
sage: a = ZZ['x'].random_element(degree=10^6)
sage: %time a ^ 2;
CPU times: user 1.22 s, sys: 0.10 s, total: 1.32 s
Wall time: 1.35 s
```

Singular is switching from NTL to FLINT 2.x for polynomial GCD and factorization (work in progress)

Features: numbers

- Optimized code for integers smaller than 2^{64}
- Primality testing, integer factorization, modular arithmetic
- FFT for huge integer multiplications (about 30% faster than GMP)
- Asymptotically fast multimodular reduction / Chinese remaindering
- Fast computation of various combinatorial and number-theoretic integer sequences (divisor sum-type functions, partitions, Bernoulli/Stirling/Bell numbers)

Features: univariate polynomials

- Multiplication (Kronecker substitution, Schönhage-Strassen)
- Division (divide-and-conquer, Newton iteration)
- GCD (hgcd, heuristic, multimodular)
- Composition, reversion
- Power series special functions (sqrt, exp, log, sin, ...)
- Multipoint evaluation and interpolation
- Restartable Hensel lifting
- Factorization (Berlekamp, Cantor-Zassenhaus)

Features: linear algebra

- Dense matrices mod p, $p < 2^{64}$
- Fast multiplication (basecase assembly, Strassen)
- Block recursive LU factorization, inverse, determinant, nullspace
- Matrices over $\mathbb{Z}$, $\mathbb{Q}$, mostly fast (but not nullspace)
- Fast multiplication and determinants of polynomial matrices

Some timings: $\mathbb{Z}[x]$ arithmetic

P, Q, R have degree n and coefficients with n digits

$n = 1000$

PQ	Mathematica: 7.2 s / 0.78 s	FLINT: 0.05 s
$\gcd(PQ, PR)$	Mathematica: 25 s	FLINT: 2.2 s

$n = 10000$

PQ	Mathematica: ? / 2500 s	FLINT: 10 s
$\gcd(PQ, PR)$	Mathematica: $>$ 15 h	FLINT: 858 s

Some timings: $(\mathbb{Z}/p\mathbb{Z})[x]$ arithmetic

P, Q, R have degree n and modulus $p = 2^{63} + 29$

$n = 1000$

PQ	Mathematica: 1.6 s / 0.0058 s	FLINT: 0.0012 s
$\gcd(PQ, PR)$	Mathematica: 0.55 s	FLINT: 0.017 s

$n = 10000$

PQ	Mathematica: 163 s / 0.082 s	FLINT: 0.02 s
$\gcd(PQ, PR)$	Mathematica: 24 s	FLINT: 0.59 s

Some timings: power series

Computing $\log(1 + \log(1 + x + x^2)) \bmod x^n$

n	Mathematica	FLINT
100	0.06 s	0.001 s
1000	94 s	0.25 s
2000	1082 s	1.6 s
10000	a long time	97 s

Computing $f(f(x)) \bmod x^n$ where $f = x + 2x^2 + 3x^3 + \ldots$

n	Mathematica	FLINT
100	0.4 s	0.001 s
1000	692 s	0.44 s
10000	a very long time	194 s

Some timings: linear algebra

Nullspace of $n \times 2n$ matrix modulo a 10-bit prime

n	Mathematica	FLINT
100	0.024	0.003 s
500	2.6 s	0.3 s
1000	17 s	2 s
5000	1400 s	200 s

Nullspace of $n \times 2n$ matrix modulo a 60-bit prime

n	Mathematica	FLINT
100	1.4 s	0.005 s
500	184 s	0.3 s
1000		3 s
5000		300 s

Power series arithmetic for special functions

Goal: compute the first n coefficients in a sequence $(c_k)_{k=0}^{\infty}$ with generating function $f(x) = c_0 + c_1 x + c_2 x^2 + \ldots$

Many uses:

- Guessing a recurrence for c_k
- Investigating the numerical growth of c_k
- Investigating the divisibility properties of c_k
- Numerical evaluation of $f(x)$

Example: testing Wilf's Bell number conjecture

The *complementary Bell numbers* are defined by

$$\sum_{n=0}^{\infty} \frac{c_n}{n!} x^n = \exp\left(1 - e^x\right)$$

$$\{c_n\}_{n=0}^{\infty} = 1, -1, 0, 1, 1, -2, -9, -9, 50, 267, \ldots$$

Wilf conjectured that c_2 is the only zero in this sequence.

It is known that at most one more zero exists.

I verified that there is no other zero below $n = 10^9$ (a single power series exponential mod $p = 10^{12} + 39$ with FLINT, 5 hours CPU time)

Example: high-precision numerical evaluation of $\Gamma(x)$

For small real x, the Taylor series

$$\frac{1}{\Gamma(x)} = c_1 x + c_2 x^2 + \ldots$$

seems to be much faster in practice than any other method. The coefficients can be computed from

$$\log \Gamma(1+x) = -\gamma x + \sum_{k=2}^{\infty} \frac{\zeta(k)}{k} (-x)^k.$$

Coefficients for 1000-digit precision take 1 s.

I have computed 10000 coefficients (precision: 50000 digits).

Very large computations with FLINT

Congruent numbers $5, 6, 7, 13, 14, 15, 20 \ldots$: n for which there exists a right triangle having area n and rational sides.

Bradshaw, Hart, Harvey, Tornaria, Watkins (2009) computed all congruent numbers up to 10^{12} (subject to the Birch and Swinnerton-Dyer conjecture).

Required special-purpose multiplication code for operands on disk.

Very large computations with FLINT, 2

The integer partition function $p(n) = 1, 1, 2, 3, 5, 7, 11, 15, 22, 30, 42 \ldots$ counts the number of ways that n can be written as a sum of positive integers.

Work by myself last year: fast computation of $p(n)$. Modular arithmetic and high-precision numerics (using FLINT arithmetic).

Computation of $p(10^{19})$ (3.5 billion digits).

Tabulation of 22 billion congruences. (470K evaluations for n up to 10^{13}). Example: $p(289952442924860052459470 69k + 28995221336976431135321047) \equiv 0 \bmod 29$.

What about multivariate polynomials?

Right now, nothing is supported in released versions of FLINT.
Experimental code for sparse polynomials and recursive dense polynomials exists.

Only multiplication (various algorithms) is implemented.

Early benchmarks show promising performance.

Multivariate polynomial multiplication

Algorithms:

- Recursive (univariate multiplication as basecase)
- KS (pack the whole polynomial into one huge integer)
- Flat (create a huge array for all possible monomials)
- Hybrid (select algorithm depending on size and density)

Fateman benchmark: $f \times (f+1)$ where $f = (1 + x_1 + \ldots + x_m)^n$

m	n	Mathematica	Recursive	KS	Flat	Hybrid
5	10	594 s	0.21 s	1.9 s		0.15 s
4	20	443 s	1.2 s	1.2 s		0.4 s
3	40	582 s	0.6 s	0.2 s		0.6 s
2	5	1.6 ms	7.2 µs	3.9 µs	2.7 µs	2.9 µs
2	500	> 1 h	6 s	0.46 s	22 s	0.47 s

Example: convert DE to RE

Input: differential operator of order n, with coefficients in $\mathbb{Z}[x]$ having degree n and n-bit coefficients.

DFiniteDE2RE (HolonomicFunctions):
n = 10: 0.38 s

n = 100: 156 s

FLINT:
n = 10: 16 µs

n = 100: 0.04 s

n = 1000: 164 s

This is not taking advantage of fast arithmetic (complexity could be reduced with fast Newton basis conversion).

Can we use FLINT to speed up symbolic summation?

Now: univariate polynomials, dense linear algebra mod p

Possibly in the near future: (dense) multivariate polynomials

Little progress (but would be useful): sparse linear algebra/polynomials

Update FLINT in Sage; Mathematica/MathLink interface?